Guide to FeynCalc

A Mathematica package

for Quantum Field Theory practitioners

Rolf Mertig, Frederik Orellana and Vladyslav Shtabovenko

Preface

This book is for practitioners of calculations in perturbative QCD, the Standard Model, Chiral Perturbation Theory and other quantum field theories. The intention is to serve educational as well as research purposes. This is reflected in the organization of the text: The first part is a tutorial-like introduction with many small examples. It is necessarily of a rather technical character and should prepare the reader for doing concrete physics calculations. The second part contains a variety of example calculations from textbook to research level. The interested student will here find calculations from the literature, worked through to a detail not usual in textbooks. The researcher will find calculations that should provide a starting point for calculations in his or her own field of research. The third part is a reference guide, listing all important functions, along with smaller examples.

Over the years, the detailed course of and motivation for the development of FeynCalc has changed somewhat both in scope and focus. Reasons for this include the different research interests of the authors, the relative slowness of Mathematica and the development of multi-loop calculations with other scientific software packages - in particular FORM and programs based on FORM [1, 2]. While such packages are typically faster than FeynCalc and thus suited for very large scale calculations, we believe FeynCalc has kept relevance by being a relatively comprehensive and complete framework while also exhibiting the user-friendliness and versatility of a pure Mathematica package.

None the less, some effort has been put in the development of FeynCalc tools, precisely for interacting with FORM. These tools allow taking advantage of FORM's efficient handling of large expressions, which can be simply unmanageable with Mathematica, while staying within the Mathematica system, which as a general purpose computer algebra system, has an intuitive and flexible user interface and built-in mathematical knowledge.

Another focus has been to explore other types of calculations than the evaluation of Feynman diagrams, e.g. complex algebraic manipulations of quantum fields and lagrangians, traditionally done by hand only. Such calculations have become feasible because of the steep increase of the computing power of commodity hardware.

Despite the shifting motivations driving development, the central vision remains largely the same: FeynCalc was conceived as and continues to be developed as a general-purpose, relatively easy to use, modular toolbox for quantum field theory calculations. Dirac algebra utilities, Lorentz and color algebra manipulation capabilities, functional derivation utilities, integral and other tables are encapsulated in distinct functions which do well-defined operations. Such basic functions can then be used on a higher level to build sophisticated model dependent functions.

The most important general principles for quantum field theory calculations that the FeynCalc project strives to promote are:

- **Systematic knowledge build-up.** In physics calculations, errors due to typos or other trivial mistakes are not acceptable, therefore the code used should be as thoroughly checked as possible by as many people as possible. This implies openness and adherence to standards. It is proposed to build up a repository of lagrangians, amplitudes and integrals, stored in a standardized electronic form.

- **Prototyping.** Electronic tools should be put to use for making it easy to try out new quantum field theory models. This should be achieved by providing predefined building blocks capturing the more general physics and the more common calculational tasks.

- **Checking.** Calculations should, when possible, be performed by different methods. Thus, FeynCalc should be used to check hand calculations or the results of other programs and vice versa.

FeynCalc is an open-source project and contributions are warmly welcomed. These are best done by checking out a working copy of code on Github and then getting in touch via the website. Though much testing of the code has been done, there is absolutely no claim that FeynCalc is bug free. Users are encouraged to be sceptical about results, and when sure that the program returns a wrong answer, to report it via the website.

Version History

The roots of FeynCalc go back to 1987. During a stay as a graduate student in Albuquerque, New Mexico, Rolf Mertig learned to program in Macsyma [4] from the experts Stanly Steinberg and Michael Wester. Back in Germany, elementary particle physicists needed automation of the calculation of Feynman diagrams of eletroweak processes to one loop. The algorithms were partially developed together with Ansgar Denner and Manfred Böhm and implemented in a purely functional way by Rolf Mertig during the years 1987 - 1989.

The basic idea was to have general functions for some of the more mechanical parts of the diagram calculations, generalizable tools for use in calculations of different 1-loop processes, especially 1→2 and 2→2 processes. These included tools for:

- Lorentz algebra. $g^{\mu\nu}\, p_\mu \;\to\; p^\nu\,,\;\; g^{\alpha\beta} g_{\alpha\beta} \;\to\; D \cdots$,
- Dirac algebra. $\gamma^\alpha\, \gamma^\nu \gamma_\alpha \;\to\; (2-n)\, \gamma^\nu,\;\; \text{v}(p,m)\, \not{p} \to$ - m $\text{v}(p,m) \cdots$,
- Passarino-Veltman tensor integral decomposition. Applications of these tools included complete 1-loop processes in the Standard Model, $\text{e}^+\text{e}^- \;\to\; \text{Z H}\,,\; \text{e}^+\text{e}^- \;\to\; \text{W}^+\text{W}^-$ and the 2-loop photon self-energy in QED.

At the end of 1989 several problems showed up with the Macsyma implementation. The purely functional programming style proved to be difficult to debug and, in fact, inappropriate. The rudimentary pattern matcher in Macsyma was not adequate. There was no way to incorporate new functions easily into the whole Macsyma system, and no possibility of providing online documentation. Furthermore, Macsyma's memory management ("garbage collection") was slow when handling large expressions.

In early 1990 it became clear that Mathematica was a much more natural programming environment for FeynCalc.

1990-1991 : The first version of FeynCalc in Mathematica

In 1990 user-friendly packages were built with extended automatic capabilities (**`OneLoop`**, **`OneLoopSum`**), and SU(3) algebra capabilities were added (**`SU3Simplify`**).

In 1991 initial documentation was written and the program was made available on anonymous ftp-servers: mathsource.wolfram.com and canc.can.nl.

Applications from 1990 - 1996 included:

- 1-loop 2 →2 processes in the Standard Model, such as:
  $\text{gg} \;\to\; \text{t}\bar{\text{t}}\,,\;\; \text{e}^+\text{e}^- \;\to\; \text{Z H}\,,\;\; \text{gg} \to \text{t}\bar{\text{t}}$,
- background field gauge calculations,
- high-energy approximation of $\text{e}^+\text{e}^- \;\to\; \text{W}^+\text{W}^-$,

- 2-loop Standard Model self-energies.

No real attempt was made to provide tools for tree-level calculations. For this purpose other programs appeared, among them another Mathematica package, HIP [5], developed at SLAC, CompHEP [6], and of course, FORM [1].

1992-1995 : FeynCalc 2.0-2.2, unification and simplification.

During this period of development the SU(3) algebra was changed to SU(N). Several tools for automatic tree-level calculation were added, for example the function **`SquareAmplitude`**. (Unfortunately, the documentation was not updated.) All sub-packages were put into one file ($\simeq 10^4$ lines), "FeynCalc2.2beta.m" [1].

1993-1996 : FeynCalc 3.0, modularization, typesetting, Operator Product Expansion

Due to the rapidly increasing amount of code, totalling 2.5 megabytes, FeynCalc was reorganized in a modular way. The definitions of each function were contained in a package file which was loaded only on demand.

- In these years many improvements were made to the code, driven by the work of the author (Rolf Mertig) together with Willy van Neerven in perturbative QCD.

  One effort was to build databases: Convolutions, integrals, tensor integral transformation formulae, Feynman rules, and Feynman parameterizations. Applications include 2-loop spin-independent and spin-dependent Altarelli-Parisi splitting functions [7].

- The new typesetting capabilities of Mathematica 3 (TraditionalForm) were used to substantially improve the look of the output. Typesetting rules were added for e.g. $\vec{\partial}_\mu$.

- Code was written to allow new abstract data types, for example for noncommutative algebra and for special integrals.

- QCD tools for OPE were added.

- Automatic Feynman rule derivation (by functional derivation) was implemented in order to get special Feynman rules for twist-2 (and higher) operators.

1997-2000 : FeynCalc 4.0-4.1, QCD and OPE, ChPT, tables

The package TARCER, which was initially an independent project, was integrated into FeynCalc. TARCER [8] adds two-loop functionality for propagator-type integrals using the recurrence relations of Tarasov [9].

In 2000, Frederik Orellana, then a PhD student at the University of Zurich, started contributing to the package. He worked in Chiral Perturbation Theory [10] (ChPT) and made some changes, as well as adding code to FeynCalc in order to support ChPT and effective theories in general, and interfacing with FeynArts. Eventually, he integrated his package, PHI, into FeynCalc. The ambition of PHI is to allow full automation of Feynman diagram calculations, starting from a Lagrangian - as opposed to starting from a FeynArts model definition, i.e. to automatize the generation of a FeynArts model and to allow the FeynArts output to be directly used by FeynCalc.

[1] This file is still available upon request

2000-2014 : FeynCalc 5-8, maintenance

With both authors out of theoretical physics, no new developments were undertaken, but the mailing list remained active and bugs were fixed.

2014 : FeynCalc 9, cleanup, testing

In 2014, Vladyslav Shtabovenko, has joined the team. He is working on effective theories in quarkonium production as a PhD student at the Technical University of Munich. To FeynCalc, he has contributed work on Dirac gamma schemes and a unit testing framework.

Contents

Chapter 1

The FeynCalc Framework

1 Introduction

FeynCalc is a Mathematica package that provides a framework for doing calculations in quantum field theory and, within this framework, a collection of utilities for the more common as well as some more specialized tasks in such calculations.

One typical task is the calculation of radiative corrections in the Standard Model of particle physics. The input for FeynCalc, the analytical expressions for the diagrams, can be entered by hand or can be taken directly from the output of the Feynman diagram generator FeynArts [3] . The user can provide certain additional information about the process under consideration, i.e. the kinematics and the choice of the standard matrix elements may be defined. Once this is done, FeynCalc performs the algebraic calculations like tensor algebra, tensor integral decomposition and reduction, yielding a polynomial in standard matrix elements, special functions, kinematical variables and scalars suitable for further numerical evaluation.

FeynCalc also provides calculator-like features. These features include basic operations like contraction of tensors, simplification of products of Dirac matrices and trace calculation. For instance, a Dirac trace is entered in a notation very similar to that used when doing calculations by hand, and an answer suitable for further manipulation is returned.

More complex algebraic operations have also been implemented, notably derivation of polynomials in quantum fields with respect to such fields.

FeynCalc is installed by following the steps below (modify version number to match the newest release).

- Download the file "HighEnergyPhysics-9.0.tar.gz" or "HighEnergyPhysics-9.0.zip".
- If you already have an old installation and have customized the file "FCConfig.m", back it up.
- Unpack the archive in any of the following places:
  - Anywhere on your file system. If the location is not on the Mathematica**$Path** you have to load FeynCalc by **«/myfullpathname/HighEnergyPhysics/FeynCalc.m**.
  - One of the Mathematica "Applications" directories: For UNIX orGNU/LINUX and Mathematica 4.2 or higher, "~/.Mathematica/Applications/" (the tilde refers to you home directory). For other operating systems: evaluate **$UserAddOnsDirectory** or **$AddOnsDirectory** in Mathematica.

- If you've made any customizations in the configuration file "FCConfig.m", merge them from the file you've moved away into the new file.
- Start Mathematica.
- Choose 'Rebuild Help Index' from the 'Help' menu.
- Load FeynCalc with **«HighEnergyPhysics`FeynCalc`** or **«HighEnergyPhysics`FeynCalc`**.

2 Input Functions

The FeynCalc syntax for metric tensors, four-vectors, Dirac matrices, etc., is such that the positioning of the arguments automatically defines the nature of the variables. No declaration of Lorentz indices or four-vectors has to be made. Covariant and contravariant Lorentz indices are treated on an equal basis, assuming summation over (twice) repeated indices, where each pair of indices are assumed to be co- and contravariant respectively. Since explicit components of Lorentz tensors are not used, this convention causes no problems. The ambiguity in the sign of the Levi-Civita tensor can be fixed by an option of the function **DiracTrace** and **EpsChisholm**, see page 45.

The input functions are macros that are immediately translated into the internal representation of FeynCalc. But these sometimes lengthy internal representations are difficult to recognize when working interactively. Therefore, some effort has been put into making the output of FeynCalc look as much as possible like typeset formulas. The typesetting of FeynCalc depends on the input/output interface in use.

- **Terminal interface.** Per default FeynCalc sets the global Mathematica variable **$PrePrint** to **FeynCalcForm**, forcing the display to be in a more readable manner. It is important to realize that the display given by **FeynCalcForm** is different from the input syntax. Thus, you may not give as input, for example, a scalar product in the result of a trace calculation by copying the form you see on the screen. To have the output be the internal representation, set **$PrePrint=.**. To return to formatted output, set **$PrePrint = FeynCalcForm**.
- **Notebook interface.** FeynCalc per default automatically sets the Mathematica default output format type to **TraditionalForm**. Because FeynCalc defines the **TraditionalForm**-look of its objects, the output looks very much like typeset formulas. Again, the screen output cannot be copied and used as input. If the Mathematica default output format type [1] is set to **InputForm** or **StandardForm**, the screen output is the internal representation of FeynCalc and can be copied and used as input. To obtain the same output as with a terminal interface, set the Mathematica default output format type to **OutputForm** and set **$PrePrint = FeynCalcForm**. To return to formatted output, set the Mathematica default output format type to **TraditionalForm**.

In the rest of this guide the examples given will use the notebook interface.

The internal structure of FeynCalc is explained in section 3 and in the reference part 3. Usually it is not necessary to know it. Notice simply that there are three layers of FeynCalc: The input functions, an output representation and the internal representation.

In order to do calculations in the framework of D-dimensional regularization, FeynCalc can be used for calculations in 4 or D dimensions. The dimension of metric tensors, Dirac matrices and four-vectors is specified by setting the option **Dimension** of the corresponding input functions. The default is four. If FeynCalc is used for interactive D-dimensional calculations, you should change the option of the input functions. Note that Lorentz

[1] The Mathematica default output format type is set using the menu 'Cell' → 'Default Output Format Type'.

indices, momenta and Dirac matrices carry their dimension locally, i.e., for calculating a Dirac trace in D dimensions the dimension of the Dirac matrices has to be set accordingly, thus there is no need of a dimension option for e.g. the function **DiracTrace**.

2.1 Lorentz Tensors and Scalar Products

FourVector[p, μ**]**	four-vector p^μ
LeviCivita[μ, ν, ρ, σ**]**	Levi-Civita tensor $\varepsilon^{\mu\nu\rho\sigma}$
LeviCivita[μ, ν**][**p, q**]**	short form for $\varepsilon^{\mu\nu\rho\sigma}\, p_\rho q_\sigma$
MetricTensor[μ, ν**]**	metric tensor $g^{\mu\nu}$
ScalarProduct[p, q**]**	$p \cdot q$

Basic Lorentz tensor input functions.

FV[p, μ**]**, **FVD[**p, μ**]**	4- and D-dimensionsal p^μ
LC[μ, ν, ρ, σ**]**, **LCD[**μ, ν, ρ, σ**]**	4- and D-dimensional Levi-Civita tensor $\varepsilon^{\mu\nu\rho\sigma}$
MT[μ, ν**]**, **MTD[**μ, ν**]**	4- and D-dimensional metric tensor $g^{\mu\nu}$
SP[p, q**]**, **SPD[**p, q**]**	4- and D-dimensional $p \cdot q$

Short input forms for basic Lorentz tensor input functions.

option name	*default value*	
Dimension	**4**	number of space-time dimensions

Option for **FourVector**, **MetricTensor**, **LeviCivita** and **ScalarProduct**.

Internally, the input given with **FourVector**, **MetricTensor**, **LeviCivita** and **ScalarProduct** is represented in terms of the lower level functions **Pair**, **Momentum** and **LorentzIndex** (this is elaborated in section 3).

The following examples show the input syntax and the typesetting of FeynCalc.

First we have to load FeynCalc.

```
In[1]:= «HighEnergyPhysics`FeynCalc`
```

Enter a scalar product $p \cdot q$. This is the typeset output.

```
In[2]:= ScalarProduct[p, q]
Out[2]= p.q
```

This is the input for $g^{\mu\nu}$.

```
In[3]:= MetricTensor[μ, ν]
Out[3]= g^μν
```

A metric tensor $g^{\mu\nu}$ in D dimensions is entered in this way. The number of dimensions is suppressed in the screen output (for notebook output. For terminal output, the number of dimensions is displayed as an index, **g_D[mu, nu]**).

```
In[4]:= MetricTensor[μ, ν, Dimension → D]
Out[4]= g^μν
```

A four-dimensional p_μ.

```
In[5]:= FourVector[p, μ]
Out[5]= p_μ
```

You may also enter a linear combination of four-vectors: $(p - 2\,q)_\mu$.

```
In[6]:= FourVector[p - 2 q, μ]
Out[6]= (p - 2 q)_μ
```

This is a D-dimensional q_μ.

```
In[7]:= FourVector[q, μ, Dimension → D]
Out[7]= q_μ
```

A polarization vector $\varepsilon_\mu(k)$ is a special four-vector.

```
In[8]:= PolarizationVector[k, μ]
Out[8]= ε_μ(k)
```

This is how to enter a $\varepsilon^*_\mu(k)$.

```
In[9]:= Conjugate[PolarizationVector[k, μ]]
Out[9]= ε*_μ(k)
```

Polarization vectors are represented in a special way (see page 45). The transversality condition $\varepsilon(k) \cdot k = 0$ is automatically fulfilled (see page 35). Polarization vectors are defined in FeynCalc in four dimensions only. For **FourVector**, **MetricTensor**, **LeviCivita**, and **ScalarProduct** other dimensions may be specified with the option **Dimension**.

```
PolarizationVector[k, mu]                    ε_μ(k)
Conjugate[PolarizationVector[k, mu]]         ε*_μ(k)
```

The input for a polarization vector and polarization.

2.2 Dirac Matrices

A Dirac matrix γ^μ is entered like **DiracMatrix[**μ**]**. For $\not{p} = p_\mu \gamma^\mu$ you may use **DiracSlash[p]**. Products of Dirac matrices or slashes are entered either by adding subsequent arguments, i.e., **DiracMatrix[***mu***,** *nu***, ...]** and **DiracSlash[***p***,** *q***, ...]**, or by multiplying the **DiracMatrix** and **DiracSlash** with the Mathematica**Dot**, "**.**".

DiracMatrix[*mu***, *nu*, ...]**	Dirac matrices $\gamma^\mu\ \gamma^\nu \ldots$
DiracSlash[p, q, ...]	Feynman slashes $\not{p}\not{q}\ldots$
DiracMatrix[5]	γ^5
ChiralityProjector[+1]	$\omega_+ = (1+\gamma^5)/2$
DiracMatrix[6]	$\gamma^6 = (1+\gamma^5)/2$
ChiralityProjector[-1]	$\omega_- = (1-\gamma^5)/2$
DiracMatrix[7]	$\gamma^7 = (1-\gamma^5)/2$

Various input functions for Dirac matrices.

The Mathematica**Dot**, "**.**", is used in the input as noncommutative multiplication operator for the objects **DiracMatrix**, **DiracSlash**, **Spinor**, **LeptonSpinor**, **QuarkSpinor** and **SUNT**. The "**.**" may also be used as a delimiter instead of the "**,**" within the functions **DiracMatrix**, **DiracSlash** and **SUNT**.

In the output with **FeynCalcForm** the "**.**" as noncommutative multiplication operator is suppressed.

This is how you enter $(\not{p}+\not{q}+m)\gamma^\mu$.

```
In[10]:= (DiracSlash[p + q] + m) . DiracMatrix[μ]
Out[10]= (m + γ · (p + q)) γ^μ
```

Here is $\gamma^\mu \gamma^5 \gamma^\nu \gamma^6 \gamma^\rho \gamma^7$.

```
In[11]:= DiracMatrix[μ, 5, ν, 6, ρ, 7]
Out[11]= γ^μ γ^5 γ^ν γ^6 γ^ρ γ^7
```

This is a four-dimensional product: $2\not{b}\not{a}2\,(\not{d}-\not{c})\,(6\not{q}-3\not{p})$.

```
In[12]:= DiracSlash[2b, a, 2(d - c), 6q - 3p]
Out[12]= 2γ · b  γ · a  2 (γ · d - γ · c)  (6 γ · q - 3  γ · p)]
```

Slashed polarization vectors $\not{\varepsilon}(k)$ are entered in this way.

```
In[13]:= DiracSlash[Polarization[k]]
Out[13]= γ · ε(k)
```

option name	*default value*	
Dimension	**4**	space-time dimension

Option for **DiracMatrix** and **DiracSlash**.

As setting for **Dimension**, a Mathematica**Symbol** *dim*, or *dim*-4 are possible.

Entering of a D-dimensional γ^μ. The number of dimensions is suppressed in the screen output (for notebook output).

```
In[14]:= DiracMatrix[μ, Dimension → D]
Out[14]= γ^μ
```

This is $\gamma^\mu \gamma^\mu$ in $D-4$ dimensions.

```
In[15]:= DiracMatrix[mu, mu, Dimension → D - 4]
Out[15]= γ^μ γ^μ
```

2.3 Spinors

In FeynCalc spinors are entered with the four functions **SpinorU**, **SpinorUBar**, **SpinorV**, **SpinorVBar**. Products of spinors with spinors or Dirac matrices (Fermion chains) are built with the Mathematica**Dot**, "**.**".

SpinorU[*p*, *m***]**	$u(p, m)$
SpinorUBar[*p*, *m***]**	$\overline{u}(p, m)$
SpinorV[*p*, *m***]**	$v(p, m)$
SpinorVBar[*p*, *m***]**	$\overline{v}(p, m)$

The input functions for fermionic spinors.

This is $\overline{u}(p, m) \not{p}$.

```
In[16]:= SpinorUBar[p, m] . DiracSlash[p]
Out[16]= u̅(p, m) γ · p
```

On the right-hand side of $\not{p}$ the spinor, $u(p, m)$ is used.

```
In[17]:= DiracSlash[p] . SpinorU[p, m]
Out[17]= γ · p  u(p, m)
```

This is $\overline{v}(p, m) \not{q} \not{p}$.

```
In[18]:= SpinorVBar[p, m] . DiracSlash[q, p]
Out[18]= v̄(p, m)  γ · q γ · p
```

Here we have $\gamma \cdot p\, v(p, m)$.

```
In[19]:= DiracSlash[p] . SpinorV[p, m]
Out[19]= γ · p  v(p, m)
```

Enter a spinor obeying the massless Dirac equation. The 0 is not displayed in the output.

```
In[20]:= SpinorU[p, 0]
Out[20]= u(p)
```

You may also omit the second argument for a massless spinor.

```
In[21]:= SpinorU[p]
Out[21]= u(p)
```

2.4 SU(N) Matrices and Structure Constants

FeynCalc provides functions for the input of SU(N) matrices and the corresponding structure constants. More specialized input functions in SU(2) and SU(3) (Gell-Mann and Pauli matrices) are provided by the subpackage PHI.

SUNT[*a*]	the a'th generator, T_a, of SU(N) in the fundamental representation

Input of SU(N) matrices.

For noncommutative multiplication of SU(N) matrices, use the Mathematica **Dot**, "**.**".

This is $T_a\, T_b\, T_c$.

```
In[22]:= SUNT[a] . SUNT[b] . SUNT[c]
Out[22]= T_a T_b T_c
```

The above can also be entered like this.

```
In[23]:= SUNT[a, b, c]
Out[23]= T_a T_b T_c
```

This is how you enter $T_a\, T_b\, T_c\, \delta_{cd}$.

```
In[24]:= SUNT[a, b, c] SUNDelta[c, d]
```

$$\text{Out[24]=}\ T_a\, T_b\, T_c\, \delta_{cd}$$

SUNDelta[i,j]	Kronecker delta function in SU(N)
SUNF[i,j,k]	antisymmetric structure constants of SU(N)
SUND[i,j,k]	symmetric structure constants of SU(N)

Input of SU(N) structure constants.

option name	*default value*	
Explicit	**True**	replace **SUNF** or **SUND** by traces

An option for the structure constants **SUNF** and **SUND**.

This is how you enter f_{abc}.

```
In[25]:= SUNF[a, b, c]
```

$$\text{Out[25]=}\ f_{abc}$$

This is how you enter d_{abc}.

```
In[26]:= SUND[a, b, c]
```

$$\text{Out[26]=}\ d_{abc}$$

They can be expressed in terms of the generating matrices (see section 4.4 for a discussion of the trace).

```
In[27]:= SUNF[a, b, c, Explicit → True]
```

$$\text{Out[27]=}\ 2\, i\, (\text{tr}(T_a\, T_c\, T_b) - \text{tr}(T_a\, T_b\, T_c))$$

Antisymmetry, $f_{acb} = -f_{abc}$.

```
In[28]:= SUNF[a, b, c, Explicit → True] +
         SUNF[a, c, b, Explicit → True] //
         Expand
```

$$\text{Out[28]=}\ 0$$

Symmetry, $d_{acb} = d_{abc}$.

```
In[29]:= SUND[a, b, c, Explicit → True] -
         SUND[a, c, b, Explicit → True] //
         Expand
Out[29]= 0
```

2.5 Denominators of Propagators

The denominators of propagators are entered in a special way. Each one-loop Feynman amplitude has factors of the form

$$df = \frac{1}{[q^2 - m_0^2]\,[(q+p_1)^2 - m_1^2]\,[(q+p_2)^2 - m_2^2]\;\ldots}$$

with q as loop momentum and the p_i denoting linear combinations of the external four-vectors.

`FeynAmpDenominator[ PropagatorDenominator[q, m0], PropagatorDenominator[q + p, m1], ...]`	$1/([q^2 - m_0^2][(q+p)^2 - m_1^2]) \ldots$
`FAD[ {q, m0}, {q-p,M}, ...]`	$1/([q^2 - m_0^2][(q+p)^2 - m_1^2]) \ldots$

Entering denominators of propagators.

This is $1/([q^2 - m_1^2][(q+p)^2 - m_2^2])$.

```
In[30]:= FeynAmpDenominator[ PropagatorDenominator[q,
         m1], PropagatorDenominator[q + p, m2]]
Out[30]= 1/((q^2 - m1^2)((q + p)^2 - m2^2))
```

This is $1/([q^2][(q-p)^2][k^2 - M^2])$.

```
In[31]:= FAD[q, q-p, k,M]
Out[31]= 1/(q^2 (q - p)^2 (k^2 - M^2))
```

2.6 Small Variables

For radiative corrections in the standard model, fermionic masses often can be neglected with respect to the gauge boson masses. Since, however, some of the Passarino-Veltman integrals may be infrared divergent, the fermionic (and photonic) small masses must remain as arguments of them and cannot be set to 0. In order to achieve this behavior, these small masses have to be entered with head **SmallVariable**. The idea is that every variable with head **SmallVariable** evaluates to 0, unless it is an argument of a scalar integral.

You can avoid to explicitly wrap the head **SmallVariable** around a mass by using the option **SmallVariables** of the function **OneLoop** (the function **OneLoop** is described in section 5.1).

SmallVariable[*m***]**	head of a small mass m, $m \ll M$

A head for small variables.

2.7 Quantum Fields

FeynCalc is meant to be a general framework for calculations in quantum field theory. Thus, FeynCalc defines notation not only for working with what comes out of a Feynman amplitude calculation, namely loop integrals, momenta, Dirac and SU(N) structures, propagators, etc., but also for the quantities needed to work out the input to such calculations, namely quantum fields and lagrangians.

QuantumField[*par1***, *par2*, ... , *ftype*, {*lorind*}, {*sunind*}]**	denotes a quantum field of type *ftype* with possible Lorentz indices *lorind* and SU(N) indices *sunind*. The optional first arguments *par1*, *par2*, $\cdots$, are partial derivatives acting on the field
PartialD[μ]	denotes a space-time derivative when given as first argument to **QuantumField**

Input of quantum fields.

A scalar field ψ.

```
In[32]:= QuantumField[ψ]
Out[32]= ψ
```

If the field is considered as part of an SU(N) multiplet and the SU(N) index is i, this is the notation.

```
In[33]:= QuantumField[ψ, {i}]
Out[33]= ψ_i
```

Some field g with an SU(N) index i and a Lorentz index ν.

```
In[34]:= QuantumField[g, {ν}, {i}]
Out[34]= g^i_μ
```

This is the notation for the space-time derivative $\partial/\partial x^\mu$ of the same field.

```
In[35]:= QuantumField[PartialD[μ],g, {ν}, {i}]
Out[35]= ∂_μ g^i_ν
```

2.8 Propagators

Several utility functions for input of propagators are available.

GluonPropagator[*p*, *μ*,*a*, *ν*,*b***]**	represents a gluon propagator $\Pi^{\mu\nu}_{ab}(p)$
GP[*p*, *1*, *2***]**	is a short form for **GluonPropagator[***p*, *li1*,*ci1*, *li2*,*ci2***]**
GhostPropagator[*p*, *μ*,*a*, *ν*,*b***]**	represents a ghost propagator
GHP[*p*, *1*,*2* **]**	is a short form for **GhostPropagator[***p*, *li1*,*ci1*, *li2*,*ci2***]**
QuarkPropagator[*p***]**	gives the massless quark propagator
QuarkPropagator[*p*,*m***]**	gives the quark propagator with mass m
QP[*p*,*m***]**	gives the quark propagator with mass m

Input functions for gluon ghost and quark propagators.

The options for **GluonPropagator** are:

option name	*default value*	
Gauge	**1**	gauge choice (1: Feynman)
OPE	**0**	whether to add the 2-gluon OPE Feynman rule
CounterTerm	**False**	whether to add counter terms
CouplingConstant	**Gstrong**	the coupling constant
Dimension	**D**	the space-time dimension

Options for **GluonPropagator**.

Enter a Gluon propagator.

```
In[36]:= GluonPropagator[p,μ,a, ν,b ]
```

$$\text{Out[36]}= -\frac{i\, g^{\mu\nu}\, \delta_{ab}}{p^2}$$

A convenient short way to enter a gluon propagator.

```
In[37]:= GP[q,1,2]
```

$$\text{Out[37]}= -\frac{i\, g^{i1i2}\, \delta_{c1c2}}{p^2}$$

Specifying the general covariant gauge.

```
In[38]:= GluonPropagator[p, {μ,a}, {ν,b},
         Gauge->ξ]
```

$$\text{Out[38]}= \frac{i\left(\frac{(1-\xi)\,(p^\mu)\,(p^\nu)}{p^2} - g^{\mu\nu}\right)\delta_{ab}}{p^2}$$

Specifying an axial gauge.

```
In[39]:= GluonPropagator[p, {μ,a}, {ν,b},
         Gauge->{n, α}]
```

$$\text{Out[39]}= \frac{i\left(-g^{\mu\nu} - \frac{(p^\mu)\,(p^\nu)\,n^2 - \alpha\,(n^\mu)\,(n^\nu)\,p^2}{n\cdot p^2} + \frac{(p^\mu)\,(n^\nu)+(n^\mu)\,(p^\nu)}{n\cdot p}\right)\delta_{ab}}{p^2}$$

A possible way of entering a gluon propagator and an OPE insertion.

```
In[40]:= GP[q,1,2, OPE->True]
```

$$\text{Out[40]}= -\frac{1}{2}\,\Omega\,((-1)^m+1)\,\delta_{\text{c1c2}}\,(\text{O}^{\text{G2}}_{\text{i1 i2}}(p))\left(\frac{1}{p^2}\right)^2 - \frac{i\, g^{\text{i1i2}}\,\delta_{\text{c1c2}}}{p^2}$$

The explicit form of the OPE-Operator can be obtained by applying the **Explicit** function.

Enter a quark propagator.

```
In[41]:= QuarkPropagator[p]
```

$$\text{Out[41]} = -\frac{i\,\gamma\cdot p}{p^2}$$

The options for **QuarkPropagator** are:

option name	*default value*	
CounterTerm	**False**	whether to add counter terms
CouplingConstant	**Gstrong**	the coupling constant
Dimension	**D**	the space-time dimension
Loop	**0**	the loop order of the propagator
OPE	**False**	whether to add the 2-gluon OPE Feynman rule
Polarization	**0**	for OPE insertions this option is taken over

Options for QuarkPropagator.

The 1-loop quark propagator.

```
In[42]:= QuarkPropagator[p, Loop -> 1]
```

$$\text{Out[42]} = -\frac{i\,C_F\,(2-\varepsilon)\,g_s^2\,S_n\,\gamma\cdot p\left(-\frac{p^2}{\mu^2}\right)^{\varepsilon/2}}{\varepsilon}$$

The 1-loop quark propagator.

```
In[43]:= QuarkPropagator[p, OPE -> True]
```

$$\text{Out[43]} = -\frac{i\,C_F\,(2-\varepsilon)\,g_s^2\,S_n\,\gamma\cdot p\left(-\frac{p^2}{\mu^2}\right)^{\varepsilon/2}}{\varepsilon}$$

2.9 Vertex functions

GluonVertex[*p*, *μ*,*a*, *q*, *ν*,*b*, *k*, *λ*,*c***]**	represents a gluon vertex $V_{abc}^{\mu\nu\lambda}(p, q, k)$
GV[*p*,*1*, *q*,*2*, *k*,*3***]**	is a short form for **GluonVertex[***p*, *i1*,*c1*, *q*,*i2*,*c2*, *k*,*i3*,*c3***]**

Input functions for gluon vertex functions.

One way how to enter a 3-gluon Vertex.

```
In[44]:= GluonVertex[p, μ,a, q, ν,b, k, λ,c]//Explicit
```

$$\text{Out[44]=}\ g_s\,(p^\lambda\,g^{\mu\nu} - q^\lambda\,g^{\mu\nu} - g^{\lambda\nu}\,k^\mu + g^{\lambda\nu}\,q^\mu + g^{\lambda\mu}\,k^\nu - g^{\lambda\mu}\,p^\nu)\,f_{abc}$$

One way how to enter a 4-gluon Vertex.

```
In[45]:= GluonVertex[p, μ,a, q, ν,b, k, λ,c, r, σ,d]//Explicit
```

$$\text{Out[45]=}\ -i\,g_s^2\,((g^{\lambda\sigma}\,g^{\mu\nu} - g^{\lambda\mu}\,g^{\nu\sigma})\,f_{ad\mathrm{u}6}\,f_{bc\mathrm{u}6} + (g^{\lambda\sigma}\,g^{\mu\nu} - g^{\lambda\nu}\,g^{\mu\sigma})\,f_{ac\mathrm{u}6}\,f_{bd\mathrm{u}6} + (g^{\lambda\mu}\,g^{\nu\sigma} - g^{\lambda\nu}\,g^{\mu\sigma})\,f_{ab\mathrm{u}6}\,f_{cd\mathrm{u}6})$$

2.10 Twist 2 Operators

$$\tfrac{1}{2}\,(1 + (-1)^m)\,g_s^2\,\left(f_{ac3d}\,f_{bc\mathrm{c}3}\,\left(\mathrm{O}^{\mathrm{G4}}_{\mathrm{la}\,\nu\,\mu\,\mathrm{si}}(k, q, p, s)\right) - f_{ac\mathrm{c}3}\,f_{bc3d}\,\left(\mathrm{O}^{\mathrm{G4}}_{\mu\,\mathrm{la}\,\nu\,\mathrm{si}}(p, k, q, s)\right) - f_{ab\mathrm{c}3}\,f_{cc3d}\,\left(\mathrm{O}^{\mathrm{G4}}_{\mu\,\nu\,\mathrm{la}\,\mathrm{si}}(p, q, k, s)\right)\right)$$

2.11 Propagators

Several utility functions for input of propagators are available.

GluonPropagator[*p*, *μ*,*a*, *ν*,*b***]**	represents a gluon propagator $\Pi^{\mu\nu}_{ab}(p)$
GP[*p*, *1*, *2***]**	is a short form for **GluonPropagator[***p*, *li1*,*ci1*, *li2*,*ci2***]**
GhostPropagator[*p*, *μ*,*a*, *ν*,*b***]**	represents a ghost propagator
GHP[*p*, *1*,*2* **]**	is a short form for **GhostPropagator[***p*, *li1*,*ci1*, *li2*,*ci2***]**
QuarkPropagator[*p***]**	gives the massless quark propagator
QuarkPropagator[*p*,*m***]**	gives the quark propagator with mass m
QP[*p*,*m***]**	gives the quark propagator with mass m

Input functions for gluon ghost and quark propagators.

The options for **GluonPropagator** are:

option name	*default value*	
Gauge	**1**	gauge choice (1: Feynman)
OPE	**0**	whether to add the 2-gluon OPE Feynman rule
CounterTerm	**False**	whether to add counter terms
CouplingConstant	**Gstrong**	the coupling constant
Dimension	**D**	the space-time dimension

Options for **GluonPropagator**.

Enter a Gluon propagator.

```
In[46]:= GluonPropagator[p,μ, a, ν,b ]
Out[46]= -(i g^(μν) δ_ab / p^2)
```

A convenient short way to enter a gluon propagator.

```
In[47]:= GP[q,1,2]
```

$$\text{Out[47]} = -\frac{i\, g^{i1i2}\, \delta_{c1c2}}{p^2}$$

A convenient short way to enter a gluon propagator and an OPE insertion.

```
In[48]:= GP[q,1,2, OPE->True]
```

$$\text{Out[48]} = -\frac{1}{2}\,\Omega\,((-1)^m+1)\,\delta_{c1c2}\,(\mathrm{O}^{\mathrm{G2}}_{i1\,i2}(p))\left(\frac{1}{p^2}\right)^2 - \frac{i\, g^{i1i2}\, \delta_{c1c2}}{p^2}$$

Enter a quark propagator.

```
In[49]:= QuarkPropagator[p]
```

$$\text{Out[49]} = -\frac{i\,\gamma\cdot p}{p^2}$$

The options for **QuarkPropagator** are:

option name	*default value*	
CounterTerm	**False**	whether to add counter terms
CouplingConstant	**Gstrong**	the coupling constant
Dimension	**D**	the space-time dimension
Loop	**0**	the loop order of the propagator
OPE	**False**	whether to add the 2-gluon OPE Feynman rule
Polarization	**0**	for OPE insertions this option is taken over

Options for QuarkPropagator.

2.12 Vertex functions

GluonVertex[*p*, μ,*a*, *q*, ν,*b*, *k*, λ,*c***]**	represents a gluon vertex $V_{abc}^{\mu\nu\lambda}(p, q, k)$
GV[*p*,*1*, *q*,*2*, *k*,*3***]**	is a short form for **GluonVertex[***p*, *i1*,*c1*, *q*,*i2*,*c2*, *k*,*i3*,*c3***]**

Input functions for gluon vertex functions.

2.13 Twist 2 Operators

3 Inside FeynCalc

This section deals with the internals of FeynCalc. Casual users may skip it on a first read. For advanced users, considering contributing to FeynCalc, it is recommended reading.

3.1 Startup Sequence, Modules, Add-ons and Extensibility

FeynCalc is loaded with the following command:

```
<<HighEnergyPhysics`FeynCalc`
```

which causes the program file "FeynCalc.m" to be loaded. This file contains definitions of the core objects, used by higher level, functions. Some of the more important of these objects are: **DiracGamma**, **FeynAmpDenominator**, **FourVector**, **LorentzIndex**, **Momentum**, **NonCommutative**, **Pair**, **PartialD**, **PropagatorDenominator**, **QuantumField**, **SUNIndex**, **ScalarProduct**, **Spinor**.

"FeynCalc.m" also scans the subdirectories of the directory "HighEnergyPhysics" for files with an extension ".m". Each such file, like "FeynRule.m", should contain the definition of a function (e.g. **FeynRule**).

Each definition, whether in one of these files or in "FeynCalc.m" is, however, not loaded into memory. Instead the function name is declared using **DeclarePackage**, so that when this function is used the first time, the definition is loaded into memory. This standard technique significantly reduces startup time and memory consumption. Subdirectories not to be scanned are defined in "FeynCalc.m". The idea is that each directory corresponds to a module, that is, a group of functions pertaining to some subject within quantum field theory. The core modules are:

- **fctables.** Databases of lagrangians, amplitudes and integrals. Important functions: **B0**, **C0**, **Lagrangian**, **Amplitude**, **Integrate2**.
- **fcloops.** Tools for calculating loop integrals. Important functions: **OneLoop**, **PaVeReduce**.
- **fctools.** Various tools for working with quantum fields and amplitudes. Important functions: **Contract**, **DiracReduce**, **FunctionalD**, **FeynRule**, **Tr**, **FermionSpinSum**.
- **general.** Mathematical tools of a general nature that are used by one or more functions of the other subpackages, but may be useful in other contexts as well.
- **qcd.** Tools for working with QCD.

The governing idea of the file layout of FeynCalc is modularity. The files belonging to each module are in a separate directory; each file corresponds to one function. If a function uses some support functions or options that are not used by other functions but desirable to have globally accessible, one may put these in the same context as the function and add them to the list **multifunpack** in "FeynCalc.m". Dependence on functions in other contexts should be clearly stated with **MakeContext** statements at the beginning of each file.

The modules listed above all adhere to these conventions.

As will be noticed there are some files and directories not mentioned so far. Apart from the directory "Documentation", which is a directory for standard Mathematica package documentation, they are listed below under the packages to which they belong. These packages are not loaded by default, but can be enabled by setting certain configuration variables (see section 3.2). The reasons for this are: The packages are not considered essential for all users and also don't follow the loading conventions described above and take some time in loading (still of the order of seconds though).

- **FeynArts** is made up of the file "FeynArts.m" , the directories "GraphInfo", "Models" and some more files on the top level of "HighEnergyPhysics". It is an independent package by Sepp Kueblbeck, Hagen Eck and Thomas Hahn for generating Feynman graphs and corresponding amplitudes with Feynman rules as input. Integration within FeynCalc is not intended, but can, with a small effort, be achieved (see PHI below). For more information on FeynArts, see [3].
- **PHI** is a package by Frederik Orellana providing utilities for working with effective field theories and interacting with FeynArts. All files belonging to PHI are contained in the directory "Phi". It was not conceived as a FeynCalc subpackage and the file layout does not adhere to the FeynCalc conventions. Never the less, it has been modified to effectively act as a FeynCalc subpackage and is now distributed only with FeynCalc.
- **TARCER** is a package by Rolf Mertig for working with 2-loop propagator type integrals. Its file layout also does not adhere to the FeynCalc conventions, but it also is integrated within FeynCalc. The files belonging to TARCER are contained in the directory "Tarcer". For more information on TARCER, see [8].
- **fcdevel** is code which is not considered production ready and is meant to be used by developers only.

3.2 Configuration and Runtime Variables

As we have seen, the behaviour of most functions is controlled by options like e.g. **`Dimension`**. Additionally the behaviour of some functions depends on the setting of certain environment variables. A few of these should be set before loading FeynCalc, namely **`$LoadTARCER, $LoadPhi, $LoadFeynArts`**.

variable name	*default value*	
`$LoadTARCER`	**`False`**	load TARCER
`$LoadPhi`	**`False`**	load PHI
`$LoadFeynArts`	**`False`**	load FeynArts

Variables switching loading of extra packages on and off.

Others can be set at runtime; e.g. **$Color, $Covariant, $Gauge, $NonComm, $Abbreviations**. **$BreitMaison** and **$Larin** are special in that they are set at runtime, but, once set cannot be changed.

variable name	*default value*	
$Color	**False**	color special variables
$Covariant	**True**	display Lorentz indices as upper or lower indices
$Gauge	**1**	the gauge fixing parameter of QED in Lorentz gauge. 1 corresponds to Feynman gauge. Notice that **$Gauge** is used by some functions, the option **Gauge** by others
$NonComm	**{DiracGamma, DiracGammaT, ..., OPESum}**	a list of all noncommutative heads
$Abbreviations	**{"^" → "", "*" → "", ..., "Vector" → "V"}**	a list of string substitution rules used when generating names for storing intermediate results. Used by **OneLoop** and **PaVeReduce**
$BreitMaison	**False**	use the Breitenlohner-Maison γ^5 scheme (currently not supported)
$Larin	**False**	use the Larin-Gorishny-Atkyampo-DelBurgo γ^5 scheme
$VeryVerbose	**0**	display intermediate messages from FeynCalc
$MemoryAvailable	**8**	the amount of available main memory in megabytes
$SpinorMinimal	**False**	a global switch for an additional simplification attempt in **DiracSimplify** for more than one Spinor line

Runtime variables.

The variable **$VeryVerbose** may be set to 0, 1, 2 or 3. The higher the setting the more intermediate information is printed during the calculations.

FeynCalc is designed in such a way that certain intermediate results can be stored in order to speed up the computations. This of course may become too memory consuming. The storing stops, when the value of **10^(-6)*MemoryInUse[ ]** gets bigger than **$MemoryAvailable-1**.

The default of FeynCalc is to use the naive γ^5 prescription, i.e., γ^5 anticommutes with D-dimensional Dirac

matrices. Setting **$BreitMaison** to **True** changes the commutation behaviour[2] of γ^5: It anti-commutes with Dirac matrices in four dimensions, but commutes with the $(D-4)$-dimensional part. The basic features of the Breitenlohner-Maison scheme are implemented in FeynCalc, but they have not been tested very thoroughly (some simple calculations are given in section 4.2). Therefore one should check the results for correctness.

3.3 Data Types

In quantum field theory, a number of types of variables are needed. E.g. variables representing Lorentz indices, momenta, SU(N) indices and quantum fields. As we have seen in previous sections, the way to tell FeynCalc that a variable is of one of these types is to wrap some head around it. However, FeynCalc provides also a method for having variables be of a certain type without wrapping a head around them. The knowledge about such variables is carried by the function **DataType**. The default setting is

```
DataType[__, _] := False
```

For example, to assign the data-type **PositiveInteger** to the variable **x**, do

```
DataType[x, PositiveInteger] = True
```

DataType[*x, type***] = True**	defines the symbol *x* to have data-type *type*
DataType[*x1, x2, ..., type***]**	defines the symbols *x1*, *x2*, ... to have data-type *type*

FeynCalc data-types.

[2] In fact, how FeynCalc treats γ^5 depends on the setting of the variables **$BreitMaison**, **$Larin** and **$West**. They are all boolean variables; the first two specify the γ^5 scheme. None of them have been neither thoroughly implemented nor tested. If **$West** is set to True (which is the default), traces involving more than 4 Dirac matrices and a γ^5 are calculated recursively according to formula (A.5) from [14], which is based on the Breitenlohner Maison -scheme.

NonCommutative	a variable treated as noncommutative by **Dot**, "."
PositiveInteger	a positive integer variable
NegativeInteger	a negative integer variable
PositiveNumber	a positive number variable
FreeIndex	a Lorentz index not to be contracted by **Contract**
GrassmannParity	**DataType[f, GrassmannParity] = 1** declares a field **f** to be bosonic, **DataType[f, GrassmannParity] = -1** declares it to be fermionic.

FeynCalc data-types.

Declare some symbols **f** and **g** to be noncommutative.

```
In[50]:= DataType[f, g, NonCommutative] = True;
```

An algebraic expression using **Dot**, ".".

```
In[51]:= t = f.g - g.(2a).f
```

```
Out[51]= f.g - g.(2a).f
```

DotSimplify only extracts **a** out of the noncommutative product.

```
In[52]:= DotSimplify[t]
```

```
Out[52]= f.g - 2a g.f
```

Clean up.

```
In[53]:= DataType[f, g, NonCommutative] = True;
```

Declare some symbols **m** and **a** to be of some data-types **even** and **odd**.

```
In[54]:= DataType[m, odd] = DataType[a, even] =
         True;
```

Define functions for reducing powers of -1.

```
In[55]:= ptest1[x_] := x /.  (-1)^n_ :> -1 /;
         DataType[n, odd] ;
         ptest2[x_] := x /.  (-1)^n_ :> 1 /;
         DataType[n, even];
```

Redefine the expression **t**.

```
In[56]:= t = (-1)^m + (-1)^a + (-1)^z
```

```
Out[56]= (-1)^a + (-1)^m + (-1)^z
```

Apply the functions to the expression **t**.

```
In[57]:= ptest1[t]

Out[57]= -1 + (-1)^a + (-1)^z

In[58]:= ptest2[%]

Out[58]= (-1)^z
```

Clear **t**.

```
In[59]:= Clear[t];
```

DeclareNonCommutative[*x*, *y*, ...]	sets **DataType[***x*, **NonCommutative] = True**; **DataType[***y*, **NonCommutative] = True** ; ...

Declaration of noncommutative variables.

3.4 Output Forms and Internal Representation

By default the internal representation is hidden. For some purposes, like debugging or hacking the FeynCalc source, however, it may be useful to know how the various objects are represented within FeynCalc.

Some input functions are transformed right away to the internal representation, others only on being acted upon by utility functions. An example of the first kind is **DiracMatrix[**μ**]**. An example of the second kind is **FourVector[p,** μ**]**; during the internal evaluation of e.g. **Contract[FourVector[p,** μ**], FourVector[p,** μ**]]**, however, a conversion to the internal representation is done. To see the internal representation of an object, the function **FeynCalcInternal** can always be used.

The basic idea is to wrap a special head around each special input variable. For example, the arguments of **MetricTensor[mu, nu]** each get a head **LorentzIndex**. These heads have only a few functional definitions. A **D**-dimensional **LorentzIndex[mu, D]**, for example, simplifies to **LorentzIndex[mu]** when **D** is replaced by 4.

Inside FeynCalc the functions and rules are specified in such a way that they only apply to those objects with the right head. This programming style, which makes strong use of the pattern matching capabilities of Mathematica, especially the so-called upvalue function definitions, has proven to be useful and reliable for the purpose of algebraic calculations in physics.

Pair[*a***, *b***]**	is a special pairing used in the internal representation: *a* and *b* may have heads **LorentzIndex** or **Momentum**. If both *a* and *b* have head **LorentzIndex**, the metric tensor is understood. If *a* and *b* have head **Momentum**, a scalar product is meant. If one of *a* and *b* has head **LorentzIndex** and the other **Momentum**, a Lorentz vector (e.g. p_μ) is understood
Momentum[*p***]**	is a four-dimensional momentum. For other than four dimensions ($d \neq 4$) use **Momentum[***p***, *d***]**. **Momentum[***p***, 4]** simplifies to **Momentum[***p***]**
LorentzIndex[μ]	is a four-dimensional Lorentz index. When μ is an integer, it evaluates to mbExplicitLorentzIndex[μ]. For other than four dimensions use **LorentzIndex[***p***, *d***]**. **LorentzIndex[***p***, 4]** simplifies to **LorentzIndex[***p***]**
ExplicitLorentzIndex[μ]	is an explicit Lorentz index, i.e., μ is an integer

Internal representation of Lorentz input functions.

Now we shall study the internal representation through a few examples. **FeynCalcInternal** shall be applied when necessary.

Suppress FeynCalc typesetting by going from **TraditionalForm** output to **StandardForm** output. This can also be achieved via the menu setting of 'Cell' → 'Default Output Format Type'.

```
In[60]:= SetOptions[$FrontEnd,
         "CommonDefaultFormatTypes" → "Output"
         → StandardForm];
```

This is the internal representation of a γ^μ in four dimensions.

```
In[61]:= DiracMatrix[μ]

Out[61]= DiracGamma[LorentzIndex[μ]]
```

Here is a γ^μ in D dimensions.

```
In[62]:= DiracMatrix[μ, Dimension→D]

Out[62]= DiracGamma[LorentzIndex[μ, D], D]
```

This is a product of a four-dimensional $\not{q}$ and a D-dimensional $\not{q}$.

```
In[63]:= DiracSlash[q] . DiracSlash[q, Dimension
         → D]

Out[63]= DiracGamma[Momentum[q]] . DiracGamma[Momentum[q,
         D], D]
```

This shows the convention for a four-dimensional scalar product.

```
In[64]:= DiracSimplify[%]

Out[64]= Pair[Momentum[q], Momentum[q]]
```

The head **Pair** is also used for metric tensors; in this case in D dimensions. The dimension is provided as an argument.

```
In[65]:= MetricTensor[μ, ν, Dimension → D]

Out[65]= Pair[LorentzIndex[μ, D], LorentzIndex[ν, D]]
```

Changing D to 4 recovers the default for a four-dimensional $g^{\mu\nu}$.

```
In[66]:= % /. D → 4

Out[66]= Pair[LorentzIndex[μ], LorentzIndex[ν]]
```

A four-vector does not immediately evaluate to the internal representation.

```
In[67]:= FourVector[p, μ]

Out[67]= FourVector[p, μ]
```

By applying **FeynCalcInternal** we force the internal representation to be used: A four-vector is also represented as a **Pair** internally.

```
In[68]:= FourVector[p, μ] // FeynCalcInternal

Out[68]= Pair[LorentzIndex[μ], Momentum[p]]
```

By applying some utility function, transformation to the internal representation is automatically done.

```
In[69]:= FourVector[p, μ] FourVector[p, μ] //
         Contract

Out[69]= Pair[Momentum[p], Momentum[p]]
```

A $(D-4)$-dimensional four-vector.

```
In[70]:= FourVector[p, μ, Dimension → D - 4] //
         FeynCalcInternal

Out[70]= Pair[LorentzIndex[μ, -4 + D], Momentum[p, -4 + D]]
```

It vanishes after changing the dimension to 4.

```
In[71]:= % /. D → 4

Out[71]= 0
```

Polarization vectors are a special kind of 4-vectors. This shows their internal representation.

```
In[72]:= PolarizationVector[k, μ]

Out[72]= Pair[LorentzIndex[μ], Momentum[Polarization[k]]]
```

DiracGamma[*x***]**	head of a four-dimensional γ^μ or $\not{p}$
DiracGamma[*x***, *D*]**	head of a D-dimensional γ^μ or $\not{p}$

Internal representation of Dirac matrices.

Polarization[*k***]**	A polarization vector $\varepsilon(k)$ is understood when the argument of **Momentum** has head **Polarization**

Internal representation of Polarization vectors.

Internally in FeynCalc, all spinors are represented with the same function, **Spinor**. Which of the Dirac spinors u, v and the conjugate spinors $\overline{u}, \overline{v}$ are understood, depends on the position of the spinors in the Dirac chain and the sign of the momentum argument.

Spinor[*p***, *m*]**	$u(p, m)$ or $\overline{u}(p, m)$
Spinor[−*p***, *m*]**	$v(p, m)$ or $\overline{v}(p, m)$

Internal representation of spinors.

The internal representation of $v(-p, m)$ and $u(p, m)$ is the same.

```
In[73]:= SpinorV[-p, m] - SpinorU[p, m] //
         FeynCalcInternal

Out[73]= 0
```

Analogously to Lorentz indices, SU(N) indices are also wrapped with a head in the internal representation.

SUNIndex[*i***]**	is an SU(N) index. When i is an integer, it evaluates to ExplicitSUNIndex[i]
ExplicitSUNIndex[*i***]**	is an explicit SU(N) index, i.e., i is an integer

Internal representation of SU(N) input functions.

After having completed a calculation with FeynCalc and obtained some screen output, a question will often arise: How does one export the result to a file or to some other application. The screen output in **TraditionalForm** is not very useful; copying and pasting it to a text editor will produce some garbled string with all the typesetting codes of Mathematica included. Saving an expression with **Put** or **Save** will produce the same as copying the screen output in **InputForm**, namely the internal representation of FeynCalc, which is also not very useful, because it contains all the heads, options, etc. FeynCalc needs to make sense of the expression. For these reasons the FeynCalc external output format exists. An expression in the internal representation is translated to the FeynCalc external output format with the command **FeynCalcExternal**. The inverse translation is done with **FeynCalcInternal**.

Despite the length of the produced output, it can still be useful to see expressions in **InputForm**. To permanently enable this, one can simply evaluate **FI**. What FeynCalc does is setting **$PrePrint** to **InputForm**. To switch back to typeset output, one can simply evaluate **FC**. Here, FeynCalc clears **$PrePrint**[3].

FeynCalcExternal[*exp***]**	translates *exp* from the internal FeynCalc representation to the simpler external one (i.e., **FV**, **GA**, **GS**, etc.)
FeynCalcInternal[*exp***]**	translates *exp* into the internal FeynCalc representation

Translation to and from external output format.

[3] With the terminal interface, **$PrePrint** gets set to **FeynCalcForm**.

FCE	just an abbreviation of **FeynCalcExternal**
FCI	just an abbreviation of **FeynCalcInternal**

Abbreviation of **FeynCalcExternal** and **FeynCalcInternal**.

FI	disables typesetting of output
FC	enables typesetting of output

Switch typesetting off and on.

option name	*default value*	
FinalSubstitutions	**{}**	additional translation rules

Option of **FeynCalcExternal** and **FeynCalcInternal**.

The external output form of γ^μ.

```
In[74]:= FeynCalcExternal[DiracMatrix[μ],
         FinalSubstitutions → {μ → mu}]

Out[74]= GA[mu]
```

Translation back to the internal representation.

```
In[75]:= FeynCalcInternal[%, FinalSubstitutions
         → {mu → σ}]

Out[75]= DiracGamma[LorentzIndex[σ]]
```

Switch back to FeynCalc typesetting by going from **StandardForm** output to **TraditionalForm** output. This can also be achieved via the menu setting of 'Cell' → 'Default Output Format Type'.

```
In[76]:= SetOptions[$FrontEnd,
         "CommonDefaultFormatTypes"→"Output" →
         TraditionalForm]
```

3.5 Help System

The help system follows standard Mathematica conventions: Description of an object **obj** is obtained by evaluating **?obj**.

E.g. the description of the function **FeynRule** is obtained by evaluating **?FeynRule**.

```
In[77]:= ?FeynRule
Out[77]= FeynRule[lag, fields] gives the Feynman rule corresponding
         to the field configuration fields of the lagrangian lag.
```

More in-depth information about e.g. **FeynRule** can be obtained by typing in FeynRule in the Add-ons part of the help browser (see figure 1.1).

The files configuring what appears in the help browser are: "FeynCalcBook.nb" and "BrowserCategories.m" in the directory "HighEnergyPhysics/Documentation/English". The first file is a notebook containing the actual content organized in cells, all of which carry a cell tag. The second file defines the organization of the contents in terms of the cell tags. After a new install of FeynCalc or a change to one of these files 'Rebuild Help Index' from the 'Help' menu must be selected. Then, after clicking the navigation button 'Add-ons' in the help browser window, in the left pane 'High Energy Physics' will appear. Clicking this and the resulting fields in the other three panes will cause FeynCalc help information to appear. The second pane has various introductory material and the field 'Functions'. Clicking this field causes the appearance in the third pane of the fields 'Core', 'Tools', 'Loops', 'Tables', 'QCD' and 'General' corresponding to the file organization of FeynCalc as described in section 3.1. Clicking each of these will display a list of the corresponding objects in the fourth pane. Each of these lists is organized in groups of objects, with the groups in the following order: Functions, abbreviations, constants, options, internal functions. The internal functions are described only for completeness; they are of no interest for other than developers.

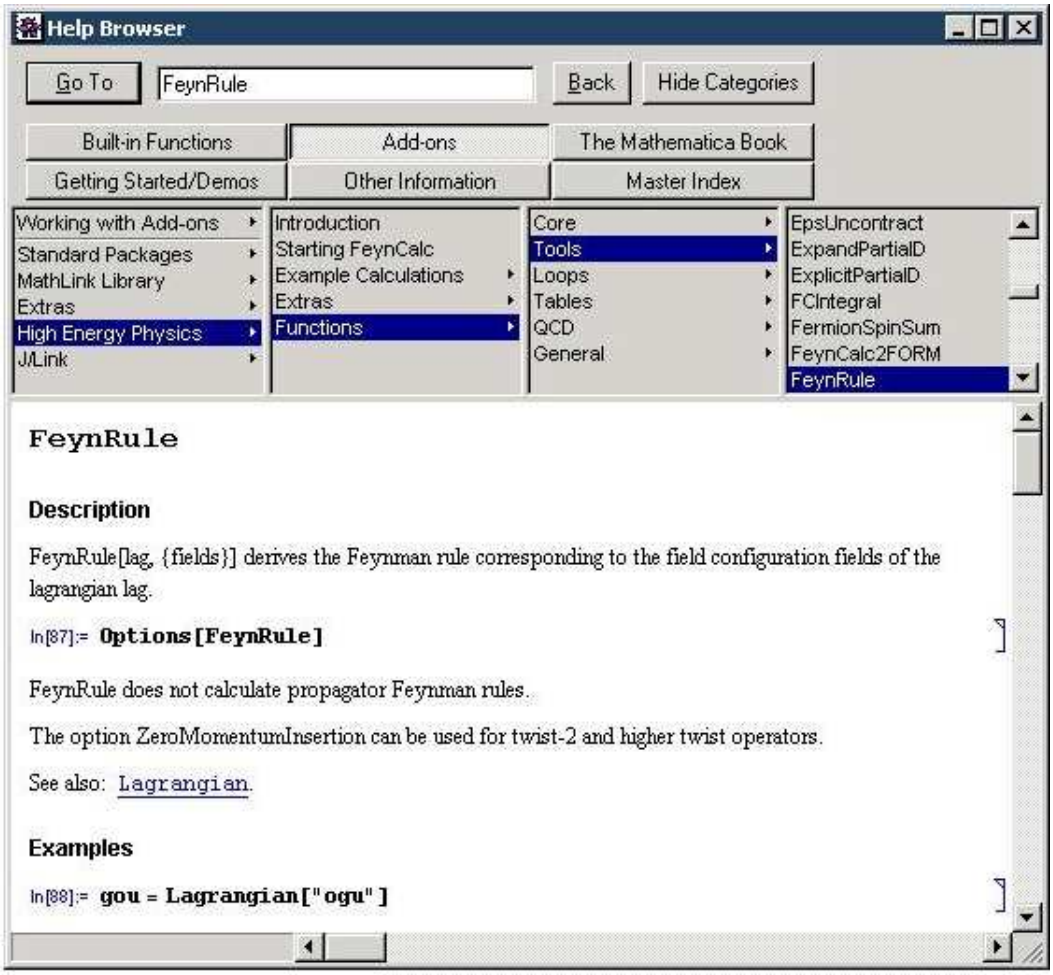


Figure 1.1: The help system of FeynCalc.

4 Elementary Calculations

You can use FeynCalc for basic calculations like Lorentz algebra and Dirac and color trace evaluations. This section contains simple examples of such calculations.

4.1 Lorentz Algebra

The **Contract** function contracts double Lorentz indices, if at least one belongs to a metric tensor, a four-vector or a Levi-Civita tensor.

Contract[*expr***]** contract double Lorentz indices in *expr*

The function for contraction of tensors.

In four dimensions $g^{\mu}_{\mu} = 4$.

```
In[78]:= Contract[MetricTensor[μ, μ]]

Out[78]= 4
```

While in D dimensions $g^{\mu}_{\mu} = D$.

```
In[79]:= Contract[MetricTensor[μ, μ, Dimension →
         D]]

Out[79]= D
```

Contract $g^{\alpha\beta} p^{\beta}$.

```
In[80]:= Contract[MetricTensor[α, β]
         FourVector[p, β]]

Out[80]= p^α
```

Contract $q^{\alpha} (p - q)^{\alpha}$.

```
In[81]:= Contract[FourVector[q, α] FourVector[p -
         q, α]]

Out[81]= (p - q)·q
```

Numerical factors are pulled out when calculating with **FourVector**s.

```
In[82]:= FourVector[2 p, μ] FourVector[2 p, μ] //
         Contract

Out[82]= 4 p^2
```

This contracts $g^{\alpha\beta}\,\gamma^{\alpha}$.

```
In[83]:= Contract[MetricTensor[α, β]
         DiracMatrix[α]]

Out[83]= γ^β
```

Contracting $q^{\alpha}\,\gamma^{\alpha}$ yields a Feynman slash.

```
In[84]:= Contract[FourVector[q, α]
         DiracMatrix[α]]

Out[84]= γ·q
```

Contracting $\varepsilon^{\mu\nu\rho\sigma}\,p^{\sigma}$ gives $\varepsilon^{\mu\nu\rho p}$.

```
In[85]:= Contract[LeviCivita[μ, ν, ρ, σ]
         FourVector[p, σ]]

Out[85]= ε^μνρp
```

The contraction of $\varepsilon^{\alpha\nu\rho\sigma}\,\varepsilon^{\beta\nu\rho\sigma}$ yields $-6\,g^{\alpha\beta}$.

```
In[86]:= Contract[LeviCivita[α, ν, ρ, σ]
         LeviCivita[β, ν, ρ, σ], ]

Out[86]= -6 g^αβ
```

option name	*default value*	
EpsContract	**False**	contract Levi-Civita **Eps**
Expanding	**True**	expand the input
Factoring	**False**	factor canonically

Options for **Contract**.

Contracting only $g^{\alpha\sigma}\,p^{\alpha}\,p^{\sigma}$ in $g^{\alpha\sigma}\,p^{\alpha}\,p^{\sigma}\,(q^{\beta}+r^{\beta})\,(p^{\beta}-s^{\beta})$.

```
In[87]:= Contract[MetricTensor[α, σ] *
         FourVector[p, α] FourVector[p, σ] *
         (FourVector[q, β] + FourVector[r, β]) *
         (FourVector[p, β] - FourVector[q, β]),
         Expanding → False]

Out[87]= (p^β - q^β) (q^β + r^β) p^2
```

FeynCalc uses the transversality condition $(k^\mu \cdot \varepsilon^\mu(k)) = 0$ for polarization vectors.

```
In[88]:= Contract[FourVector[k, μ]
         PolarizationVector[k, μ]]

Out[88]= 0
```

ExpandScalarProduct[*expr***]**	expand scalar products and four-vectors in *expr*
MomentumExpand[*expr***]**	expand **Momentum[a+b+ ...]** in *expr* into **Momentum[a] + Momentum[b] + ...**
MomentumCombine[*expr***]**	invert the operation of **MomentumExpand** and **ExpandScalarProduct**

Functions for expansion and combination of scalar products and four-vectors.

As an example, expand $(a + b) \cdot (c - 2d)$.

```
In[89]:= ExpandScalarProduct[ScalarProduct[a + b,
         c - 2 d]]

Out[89]= a·c - 2 a·d + b·c - 2 b·d
```

Combine again.

```
In[90]:= MomentumCombine[%]
Out[90]= (a + b)·(c - 2d)
```

Consider again $q^\alpha (p - q)^\alpha$. **Contract** expands scalar products.

```
In[91]:= Contract[FourVector[q, α] FourVector[p -
         q, α]]

Out[91]= (p - q)·q
```

This is how you can substitute $q^2 \to 0$ afterwards.

```
In[92]:= % /. ScalarProduct[q, q] → 0

Out[92]= p·q
```

Instead of substituting scalar products at the end of the calculation another possibility is to assign special values for scalar products first. These special values are inserted immediately whenever possible during the calculation.

Set $q^2 = 0$ before a calculation.

```
In[93]:= ScalarProduct[q, q] = 0;
```

Contracting (and expanding) $q^\alpha (p-q)^\alpha$ now yields $(p \cdot q)$.

```
In[94]:= Contract[FourVector[q, α] FourVector[p -
           q, α]]

Out[94]= p·q
```

Clear the value of $(p \cdot q)$.

```
In[95]:= DownValues[Pair] = Select[DownValues[Pair],
           FreeQ[#, q]&];
```

This expands $(a+b)^\mu$ to $a^\mu + b^\mu$.

```
In[96]:= ExpandScalarProduct[FourVector[a + b,
           μ]]

Out[96]= a^μ + b^μ
```

4.2 Dirac Algebra

For the manipulation of noncommutative products of Dirac matrices and spinors, a number of functions are provided (see also section 6.1). **DiracEquation** applies the Dirac equation without expanding. **DiracOrder** orders products of Dirac matrices in a canonical way: Basically it is just the implementation of the anticommutator relation $\{\gamma^\mu, \gamma^\nu\} = 2\, g^{\mu\nu}$. **Chisholm** substitutes products of three Dirac matrices or slashes in *expr* using the Chisholm identity.

DiracEquation[*expr***]**	apply the Dirac equation
Chisholm[*expr***]**	apply the Chisholm identity
DiracOrder[*expr***]**	alphabetically order Dirac matrices
DiracOrder[*expr***, {***a***, *b*, ...}]**	order according to *a*, *b*, ...

Manipulation functions for Dirac matrices and spinors.

All functions take as *expr* any expression with "." as the noncommutative multiplication operator between Dirac matrices and/or Dirac slashes.

The Dirac equation.

```
In[97]:= (# == DiracEquation[#])&[DiracSlash[p] .
           Spinor[p, m]]

Out[97]= (γ·p).φ(p, m) == m φ(p, m)
```

The Chisholm identity.

```
In[98]:= (# == Chisholm[#])&[DiracMatrix[μ, ν, ρ]]
```

$$\text{Out[98]}= \gamma^\mu \gamma^\nu \gamma^\rho == i\,\gamma^{\$MU\$26}\cdot\gamma^5\,\epsilon^{\mu\mu\rho\$MU\$26} + \gamma^\rho\, g^{\mu\nu} - \gamma^\nu\, g^{\mu\rho} + \gamma^\mu\, g^{\nu\rho}$$

Order the product $\gamma^\beta\,\gamma^\alpha$.

```
In[99]:= DiracOrder[DiracMatrix[β, α]]
```

$$\text{Out[99]}= 2\,g^{\alpha\beta} - \gamma^\alpha\,\gamma^\beta$$

Anticommute back to $\gamma^\beta\,\gamma^\alpha$.

```
In[100]:= DiracOrder[%, {β, α}]
```

$$\text{Out[100]}= \gamma^\beta\,\gamma^\alpha$$

Simplifications like $\gamma^\mu\,\gamma^\mu\,\not{p}\not{p} = 4\,p^2$ are built in.

```
In[101]:= DiracOrder[DiracMatrix[μ, μ],
            DiracSlash[p, p]]
```

$$\text{Out[101]}= 4\,p^2$$

$\gamma^\alpha\,\gamma^\mu\,\gamma^\alpha = (2-D)\,\gamma^\mu$ in D dimensions.

```
In[102]:= DiracOrder[DiracMatrix[α, μ, α,
            Dimension → D]]
```

$$\text{Out[102]}= 2\,\gamma^\mu - D\,\gamma^\mu$$

$-\not{p}\not{q}\not{p} = \not{q}\,p^2 - 2\,\not{p}\,(p\cdot q)$.

```
In[103]:= DiracOrder[DiracSlash[-p, q, p]]
```

$$\text{Out[103]}= \gamma\cdot q\;\; p^2 - 2\,\gamma\cdot p\;\; p\cdot q$$

DotSimplify expands and reorders noncommutative terms using relations specified by the option **DotSimplifyRelations** or by **Commutator** and/or **AntiCommutator** definitions. Whether noncommutative expansion is done depends on the option **Expanding**. Notice that in the rules of the setting of **DotSimplifyRelations**, **Condition** should not be used and patterns should be avoided on the right-hand sides. Also, the performance of **DotSimplify** scales inversely and very badly with the complexity of **DotSimplifyRelations** and the number of terms of the expression to be simplified.

DotSimplify[*expr***]**	expand and reorder noncommutative terms

Simplification of noncommutative products.

option name	*default value*	
Expanding	**True**	noncommutatively expand the result
DotSimplifyRelations	**{}**	a list of substitution rules of the form **DotSimplifyRelations → {a . b → c, b^2 → 0, ...}**
DotPower	**False**	whether noncommutative powers are represented by successive multiplication or by **Power**.

Options for **DotSimplify**.

This is a four-dimensional product: $2 b \not{a} 2 (\not{d} - \not{c})(6 \not{q} - 3 \not{p})$. **DotSimplify** pulls common numerical factors out.

```
In[104]:= DiracSlash[2 b, a, 2 (d - c), (6 q - 3
          p)] // DotSimplify
```

$$\text{Out[104]=}\ -12\,\gamma\cdot b\,\gamma\cdot a\,\gamma\cdot(d-c)\,\gamma\cdot(p-2\,q)$$

Here is another product. Notice that we are using the shorthand **FeynCalcExternal** form (see section 3) for input.

```
In[105]:= GA[μ].(a GS[p] - b GS[q]).GS[q].GA[ν]
```

$$\text{Out[105]=}\ \gamma^{\mu}.(a\ \gamma\cdot p - b\ \gamma\cdot q).(\gamma\cdot q).\gamma^{\nu}$$

With **Expanding → True** sums are distributed over. This also causes symbols not known as being noncommutative (see section 3.3) to be pulled out.

```
In[106]:= DotSimplify[%, Expanding → True]
```

$$\text{Out[106]=}\ a\,\gamma^{\mu}.(\gamma\cdot p)(\gamma\cdot q).\gamma^{\nu} - b\,\gamma^{\mu}(\gamma\cdot q).(\gamma\cdot q).\gamma^{\nu}$$

With **DotPower → True** dot products of identical objects are replaced with powers.

```
In[107]:= DotSimplify[%, DotPower → True]
```

$$\text{Out[107]=}\ a\,\gamma^{\mu}.(\gamma\cdot p)(\gamma\cdot q).\gamma^{\nu} - b\,\gamma^{\mu}(\gamma\cdot q)^2.\gamma^{\nu}$$

One may specify relations to be included in the simplification process.

```
In[108]:= DotSimplify[%, DotPower → True,
          DotSimplifyRelations → {GS[q]^2 → 1}]
```

$$\text{Out[108]=}\ a\,\gamma^{\mu}.(\gamma\cdot p)(\gamma\cdot q).\gamma^{\nu} - b\,\gamma^{\mu}.\gamma^{\nu}$$

Declare some symbols as noncommuting and define a commutator.

```
In[109]:= DeclareNonCommutative[a, b, c];
          Commutator[a, c] = 1;
```

DotSimplify uses this information.

```
In[110]:= DotSimplify[a . (b - z c) . a]

Out[110]= a.b.a - z(a + c.a.a)
```

Clear definitions.

```
In[111]:= UnDeclareNonCommutative[a, b, c];
          Commutator[a, c] = 0;
```

DiracSimplify[*expr***]**	contract all Lorentz indices and simplify
DiracSimplify2[*expr***]**	like **DiracSimplify** but leaves any γ^5 untouched. γ^6 and γ^7 are replaced with their definitions
DiracReduce[*expr***]**	reduce Dirac matrices to the standard basis (S, P, V, A, T) using the Chisholm identity
DiracTrick[*expr***]**	contracts gamma matrices with each other and performs several simplifications, but no expansion

Simplification functions for Dirac matrices and spinors.

All functions take as *expr* any expression with "." as the noncommutative multiplication operator between Dirac matrices and/or Dirac slashes.

DiracBasis	a head wrapped around Dirac structures (and 1) by **DiracReduce**)

A head used by **DiracReduce**.

DiracSimplify contracts Dirac matrices with equal indices, moves γ^5, γ^6 and γ^7 to the right, applies the Dirac equation and expands noncommutative products (see section 3.2). The Dirac matrices in the result of **DiracSimplify** are only ordered in a canonical way if they are between spinors. See below and section 3.2 for the treatment of γ^5 in D dimensions.

This is $\gamma^\mu \gamma^\mu = D$.

```
In[112]:= DiracSimplify[DiracMatrix[μ, μ,
          Dimension → D]]

Out[112]= D
```

Here the Kahane algorithm is used.
$\gamma^\mu \gamma^\nu \gamma^\rho \gamma^\sigma \gamma^\mu = -2\,\gamma^\sigma \gamma^\rho \gamma^\nu$.

```
In[113]:= DiracSimplify[DiracMatrix[μ, ν, ρ, σ,
          μ]]

Out[113]= -2 γ^σ . γ^ρ . γ^ν
```

Kahane also gives this identity:
$\frac{1}{2}\gamma^\mu \gamma^\alpha \gamma^\beta \gamma^\gamma \gamma^\delta \gamma^\mu =$
$\gamma^\gamma \gamma^\beta \gamma^\alpha \gamma^\delta + \gamma^\delta \gamma^\alpha \gamma^\beta \gamma^\gamma$.

```
In[114]:= DiracSimplify[1/2 DiracMatrix[μ, α, β,
          γ, δ, μ]]

Out[114]= γ^γ . γ^β . γ^α . γ^δ + γ^δ . γ^α . γ^β . γ^γ
```

This is
$\not{p}(m-\not{q})\not{p} = \not{q}p^2 + p^2 m - 2\not{p}(p\cdot q)$.

```
In[115]:= DiracSimplify[DiracSlash[p],
          DiracSlash[-q] + m, DiracSlash[p]]

Out[115]= γ·q  p^2 + m p^2 - 2 γ·p  p·q
```

This is $\gamma^5 \gamma^\mu = -\gamma^\mu \gamma^5$.

```
In[116]:= DiracSimplify[DiracMatrix[5],
          DiracMatrix[μ]]

Out[116]= -γ^μ γ^5
```

$\gamma^6 \gamma^\nu \gamma^7 \gamma^\mu = \gamma^\nu \gamma^\mu \gamma^6$.

```
In[117]:= DiracSimplify[DiracMatrix[6, ν, 7, μ]]

Out[117]= γ^ν γ^μ γ^6
```

This is $(\not{p} - m)\,u(p) = 0$.

```
In[118]:= DiracSimplify[(DiracSlash[p] - m) .
          SpinorU[p, m]]

Out[118]= 0
```

Here is the Dirac equation for $v(p)$: $(\not{p} + m)\,v(p) = 0$.

```
In[119]:= DiracSimplify[(DiracSlash[p] + m) .
          SpinorV[p, m]]

Out[119]= 0
```

For the conjugate spinor:
$\overline{u}\,(\not{p} - m) = 0$.

```
In[120]:= DiracSimplify[SpinorUBar[p, m] .
          (DiracSlash[p] - m)]

Out[120]= 0
```

This is $\overline{v}\,\not{q}\,(\not{p}-m) = 2\,\overline{v}\,p\cdot q$.

```
In[121]:= DiracSimplify[SpinorVBar[p, m] .
            DiracSlash[q] .  (DiracSlash[p] - m)]

Out[121]= 2 p·q φ(-p, m)
```

Also more complicated structures are simplified; for example, $\overline{v}(p)\,\not{q}\,\not{p}\,u(q) = \overline{v}(p)\,u(p)\,[\,2\,(p\cdot q) + m_1\,m_2\,]$.

```
In[122]:= DiracSimplify[SpinorVBar[p, m1] .
            DiracSlash[q, p] . SpinorU[q, m2]]

Out[122]= φ(-p, m1) . φ(q, m2) (m1 m2 + 2 p·q)
```

The behaviour of **DiracSimplify** may be tuned with the setting of various options.

option name	*default value*	
DiracCanonical	**False**	use **DiracOrder** internally
DiracSigmaExplicit	**True**	substitute the explicit representation of σ (also a function)
DiracSimpCombine	**False**	try merging **DiracGamma**s in **DiracGamma[.. + .. +]**'s
DiracSubstitute67	**False**	substitute the explicit representation of γ^6 and γ^7
Expanding	**True**	when set to **False** only a limited set of simplification rules are used
Factoring	**False**	factor canonically
InsideDiracTrace	**False**	assume the expression being simplified is inside a Dirac trace

Options for **DiracSimplify**.

DiracReduce reduces all four-dimensional Dirac matrices to the standard basis (S, P, V, A, T) using the Chisholm identity. In the result the basic Dirac structures are wrapped with a head **DiracBasis**. I.e., S corresponds to **DiracBasis[1]**, P to **DiracBasis[DiracMatrix[5]]**, V to **DiracBasis[DiracMatrix[μ]]**, A to **DiracBasis[DiracMatrix[μ, 5]]**, T to **DiracBasis[DiracSigma[DiracMatrix[μ, ν]]]**. By default **DiracBasis** is substituted with **Identity**. Notice that the result of **DiracReduce** is given in **FeynCalcExternal** notation, i.e., evtl. you may want to use **FeynCalcInternal** on the result.

option name	*default value*	
Factoring	**False**	factor canonically
FinalSubstitutions	**{}**	substitutions done at the end of the calculation

Options for **DiracReduce**.

Reducing a product of two Dirac matrices to a standard basis.

```
In[123]:= DiracReduce[DiracMatrix[μ, ν]]
```

$$\text{Out[123]}= g^{\mu\nu} - \sigma^{\mu\nu}$$

Reducing a product of three Dirac matrices to a standard basis.

```
In[124]:= DiracReduce[DiracMatrix[μ, ν, ρ]]
```

$$\text{Out[124]}= i\gamma^{\text{Mu(1)}}.\gamma^5\,\epsilon^{\mu\nu\rho\text{Mu(1)}} + \gamma^\rho\, g^{\mu\nu} - \gamma^\nu\, g^{\mu\rho} + \gamma^\mu\, g^{\nu\rho}$$

Reducing a product of four Dirac matrices to a standard basis.

```
In[125]:= DiracReduce[DiracMatrix[μ, ν, ρ, σ]]
```

$$\text{Out[125]}= -i\,\gamma^5\,\epsilon^{\mu\nu\rho\sigma} - i\,\sigma^{\rho\sigma}\,g^{\mu\nu} + i\,\sigma^{\nu\sigma}\,g^{\mu\rho} - i\,\sigma^{\nu\rho}\,g^{\mu\sigma} - i\,\sigma^{\mu\sigma}\,g^{\nu\rho} + g^{\mu\sigma}\,g^{\nu\rho} + i\,\sigma^{\mu\rho}\,g^{\nu\sigma} - g^{\mu\rho}\,g^{\nu\sigma} - i\,\sigma^{\mu\nu}\,g^{\rho\sigma} + g^{\mu\nu}\,g^{\rho\sigma}$$

Contract the square of the this.

```
In[126]:= Contract[DiracSimplify[% . %]]
```

$$\text{Out[126]}= -128$$

Calc does the full job.

```
In[127]:= Calc[%% . %%]
```

$$\text{Out[127]}= -128$$

Calc[*expr***]**	applies **DotSimplify**, **DiracSimplify**, **EpsEvaluate**, **Contract** and other functions to *expr*, trying to reduce to the simplest form

The highest-level FeynCalc simplification function.

As mentioned in section 3.2, basic features of the Breitenlohner-Maison scheme [16] (for a short explanation of the Breitenlohner-Maison symbols like $\hat{\gamma}^\mu$, see e.g. [17]) are implemented. Below are given some simple illustrations.

Setting the Breitenlohner-Maison γ^5 scheme.

```
In[128]:= $BreitMaison = True;
```

Entering $\gamma^5\,\gamma^\mu$, with γ^μ D-dimensional.

```
In[129]:= DiracMatrix[5] . DiracMatrix[μ,
          Dimension →D]
```

$$\text{Out[129]=}\quad \gamma^5 . \gamma^\mu$$

Now only the 4-dimensional part of γ^μ anticommutes, while the $D-4$ dimensional part $2\,\hat{\gamma}^\mu$ commutes.

```
In[130]:= DiracSimplify[%]
```

$$\text{Out[130]=}\quad 2\,\hat{\gamma}^\mu . \gamma^5 - \gamma^\mu . \gamma^5$$

Project out the positive chirality part of γ^μ.

```
In[131]:= DiracMatrix[6] . DiracMatrix[μ,
          Dimension→D]
```

$$\text{Out[131]=}\quad \gamma^6 . \gamma^\mu$$

The expression is expanded and γ^5 is moved to the right.

```
In[132]:= DiracSimplify[%]
```

$$\text{Out[132]=}\quad \frac{\gamma^\mu}{2} + \hat{\gamma}^\mu . \gamma^5 - \frac{\gamma^\mu . \gamma^5}{2}$$

Go back to naive γ^5 scheme (for some operations a kernel restart is necessary).

```
In[133]:= $BreitMaison = True;
```

4.3 Dirac Traces

The function **DiracTrace** takes as *expr* any expression with "." as noncommutative multiplication operator.

The default of **DiracTrace** is not to evaluate Dirac traces directly. For direct calculation the function **Tr** can be used.

DiracTrace[*expr***]**	head of a Dirac trace
Tr[*expr***]**	calculate the trace directly

Two Dirac trace functions.

$\mathrm{tr}(\gamma^\alpha \gamma^\beta) = 4\, g^{\alpha\beta}$.

```
In[134]:= Tr[DiracMatrix[α, β]]

Out[134]= 4 g^(α β)
```

$\mathrm{tr}(\not{a}\not{b}\not{c}\not{d}) = 4\{(a\cdot d)(b\cdot c) - (a\cdot c)(b\cdot d) + (a\cdot b)(c\cdot d)\}$.

```
In[135]:= Tr[DiracSlash[a, b, c, d]]

Out[135]= 4 (a·d  b·c - a·c  b·d + a·b  c·d)
```

$\mathrm{tr}(\gamma^a \gamma^b \gamma^c \gamma^d \gamma^5) = -4\, i\, \varepsilon^{a b c d}$.

```
In[136]:= Tr[DiracMatrix[α, β, γ, δ, 5]]

Out[136]= -4 i ε^(α β γ δ)
```

You may include metric tensors or four-vectors, for example, $\mathrm{tr}(\frac{1}{4} g^{\alpha\beta} \gamma^\mu \gamma^\alpha p^\mu) = p^\beta$.

```
In[137]:= Tr[MetricTensor[α, β]/4
          DiracMatrix[μ].DiracMatrix[α]
          FourVector[p, μ]] // Contract

Out[137]= p^β
```

If you want to do more complicated traces it is often convenient to introduce your own abbreviations. The following examples, some of which verify results given in [21], show how to do this.

Consider a trace corresponding to the square of the s-channel diagram for γe scattering:

$$T_1 = \frac{1}{16}\,\mathrm{tr}[\,(\not{p}' + m)\,\gamma^\alpha\,(\not{p} + \not{k} + m)\,\gamma^\beta\,(\not{p} + m)\,\gamma^\beta\,(\not{p} + \not{k} + m)\,\gamma^\alpha\,]$$

Set the abbreviations for Dirac matrices and slashes here.

```
In[138]:= pps = DiracSlash[p'];
          ps = DiracSlash[p];
          ks = DiracSlash[k];
          a = DiracMatrix[α];
          b = DiracMatrix[β];
```

This is the input for the trace T_1.

```
In[139]:= Tr[(pps + m).a.(ps + ks + m).b.(ps +
          m).b.(ps + ks + m).a/16]//Expand

Out[139]= 4 m^4 + 4 k^2 m^2 + 4 k·p m^2 - 4 k·p' m^2 - 3 p·p' m^2 +
          2 k·p k·p' + 2 k·p' p^2 - k^2 p·p' + p^2 p·p'
```

Clear symbols used.

```
In[140]:= Clear[pps, ps, ks, a, b];
```

Another nontrivial example is a D-dimensional trace involving 14 Dirac matrices:

$$T_2 = \mathrm{tr}(\gamma^\beta \gamma^\alpha \not{p}_1 \not{p}_2 \gamma^\nu \gamma^\beta \not{p}_2 \not{p}_3 \gamma^\alpha \not{p}_1 \gamma^\nu \not{p}_3 \not{p}_1 \not{p}_2)$$

This defines abbreviations for trace T_2. a, b, n denote $\gamma^\alpha, \gamma^\beta, \gamma^\nu$ in D dimensions. The last command sets **ps1** to $\not{p}_1$, **ps2** to $\not{p}_2$ and **ps3** to $\not{p}_3$.

```
In[141]:= a = DiracMatrix[α, Dimension → D];
          b = DiracMatrix[β, Dimension → D];
          n = DiracMatrix[ν, Dimension → D];
          {ps1, ps2, ps3} = Map[DiracSlash, {p1,
          p2, p3}];
```

Here is the input for trace T_2. The result is again collected with respect to scalar products.

```
In[142]:= Tr[b.a.ps1.ps2.n.b.ps2.ps3.a.ps1.n.ps3.ps1.ps2,
          PairCollect → True]
```

$$\begin{aligned}\text{Out[142]}= {} & 4\,((-288 + 224\,D - 56\,D^2 + 4\,D^3)\,p_1\cdot p_2\,p_1\cdot p_3{}^2\,p_2^2 + \\ & (256 - 128\,D + 16\,D^2)\,p_1\cdot p_2{}^2\,p_1\cdot p_3\,p_2\cdot p_3 + (112 - \\ & 104\,D + 28\,D^2 - 2\,D^3)\,p_1^2\,p_1\cdot p_3\,p_2^2\,p_2\cdot p_3 + (-128 + 64\,D - \\ & 8\,D^2)\,p_1^2\,p_1\cdot p_2\,p_2\cdot p_3{}^2 + (-128 + 64\,D - 8\,D^2)\,p_1\cdot p_2{}^3\,p_3^2 + \\ & (168 - 104\,D + 20\,D^2 - D^3)\,p_1^2\,p_1\cdot p_2\,p_2\cdot p_2\,p_3^2)\end{aligned}$$

This calculates T_2 in four dimensions. Since the "**.**" is used, the replacement **D → 4** applies to all Dirac matrices. The time needed would be twice as much without calculating the D-dimensional case before.

```
In[143]:= Tr[b.a.ps1.ps2.n.b.ps2.ps3.a.ps1.n.ps3.ps1.ps2
          /. D →4, PairCollect → True]
```

$$\begin{aligned}\text{Out[143]}= {} & 4\,(-32\,p1\cdot p2\,p2^2\,p1\cdot p3^2 + 16\,p1^2\,p2^2\,p2\cdot p3\,p1\cdot p3 + \\ & 8\,p1^2\,p1\cdot p2\,p2^2\,p3^2)\end{aligned}$$

Clear symbols used.

```
In[144]:= Clear[a, b, n, ps1, ps2, ps3];
```

Sometimes you do not want a trace to be evaluated immediately. Here you get the input $\mathrm{tr}(\gamma^\alpha\,\gamma^\beta\,\gamma^\rho\,\gamma^\sigma)$ back (typeset).

```
In[145]:= DiracTrace[DiracMatrix[α, β, ρ, σ]]
```

$$\text{Out[145]}= \mathrm{tr}(\gamma^\alpha\,\gamma^\beta\,\gamma^\rho\,\gamma^\sigma)$$

You may then contract, e.g., with $g^{\alpha\beta}$.

```
In[146]:= Contract[% MetricTensor[α, β]]
```

$$\text{Out[146]}= \mathrm{tr}(\gamma^\beta\,\gamma^\beta\,\gamma^\rho\,\gamma^\sigma)$$

This evaluates the Dirac trace.

```
In[147]:= % /.  DiracTrace → Tr
```

$$\text{Out[147]}= 16\,g^{\rho\sigma}$$

option name	*default value*	
DiracTraceEvaluate	**False**	evaluate the trace
LeviCivitaSign	**-1**	which sign convention to use in the result of tr($\gamma^a \gamma^b \gamma^c \gamma^d \gamma^5$). The default gives $(-1) 4\, i\, \varepsilon^{a\,b\,c\,d}$
Factoring	**False**	factor canonically
Mandelstam	**{}**	utilize the Mandelstam relation
PairCollect	**True**	collect **Pair**s
Schouten	**0**	maximum number of terms on which to apply the Schouten identity
TraceOfOne	**0**	the trace of an identity matrix
FeynCalcExternal	**0**	give output in **FeynCalcExternal** form (see section 3)
EpsContract	**False**	contract Levi-Civita **Eps**

Options for **DiracTrace**.

Tr takes the options of **DiracTrace**, but the default setting of **DiracTraceEvaluate** is **True**. Additionally, **Tr** takes the two options **SUNTrace** and **SUNNToCACF**, which control if and how SU(N) traces are evaluated. This is elaborated in section 4.4.

The option **PairCollect** determines whether the resulting polynomial is collected with respect to metric tensors, four-vectors and scalar products. In the internal representation these three objects have the same head **Pair**, hence the name **PairCollect**.

For $2 \to 2$ processes the traces are often expressed in terms of Mandelstam variables. In order to replace these for the scalar products you can use **SetMandelstam**.

SetMandelstam[s, t, u, p_1, p_2, p_3, p_4, m_1, m_2, m_3, m_4**]**	define scalar products in terms of Mandelstam variables and put the p_i on-shell

A function for introducing Mandelstam variables.

Assuming all p_i incoming, i.e., $p_1 + p_2 + p_3 + p_4 = 0$, the Mandelstam variables are defined by

$$s = (p_1 + p_2)^2, t = (p_1 + p_3)^2, u = (p_1 + p_4)^2.$$

Using these three equations and the on-shell conditions, $p_i^2 = m_i^2$, **SetMandelstam** sets the 10 possible scalar products $(p_i \cdot p_j)$ in terms of s, t, u and m_i^2.
For calculation of traces the Mandelstam relation

$$s + t + u = m_1^2 + m_2^2 + m_3^2 + m_4^2$$

can often be used to get a compact result. If you set the option

```
Mandelstam → {s, t, u, m1^2 + m2^2 + m3^2 + m4^2}
```

FeynCalc tries to figure out the best choice of s, t or u in each factor of the result.
As an example for calculating a trace in terms of Mandelstam variables, consider the following squared amplitude from the process $gg \to t\bar{t}$, with $\Sigma_1^{\alpha\rho}$ and $\Sigma_2^{\beta\rho}$ as polarization sums for the gluons.

$$\begin{aligned} T_3 &= \mathrm{tr}(\gamma^\sigma (\not{k}_1 - \not{p}_1 - m_t)\gamma^\rho (\not{p}_1 + m_t)(\not{p}_2 - m_t))\, p_1^\alpha\, p_2^\beta\, \Sigma_1^{\alpha\rho}\, \Sigma_2^{\beta\sigma}, \\ \Sigma_1^{\alpha\rho} &= -g^{\alpha\rho} + \tfrac{4}{(u-t)^2}(4m_t^2 - s)\, k_1^\alpha k_1^\rho + \tfrac{2}{u-t}[k_1^\rho (p_1 - p_2)^\alpha + k_1^\alpha (p_1 - p_2)^\rho] \\ \Sigma_2^{\beta\sigma} &= -g^{\beta\sigma} + \tfrac{4}{(t-u)^2}(4m_t^2 - s)\, k_2^\beta k_2^\sigma + \tfrac{2}{t-u}[k_2^\sigma (p_1 - k_2)^\beta + k_2^\beta (p_1 - p_2)^\sigma] \end{aligned}$$

Set up s, t, u for $gg \to t\bar{t}$, with k_1, k_2 as gluon and p_1, p_2 as fermion momenta. Again abbreviations are introduced for the Dirac matrices and slashes. **polsum1** and **polsum2** are the polarization sums for the gluons.

```
In[148]:= SetMandelstam[s,t,u, k1,k2,-p1,-p2,
          0,0,m,m];
          {ks1, ps1, ps2} = Map[ DiracSlash, {k1,
          p1, p2} ];
          {si, ro} = Map[ DiracMatrix,{σ, ρ} ];
          polsum1 = PolarizationSum[α, ρ, k1, p1 -
          p2];
          polsum2 = PolarizationSum[β, σ, k2, p1 -
          p2];
          p1al = FourVector[p1, alpha];
          p2be = FourVector[p2, β];
```

This is a possible input for the trace T_3. FeynCalc contracts first all Lorentz indices and then calculates the trace.
Since the option **Mandelstam** has been specified, the result is given in a factored form, where in each factor one of s, t or u is eliminated via the Mandelstam relation. Note that a factor $(t - u)$ has been cancelled.

```
In[149]:= Tr[ (polsum1 polsum2 p1al p2be si) .
          (ks1 - ps1 - m) . ro . (ps1 + m) .
          (ps2 - m), Mandelstam → {s, t, u, 2 m^2}
          ]
```

$$\text{Out[149]} = -\frac{2\, m\, s\, (m^4 - t\, u)\, (8\, m^4 - t^2 - u^2 - 6\, t\, u)}{(t - u)^3}$$

An alternative method would be to first calculate the trace without the polarization sums.

```
In[150]:= Tr[si, ks1 - ps1 - m, ro, ps1 + m, ps2 -
          m]
```

$$\text{Out[150]= } (-2\,m\,t + 2\,m\,u)\,g^{\rho\sigma} - 4\,m\,\text{k1}^{\sigma}\,\text{p1}^{\rho} - 4\,m\,\text{k1}^{\rho}\,\text{p1}^{\sigma} + 8\,m\,\text{p1}^{\rho}\,\text{p1}^{\sigma} + 4\,m\,\text{k1}^{\sigma}\,\text{p2}^{\rho} + 4\,m\,\text{k1}^{\rho}\,\text{p2}^{\sigma} - 8\,m\,\text{p1}^{\rho}\,\text{p2}^{\sigma}$$

Then contract the result with the polarization sums, expand the scalar products and use **TrickMandelstam** (see section 6.7) in order to get the Mandelstam variable substitution. This method is faster; but that is not the case for all trace calculations.

```
In[151]:= TrickMandelstam[ExpandScalarProduct[Contract[
          % polsum1 polsum2 p1al p2be]], {s, t, u,
          2 m^2}]
```

$$\text{Out[151]= } -\frac{2\,m\,s\,(m^4 - t\,u)\,(8\,m^4 - t^2 - u^2 - 6\,t\,u)}{(t-u)^3}$$

Clear symbols used.

```
In[152]:= Clear[ks1,ps1,ps2,si,ro,polsum1,polsum2,p1al,p2be];
```

Since Dirac matrices can be given in any number of dimensions, FeynCalc is also able to calculate traces in e.g. $d-4$ dimensions.

Defining $T(n) = \text{tr}(\gamma_{\mu_1}\gamma_{\mu_2}...\gamma_{\mu_n}\gamma_{\mu_1}\gamma_{\mu_2}...\gamma_{\mu_n})$ we give a list of timings and results for $T(8)$ to $T(11)$. The trace $T(10)$ is a verification of the result given in [20].

This is a little program defining t. The dimension of each particular Dirac matrix is set to $d-4$. The calculations were done with Mathematica 4.2 under Linux on a 1.8 GHz Pentium 4 box with 256 MB of RAM.

```
In[153]:= t[n_] := t[n] = Block[{gammas, calc},
          gammas = Dot @@ Table[
          DiracMatrix[a[i], Dimension → (d - 4)],
          {i, 1, n} ];
          calc = Timing[ Tr[ gammas . gammas ] //
          Expand ];
          Print["Time = ", calc[[1]] ];
          calc[[2]]];
```

This calculates a trace of 16 matrices.

```
In[154]:= t[8]

Time = 0.58 Second
```

$$\text{Out[154]= } 123469824 - 135962624\,d + 63224832\,d^2 - 16145920\,d^3 + 2461760\,d^4 - 227584\,d^5 + 12320\,d^6 - 352\,d^7 + 4\,d^8$$

Here we have 18.

```
In[155]:= t[9]

Time = 1.6 Second
```

$$\text{Out[155]= } -1879576576 + 2220901376\,d - 1127626752\,d^2 + 321806848\,d^3 - 56625408\,d^4 + 6331584\,d^5 - 446208\,d^6 + 18912\,d^7 - 432\,d^8 + 4\,d^9$$

The trace of 20 Dirac matrices.

```
In[156]:= t[10]

Time = 5.88 Second

Out[156]= -31023169536 + 38971179008 d - 21328977920 d^2 +
          6679521280 d^3 - 1320732160 d^4 + 171464832 d^5 -
          14710080 d^6 + 816960 d^7 - 27840 d^8 + 520 d^9 - 4 d^10
```

With 22 Dirac matrices it gets slow.

```
In[157]:= t[11]

Time = 23.15 Second

Out[157]= 551768735744 - 731506905088 d + 427299186688 d^2 -
          144858475520 d^3 + 31576821760 d^4 - 4629805312 d^5 +
          463655808 d^6 - 31521600 d^7 + 1415040 d^8 - 39600 d^9 +
          616 d^10 - 4 d^11
```

Clean up.

```
In[158]:= Clear[t];
```

4.4 SU(N) Traces and Algebra

The functions for calculations with SU(N) matrices and the corresponding structure constants were developed for calculations in N-dimensional color space. More specialized functions developed for calculations in 2- and 3-dimensional flavor space (Pauli and Gell-Mann matrices) are proveded by the subpackage PHI. These are described in the PHI user's guide ???.

SUNTrace[*expr***]**	calculate the trace of SU(N) matrices

Trace calculation of SU(N) matrices in the fundamental representation.

Like Dirac traces, traces of the SU(N) matrices T_i are calculated algebraically. The matrices are assumed to be in the fundamental representation and traces are given in terms of N.

option name	*default value*	
Explicit	**False**	evaluate the trace

Option for **SUNTrace**.

SUNDeltaContract[*expr***]**	contract **SUNDelta**s
SUNSimplify[*expr***]**	simplify polynomials in the SU(N) Kronecker delta δ_{ij} and structure functions f_{ijk}, d_{ijk} and generating matrices T_i

Functions for simplifying expressions with SU(N) matrices and structure functions.

The result of **SUNSimplify** involves either N or the Casimir invariants C_A and C_F. Which, depends on the setting of the option **SUNNToCACF**.

SUNN	the N of SU(N)
CA	$C_A = N$ in the funcamental representation
CF	$C_F = (N^2 - 1)/(2N)$ in the funcamental representation

Casimir invariants of SU(N).

option name	*default value*	
SUNTrace	**False**	if set to False, then any SUNT-matrices are taken out of **DiracTrace[...]**; otherwise a color-trace is taken (by **SUNTrace**) before taking the SU(N) objects in front of **DiracTrace[...]**
Explicit	**False**	express structure functions (f_{ijk}, d_{ijk}) in terms of traces of generator matrices (T_i)
Factoring	**False**	factor the result
SUNIndexRename	**True**	rename contracted SU(N) indices systematically
SUNFJacobi	**False**	use the Jacobi identity
SUNNToCACF	**True**	express the result in terms of the Casimir invariants C_A and C_F instead of N
Expanding	**False**	do noncommutative expansion

Options for **SUNSimplify**.

To evaluate f_{abc}, use the option **Explicit → True**.

```
In[159]:= SUNF[a, b, c, Explicit → True]
```

$$\text{Out[159]=}\quad 2i\,(\mathrm{tr}(T_a T_c T_b) - \mathrm{tr}(T_a T_b T_c))$$

$\mathrm{tr}(T_a T_b) = \delta_{ab}/2$.

```
In[160]:= SUNTrace[SUNT[a] . SUNT[b]]
```

$$\text{Out[160]=}\quad \frac{\delta_{ab}}{2}$$

$\mathrm{tr}(3\, T_a T_b T_a T_b) = 1/(4N) - N/4$.

```
In[161]:= SUNTrace[SUNT /@ (a.b.a.b)]
```

$$\text{Out[161]=}\quad \frac{1}{4N} - \frac{N}{4}$$

$\mathrm{tr}(T_a T_b T_c)$ stays as it is.

```
In[162]:= SUNTrace[SUNT /@ (a.b.c)]
```

$$\text{Out[162]=}\quad \mathrm{tr}(T_a T_b T_c)$$

$\mathrm{tr}(T_a T_b T_c f_{abc}) = i C_A^2 C_F / 2.$

```
In[163]:= SUNTrace[SUNF[a, b, c] SUNT /@ (a.b.c)]
          // SUNSimplify
```

$$\text{Out[163]=}\ \frac{1}{2} i C_A^2 C_F$$

$\mathrm{tr}(f_{ars} f_{brs}) = C_A f_{ars} f_{brs} = C_A^2 \delta_{ab}.$

```
In[164]:= SUNF[a, r, s] SUNF[b, r, s] //
          SUNSimplify
```

$$\text{Out[164]=}\ C_A^2 \delta_{ab}$$

The Jacobi identity: $f_{abr} f_{rcs} + f_{bcr} f_{ras} + f_{car} f_{rbs} = 0.$

```
In[165]:= SUNSimplify[SUNF[a,b,r] SUNF[r,c,s] +
          SUNF[b,c,r] SUNF[r,a,s] + SUNF[c,a,r]
          SUNF[r,b,s], SUNFJacobi → True]
```

$$\text{Out[165]=}\ 0$$

$T_a T_b T_a = -\frac{1}{2}(C_A - 2C_F) T_b.$

```
In[166]:= SUNT[a, b, a] // SUNSimplify
```

$$\text{Out[166]=}\ -\frac{1}{2}(C_A - 2C_F) T_b$$

This is $f_{abc} T_b T_c = i C_A T_a / 2.$

```
In[167]:= SUNF[c, a, b] SUNT[b, c] // SUNSimplify
```

$$\text{Out[167]=}\ \frac{1}{2} i C_A T_a$$

4.5 Green's functions

Quantum fields (see section 2.7) can be combined in polynomials to form lagrangians. From such lagrangians, the Green's function or Feynman rule of an interaction vertex can be found by calculating functional derivatives with respect to the fields of the vertex. As a very simple example, consider the vertex obtained from the following term of the QED lagrangian:

Mathematica (FeynCalc) notation for $e A_\mu \bar{\psi} \gamma^\mu \psi$.

```
In[168]:= e QuantumField[γ, {μ}].QuantumField[ψ̄].
          DiracMatrix[{μ}].QuantumField[ψ]
```

$$\text{Out[168]=}\ e A_\mu \bar{\psi} \gamma^\mu \psi$$

Calculation of the $\psi\overline{\psi}\gamma_{\mu_3}$ Feynman rule.

```
In[169]:= FeynRule[%,
            {QuantumField[ψ][p1],
            QuantumField[ψ̄][p2],
            QuantumField[γ, {μ3}][p3]}]

Out[169]= ieγ^μ3
```

Notice that **FeynRule** does not write out the momentum conserving $\delta(p_1 + p_2 + p_3)$. As we shall see later, this is anyway enforced if the vertex is used for Feynman diagram calculations with FeynArts.

Besides calculating the functional derivative, **FeynRule** does some simplification on the result and transforms to momentum space. To calculate the functional derivative **FeynRule** uses a lower level function: **FunctionalD**. Contrary to **FeynRule** **FunctionalD** does not do any cleaning up on the result and may therefore be faster but return longer expressions.

By default **FunctionalD** operates in position space. However, no explicit space-time symbols should be given and none will be returned. Thus, a few conventions are used: Instead of the usual $\delta\phi(x)/\delta\phi(y) = \delta^{(D)}(x - y)$ the arguments and the δ function are omitted, i.e., for simplicity $\delta\phi/\delta\phi$ is taken to be 1. Similarly, instead of the usual $\delta\partial_\mu\phi(x)/\delta\phi(y) = \partial_\mu\delta^{(D)}(x - y)$ the arguments are omitted, and the ∂_μ operator is specified by default to be an integration by parts operator, i.e., the right-hand side will be just $-\partial_\mu$ or, more precisely (by default), $-\overrightarrow{\partial}_\mu$.

If the **QuantumField**s of the first argument (e.g. a lagrangian) of **FunctionalD** are given an extra argument (e.g. **QuantumField[...][*p*]**), the argument is assumed to be a momentum and transformation to momentum space is done.

FunctionalD[*expr*, *fi1***[***p1***], *fi2***[***p2***], ...]**	calculates the functional derivative of *expr* with respect to quantum fields *fi1*, *fi2*, $\cdots$ and does the Fourier transform to field momenta *p1*, *p2*, $\cdots$
FunctionalD[*expr*, *fi1*, *fi2*, ...**]**	calculates the functional derivative of *expr* and does partial integration but omits the x-space delta functions.
FeynRule[*lag*, *fi1***[***p1***],** *fi2***[***p2***], ...]**	calculates the Feynman rule of *lag* with respect to fields *fi1*, *fi2*, $\cdots$ and momenta *p1*, *p2*, $\cdots$

Calculation of Green's functions.

Define a two-gluon product **aa** with contracted indices.

```
In[170]:= aa = QuantumField[PartialD[LorentzIndex[β]],
          GaugeField, LorentzIndex[α],
          SUNIndex[a]].QuantumField[
          PartialD[LorentzIndex[β]], GaugeField,
          LorentzIndex[α], SUNIndex[a]]
```

$$\text{Out[170]=}\ \partial_\beta A^a_\alpha\, \partial_\beta A^a_\alpha$$

Calculate the functional derivative.

```
In[171]:= dd = FunctionalD[aa,
          QuantumField[GaugeField, {μ1}, {i1}],
          QuantumField[GaugeField, {μ2}, {i2}]]
```

$$\text{Out[171]=}\ -2\overrightarrow{\partial}_\beta.\overrightarrow{\partial}_\beta g^{\mu 1\mu 2}\delta_{\text{i1i2}}$$

See the internal representation. Notice the symbol **RightPartialD**.

```
In[172]:= dd // StandardForm
```

```
Out[172]= -2 RightPartialD[LorentzIndex[β]] .
          RightPartialD[LorentzIndex[β]]
          Pair[LorentzIndex[μ1], LorentzIndex[μ2]]
          SUNDelta[SUNIndex[i1], SUNIndex[i2]]
```

We can apply **dd** to some other field ψ.

```
In[173]:= dd .  QuantumField[ψ, {μ1}, {i1}] //
          ExpandPartialD
```

$$\text{Out[173]=}\ -2g^{\mu 1\mu 2}(\partial_\beta\partial_\beta\psi^{\text{i2}}_{\mu 1})$$

See the internal representation. Notice that applying **ExpandPartialD** causes **RightPartialD** to disappear in favour of **PartialD**.

```
In[174]:= dd // StandardForm
```

```
Out[174]= -2 SUNDelta[SUNIndex[a], SUNIndex[i2]].
          SUNDelta[SUNIndex[a], SUNIndex[i1]].
          QuantumField[PartialD[LorentzIndex[β]],
          PartialD[LorentzIndex[β]], ψ,
            LorentzIndex[μ1], SUNIndex[i1]]
          Pair[LorentzIndex[μ1], LorentzIndex[μ2]]
```

Calculate the 2-gluon Feynman rule (in momentum space).

```
In[175]:= FunctionalD[aa, {QuantumField[GaugeField,
          {μ1}, {i1}][p1], QuantumField[GaugeField,
          {μ2}, {i2}][p2]}]
```

$$\text{Out[175]=}\ \left(-ig^{\alpha\mu 1}\text{p1}^\beta\delta_{\alpha\text{i1}}\right).\left(-ig^{\alpha\mu 2}\text{p2}^\beta\delta_{\alpha\text{i2}}\right)+\left(-ig^{\alpha\mu 2}\text{p2}^\beta\delta_{\alpha\text{i2}}\right).\left(-ig^{\alpha\mu 1}\text{p1}^\beta\delta_{\alpha\text{i1}}\right)$$

Clear intermediate variables.

```
In[176]:= Clear[aa];
```

RightPartialD[μ]	denotes partial space-time differentiation $\partial/\partial x^\mu$, acting to the right.
LeftPartialD[μ]	denotes partial space-time differentiation $\partial/\partial x^\mu$, acting to the left.
LeftRightPartialD[μ]	denotes partial space-time differentiation $\partial/\partial x^\mu$, acting to the left and right. **ExplicitPartialD[LeftRightPartialD[μ]]** gives **1/ 2 (RightPartialD[μ] - LeftPartialD[μ])**
LeftRightPartialD2[μ]	denotes partial space-time differentiation $\partial/\partial x^\mu$, acting to the left and right. **ExplicitPartialD[LeftRightPartialD[μ]]** gives **(RightPartialD[μ] + LeftPartialD[μ])**
ExplicitPartialD[*exp*]	inserts in *exp* the definitions for **LeftRightPartialD** and **LeftRightPartialD2**.
ExpandPartialD[*exp*]	expands **DOT** products of **RightPartialD**s and/or **LeftPartialD**s with **QuantumField**s in *exp* using the Leibniz rule.

Partial space-time derivatives.

As another example consider the strong QCD 4-gluon vertex. The relevant part of the QCD lagrangian is already known by FeynCalc.

The gluon self-interaction part of the QCD lagrangian .

```
In[177]:= Lagrangian["QCD"]
```

$$\text{Out[177]= } -\frac{1}{4} F^a_{\alpha\beta} . F^a_{\alpha\beta}$$

We could write out the field strength tensors.

```
In[178]:= Lagrangian["QCD"] /.
          FieldStrength[x__] :→
          FieldStrength[x, Explicit → True] //
          DotExpand
```

$$\text{Out[178]}= \frac{1}{4}\Big(-A_{\alpha}^{b11}.A_{\beta}^{c15}.A_{\alpha}^{b12}.A_{\beta}^{c16}\, f_{ab11c15}\, f_{ab12c16}\, g_s^2 - A_{\alpha}^{b11}.A_{\beta}^{c15}.\partial_{\alpha}A_{\beta}^{a}\, f_{ab11c15}\, g_s + A_{\alpha}^{b11}.A_{\beta}^{c15}.\partial_{\beta}A_{\alpha}^{a}\, f_{ab11c15}\, g_s - \partial_{\alpha}A_{\beta}^{a}.A_{\alpha}^{b12}.A_{\beta}^{c16}\, f_{ab12c16}\, g_s + \partial_{\beta}A_{\alpha}^{a}.A_{\alpha}^{b12}.A_{\beta}^{c16}\, f_{ab12c16}\, g_s - \partial_{\alpha}A_{\beta}^{a}.\partial_{\alpha}A_{\beta}^{a} + \partial_{\alpha}A_{\beta}^{a}.\partial_{\beta}A_{\alpha}^{a} + \partial_{\beta}A_{\alpha}^{a}.\partial_{\alpha}A_{\beta}^{a} - \partial_{\beta}A_{\alpha}^{a}.\partial_{\beta}A_{\alpha}^{a}\Big)$$

But don't need to do so to calculate the 4-gluon Feynman rule.

```
In[179]:= FeynRule[Lagrangian["QCD"], {
          QuantumField[GaugeField, {μ1},
          {i1}][p1],
          QuantumField[GaugeField, {μ2},
          {i2}][p2],
          QuantumField[GaugeField, {μ3},
          {i3}][p3],
          QuantumField[GaugeField, {μ4},
          {i4}][p4]}]
```

$$\text{Out[179]}= i\left(g^{\mu1\mu3}\, g^{\mu2\mu4} - g^{\mu1\mu2}\, g^{\mu3\mu4}\right) f_{i1i4si3}\, f_{i2i3si3}\, g_s^2 + i\left(g^{\mu1\mu4}\, g^{\mu2\mu3} - g^{\mu1\mu2}\, g^{\mu3\mu4}\right) f_{i1i3si3}\, f_{i2i4si3}\, g_s^2 + i\left(g^{\mu1\mu4}\, g^{\mu2\mu3} - g^{\mu1\mu3}\, g^{\mu2\mu4}\right) f_{i1i2si3}\, f_{i3i4si3}\, g_s^2$$

The function **Lagrangian** reads a database of lagrangians and returns the one corresponding to the name (a string) given as argument. Currently the following names are known by **Lagrangian**:

- **Lagrangian["oqu"]** gives the unpolarized, twist 2, OPE quark operator.
- **Lagrangian["oqp"]** gives the polarized, twist 2, OPE quark operator.
- **Lagrangian["ogu"]** gives the unpolarized, twist 2, OPE gluon operator.
- **Lagrangian["ogp"]** gives the polarized, twist 2, OPE gluon operator.
- **Lagrangian["ogd"]** gives the sigma-term part of the QCD lagrangian.
- **Lagrangian["QCD"]** gives the gluon self interaction part of the QCD lagrangian.

More can be added easily (as done e.g. by the optional subpackage PHI).

Lagrangian["*name*"] returns a lagrangian *name* in FeynCalc notation.

A database of lagrangians.

5 Tensor and Scalar Integrals

In this section is described the capabilities of FeynCalc to reduce one-loop Feynman diagrams. If is also discussed how to provide input for FeynCalc and how to further process the output of FeynCalc.

The methods and conventions implemented in FeynCalc for the evaluation of one-loop diagrams are described in [12] and [13]. The usual Passarino-Veltman scheme for the one-loop integrals is adapted to a large extent [12]. The coefficient functions of the tensor integrals are defined similar to [12], except that the Passarino-Veltman integrals take internal masses squared as arguments.

FeynCalc can reduce all n-point integrals with $n \leq 4$ to scalar integrals A_0, B_0, C_0 and D_0.

For the numerical evaluation of scalar integrals, several possibilities exist: 1) The program FormF by M. Veltman. Unfortunately this only runs in CDC and 68000 assembler and is thus not a realistic alternative. 2) The library FF by G.J. van Oldenborgh [15] written in Fortran 77. The library can be put to use with the LoopTools package [2] written by Thomas Hahn, which features a Fortran, c++, and Mathematica interface. Loading the Mathematica package will simply cause the functions A_0, B_0, C_0 and D_0 (and E_0 and F_0) to return numerical values when given numerical arguments. This is the recommended procedure for obtaining numerical output from FeynCalc results. A more simple Fortran wrapper program to link FF and FeynCalc is available from Rolf Mertig. 3) The optional subpackage PHI provides some functions for evaluating B_0 and C_0. These are written purely in Mathematica but are not very well tested.

5.1 Passarino-Veltman Integrals and Reduction of Coefficient Functions

The scalar integrals A_0, B_0, C_0 and D_0 are represented in FeynCalc as functions with all arguments consisting of scalar products or masses squared. Thus A_0 has one, B_0 three, C_0 six, and D_0 ten arguments. The symmetry properties of the arguments are implemented, i.e., a standard representative of all possible argument permutations of each B_0, C_0 and D_0 is returned. For example, `B0[pp, m2^2, m1^2]` → `B0[pp, m1^2, m2^2]`, where `pp` denotes the scalar product $(p \cdot p) = p^2$.

A0[*m02*]	$(i\pi^2)^{-1}\int d^Dq(q^2-m_0^2)^{-1}$
B0[*p10*, *m02*, *m12*]	$(i\pi^2)^{-1}\int d^Dq([q^2-m_0^2][(q+p_1)^2-m_1^2])^{-1}$
DB0[*p10*, *m02*, *m12*]	$\partial B_0(p_1^2, m_0^2, m_1^2)/\partial p_1^2$
C0[*p10*, *p12*, *p20*, *m02*, *m12*, *m22*]	$(i\pi^2)^{-1}\int d^Dq([q^2-m_0^2][(q+p_1)^2-m_1^2][(q+p_2)^2-m_2^2])^{-1}$
D0[*p10*, *p12*, *p23*, *p30*, *p20*, *p13*, *m02*, *m12*, *m22*, *m32*]	$(i\pi^2)^{-1}\int d^Dq([q^2-m_0^2][(q+p_1)^2-m_1^2][(q+p_2)^2-m_2^2][(q+p_3)^2-m_3^2])^{-1}$

Scalar Passarino-Veltman functions A_0, B_0,B_0', C_0 and D_0.

In FeynCalc and in the mathematical definitions given above, the factor $(2\pi\mu)^{(4-D)}$ with the scaling variable μ is suppressed. The convention for the scalar arguments is $pi0 = p_i^2$ $pij = (p_i - p_j)^2$, $mi2 = m_i^2$.

The B_0 function is symmetric in the mass arguments.

```
In[180]:= B0[s, mz2, mw2]
Out[180]= B0(s, mw2, mz2)
```

Taking the derivative with respect to the first argument yields **DB0**.

```
In[181]:= D[%, s]
Out[181]= DB0(s, mw2, mz2)
```

The tensor-integral decomposition is automatically done by FeynCalc when calculating one-loop amplitudes, but extra functions are provided to reduce the coefficients of the tensor-integral decomposition.

For fixing the conventions of the coefficient functions the definitions of the tensor-integrals and the decomposition are given below. In general the one-loop tensor integral is

$$T^N_{\mu_1\ldots\mu_P}(p_1,\ldots,p_{N-1},m_0,\ldots,m_{N-1}) = \frac{(2\pi\mu)^{4-D}}{i\pi^2}\int d^Dq\frac{q_{\mu_1}\cdots q_{\mu_P}}{\mathcal{D}_0\mathcal{D}_1\cdots\mathcal{D}_{N-1}}$$

with the denominator factors

$$\mathcal{D}_0 = q^2 - m_0^2 \qquad \mathcal{D}_i = (q+p_i)^2 - m_i^2, \qquad i = 1,\ldots,N-1$$

originating from the propagators in the Feynman diagram. The $i\varepsilon$ part of the denominator factors is suppressed.

The tensor integral decompositions for the integrals that FeynCalc can do are listed below. The coefficient functions B_i, B_{ij}, C_i, C_{ij}, C_{ijk}, D_i, D_{ij}, D_{ijk} and D_{ijkl} are totally symmetric in their indices.

$$
\begin{aligned}
B_\mu &= p_{1\mu} B_1 \\
B_{\mu\nu} &= g_{\mu\nu} B_{00} + p_{1\mu} p_{1\nu} B_{11} \\
C_\mu &= p_{1\mu} C_1 + p_{2\mu} C_2 = \sum_{i=1}^{2} p_{i\mu} C_i \\
C_{\mu\nu} &= g_{\mu\nu} C_{00} + p_{1\mu} p_{1\nu} C_{11} + p_{2\mu} p_{2\nu} C_{22} + (p_{1\mu} p_{2\nu} + p_{2\mu} p_{1\nu}) C_{12} \\
&= g_{\mu\nu} C_{00} + \sum_{i,j=1}^{2} p_{i\mu} p_{j\nu} C_{ij} \\
C_{\mu\nu\rho} &= (g_{\mu\nu} p_{1\rho} + g_{\nu\rho} p_{1\mu} + g_{\mu\rho} p_{1\nu}) C_{001} + (g_{\mu\nu} p_{2\rho} + g_{\nu\rho} p_{2\mu} + g_{\mu\rho} p_{2\nu}) C_{002} \\
&\quad + p_{1\mu} p_{1\nu} p_{1\rho} C_{111} + p_{2\mu} p_{2\nu} p_{2\rho} C_{222} \\
&\quad + (p_{1\mu} p_{1\nu} p_{2\rho} + p_{1\mu} p_{2\nu} p_{1\rho} + p_{2\mu} p_{1\nu} p_{1\rho}) C_{112} \\
&\quad + (p_{2\mu} p_{2\nu} p_{1\rho} + p_{2\mu} p_{1\nu} p_{2\rho} + p_{1\mu} p_{2\nu} p_{2\rho}) C_{122} \\
&= \sum_{i=1}^{2} (g_{\mu\nu} p_{i\rho} + g_{\nu\rho} p_{i\mu} + g_{\mu\rho} p_{i\nu}) C_{00i} + \sum_{i,j,k=1}^{2} p_{i\mu} p_{j\nu} p_{k\rho} C_{ijk} \\
D_\mu &= \sum_{i=1}^{3} p_{i\mu} D_i \\
D_{\mu\nu} &= g_{\mu\nu} D_{00} + \sum_{i,j=1}^{3} p_{i\mu} p_{j\nu} D_{ij} \\
D_{\mu\nu\rho} &= \sum_{i=1}^{3} (g_{\mu\nu} p_{i\rho} + g_{\nu\rho} p_{i\mu} + g_{\mu\rho} p_{i\nu}) D_{00i} + \sum_{i,j,k=1}^{3} p_{i\mu} p_{j\nu} p_{k\rho} D_{ijk} \\
D_{\mu\nu\rho\sigma} &= (g_{\mu\nu} g_{\rho\sigma} + g_{\mu\rho} g_{\nu\sigma} + g_{\mu\sigma} g_{\nu\rho}) D_{0000} \\
&\quad + \sum_{i,j=1}^{3} (g_{\mu\nu} p_{i\rho} p_{j\sigma} + g_{\nu\rho} p_{i\mu} p_{j\sigma} + g_{\mu\rho} p_{i\nu} p_{j\sigma} + g_{\mu\sigma} p_{i\nu} p_{j\rho} + g_{\nu\sigma} p_{i\mu} p_{j\rho} + g_{\rho\sigma} p_{i\mu} p_{j\nu}) D_{00ij} \\
&\quad + \sum_{i,j,k,l=1}^{3} p_{i\mu} p_{j\nu} p_{k\rho} p_{l\sigma} D_{ijkl}
\end{aligned}
$$

All coefficient functions and the scalar integrals are summarized in one generic function, **`PaVe`**.

PaVe[*i, j, ...,* **{***P10, P12, ...}, {m02, m12, ...}]* Passarino-Veltman coefficient functions

Passarino-Veltman coefficient functions of the tensor integral decomposition.

The first set of arguments *i, j, ...* are exactly those indices of the coefficient functions of the tensor integral decomposition. If only a **0** is given as first argument, the scalar integrals are understood. The last argument, the list of inner masses m_0^2, m_1^2, ..., determines whether a one-, two-, three- or four-point function is meant. **PaVe** is totally symmetric in the *i, j, ...* arguments. The foremost argument is the list of scalar products of the p_i. They are the same as defined above for the scalar B_0, C_0 and D_0 functions. For A_0 an empty list has to be given.

A certain set of special **PaVe** shown in the following examples simplify to the usual notation.

To shorten the input squared masses are abbreviated with a suffix 2, i.e., a mass m^2 is denoted by **m2**. The scalar quantity p^2 is entered as **pp**, p_i^2 as **pi0** and $(p_i - p_j)^2$ as **pij**.

For the following examples some options are set.

```
In[182]:= SetOptions[A0, A0ToB0 → False];
          SetOptions[{B1, B00, B11}, BReduce →
          False];
```

This is the scalar Passarino-Veltman one-point function.

```
In[183]:= PaVe[0, {}, {m02}]

Out[183]= A0(m02)
```

This is the two-point function $B_0(p^2, m_0^2, m_1^2)$.

```
In[184]:= PaVe[0, {pp}, {m02, m12}]

Out[184]= B0(pp, m02, m12)
```

Here is the coefficient function $B_1(p^2, m_0^2, m_1^2)$.

```
In[185]:= PaVe[1, {pp}, {m12, m22}]

Out[185]= B1(pp, m12, m22)
```

Coefficient functions of metric tensors have two "0" indices for each $g^{\mu\nu}$.

```
In[186]:= PaVe[0,0, {pp}, {m02, m12}]

Out[186]= B00(pp, m02, m12)
```

This is $B_{11}(p_1^2, m_1, m_2)$, the coefficient function of $p_{1\mu}\, p_{1\nu}$ of $B_{\mu\nu}$.

```
In[187]:= PaVe[1,1, {p10}, {m12, m22}]

Out[187]= B11(p10, m12, m22)
```

PaVe with one 0 as first argument are scalar Passarino-Veltman integrals.

```
In[188]:= PaVe[0, {p10, p12, p20}, {m12, m22,
            m32}]

Out[188]= C0(p10, p12, p20, m12, m22, m32)
```

This is D_0 with 10 arguments.

```
In[189]:= PaVe[0, {p10, p12, p23, p30, p20, p13},
            {m12, m22, m32, m42}]

Out[189]= D0(p10, p12, p23, p30, p20, p13, m12, m22, m32, m42)
```

B1[*p10, m02, m12***]**	coefficient function $B_1(p_1^2, m_0^2, m_1^2)$
B00[*p10, m02, m12***]**	coefficient function $B_{00}(p_1^2, m_0^2, m_1^2)$
B11[*p10, m02, m12***]**	coefficient function $B_{11}(p_1^2, m_0^2, m_1^2)$

Two-point coefficient functions.

The two-point coefficient functions can be reduced to lower-order ones. For special arguments, also B_0 is expressed in terms of A_0, if the option **BReduce** is set. Setting the option **B0Unique** to **True** simplifies $B_0(m^2, 0, m^2) \to 2 + B_0(0, m^2, m^2)$ and $B_0(0, 0, m^2) \to 1 + B_0(0, m^2, m^2)$.

option name	*default value*	
A0ToB0	**False**	express $A_0(m^2)$ by $m^2(1 + B_0(0, m^2, m^2))$
BReduce	**True**, **False** for **B0**	reduce **B0**, **B1**, **B00**, **B11** to **A0** and **B0**
B0Unique	**False**	simplify $B_0(a, 0, a)$ and $B_0(0, 0, a)$

Reduction options for **A0** and the two-point functions.

The default is to reduce B_1 to B_0.

```
In[190]:= B1[pp, m12, m22]
```

$$\text{Out[190]= } -\frac{B_0(\text{pp},\ \text{m12},\ \text{m22})}{2} + \frac{(-\text{m12} + \text{m22})\,(-B_0(0,\ \text{m12},\ \text{m22}) + B_0(\text{pp},\ \text{m12},\ \text{m22}))}{2\ \text{pp}}$$

Arguments of two-point functions with head **SmallVariable** are replaced by 0, if the other arguments have no head **SmallVariable** and are nonzero. **A0[0]** and **A0[SmallVariable[m]^2]** simplify to 0.

The small mass **m** is set to 0, since the other arguments are non-zero and not **SmallVariable**.

```
In[191]:= B1[pp, SmallVariable[me2], m22]
```

$$\text{Out[191]= } -\frac{B_0(\text{pp},\ 0,\ \text{m22})}{2} + \frac{\text{m22}\,(-B_0(0,\ 0,\ \text{m22}) + B_0(\text{pp},\ 0,\ \text{m22}))}{2\ \text{pp}}$$

But in this case no arguments are replaced by 0. **SmallVariable** heads are not displayed int **TraditionalForm**

```
In[192]:= B1[SmallVariable[me2],
          SmallVariable(me2), 0]
```

$$\text{Out[192]= } -\frac{1}{2} - \frac{\text{B0(me2,\ 0,\ me2)}}{2}$$

In **StandardForm** we see them.

```
In[193]:= % // StandardForm
```

$$\text{Out[193]= } -\frac{1}{2} - \frac{\text{B0[SmallVariable[me2],\ 0,\ SmallVariable[me2]]}}{2}$$

Any mass with head **SmallVariable** is neglected if it appears in a sum, but not as an argument of Passarino-Veltman (**PaVe**) functions or **PropagatorDenominator**.

SmallVariable[*var***]**	is a small (negligible) variable

Head for small variables.

PaVeReduce[*expr***]**	reduces coefficient functions **PaVe** to **A0**, **B0**, **C0**, **D0**
KK[*i***]**	abbreviations in **HoldForm** in the result of **PaVeReduce**

Reduction function for Passarino-Veltman coefficient functions.

Depending on the option **BReduce** **B1**, **B00** and **B11** may also remain in the result of **PaVeReduce**.

option name	*default value*	
IsolateNames	**False**	use **Isolate** with this option
Mandelstam	**{}**	Mandelstam relation, e.g., **{s, t, u, 2 mw^2}**

Options for **PaVeReduce**.

The function **Isolate** is explained in section 6.3.

Reduce $C_2(m_e^2, m_W^2, t, m_e^2, 0, m_W^2)$ to scalar integrals.

```
In[194]:= PaVeReduce[ PaVe[2, {SmallVariable[me2],
            mw2, t},{SmallVariable[me2], 0, mw2} ]]
```

$$\text{Out[194]}= \frac{B_0(\text{mw2},\ 0,\ \text{mw2})}{\text{mw2} - t} - \frac{B_0(t,\ 0,\ \text{mw2})}{\text{mw2} - t}$$

Break down the coefficient function $C_{12}(s, m^2, m^2, m^2, m^2, w^2)$. This is the result in **HoldForm**.

```
In[195]:= c12 = PaVeReduce[ PaVe[1, 2, {s, m2,
           m2}, {m2, m2, w2}],
           IsolateNames → c ]
```

$$Out[195]= c(11)$$

The **FullForm** of the assignment to **c12** is **HoldForm[c[11]]**. If you want to get the value of **c[11]**, you can either type **ReleaseHold[c12]** or **c[11]** as done in the example.

```
In[196]:= c[11]
```

$$Out[196]= \frac{2\,\mathrm{m2}\,c(3)}{s\,c(6)} + \frac{c(1)\,c(7)}{s\,c(6)^2} + \frac{c(2)\,c(8)}{2\,c(6)^2} + \frac{c(4)\,c(9)}{s\,c(6)} + \frac{\mathrm{w2}\,c(5)\,c(10)}{c(6)^2} + \frac{1}{2\,c(6)}$$

Have B_1's be reduced to B_0's.

```
In[197]:= SetOptions[B1, BReduce → True];
```

Repeated application of **ReleaseHold** reinserts all **K**.

```
In[198]:= Simplify /@ Collect[FixedPoint[ReleaseHold,
           c12], _B0, _C0]
```

$$Out[198]= \frac{1}{2\,(4\,\mathrm{m2} - s)} + \frac{(\mathrm{m2} - \mathrm{w2})\,B_0(0,\ \mathrm{m2},\ \mathrm{w2})}{2\,\mathrm{m2}\,(4\,\mathrm{m2} - s)} - \frac{(8\,\mathrm{m2}^2 - 10\,\mathrm{m2}\,\mathrm{w2} - 2\,\mathrm{m2}\,s + \mathrm{w2}\,s)\,B_0(\mathrm{m2},\ \mathrm{m2},\ \mathrm{w2})}{2\,\mathrm{m2}\,(4\,\mathrm{m2} - s)^2} + \frac{(4\,\mathrm{m2} - 6\,\mathrm{w2} - s)\,B_0(s,\ \mathrm{m2},\ \mathrm{m2})}{2\,(4\,\mathrm{m2} - s)^2} + \frac{\mathrm{w2}\,(8\,\mathrm{m2} - 3\,\mathrm{w2} - 2\,s)\,C_0(\mathrm{m2},\ \mathrm{m2},\ s,\ \mathrm{m2},\ \mathrm{w2},\ \mathrm{m2})}{(4\,\mathrm{m2} - s)^2}$$

Take a coefficient function from $D_{\mu\nu\rho}$: $D_{122}(m_e^2, m_w^2, m_w^2, m_e^2, s, t, 0, m_e^2, 0, m_e^2)$.

```
In[199]:= d122 = PaVeReduce[
           PaVe[1, 2, 2, {SmallVariable[me2], mw2,
           mw2, SmallVariable[me2], s, t},
           {0, SmallVariable[me2], 0,
           SmallVariable[me2]}],
           Mandelstam → {s, t, u, 2 mw2},
           IsolateNames → f ]
```

$$Out[199]= f[16]$$

Write the result out into a Fortran file.

```
In[200]:= Write2[ "d122.for", d122res = d122,
          FormatType → FortranForm ];
```

This shows the resulting Fortran file.
The first abbreviations are always the scalar integrals.
The partially recursive definitions of the abbreviations are not fully optimized.
The function **Write2** is explained in section 6.
Note that the head **SmallVariable** is eliminated in the Fortran output automatically.

```
In[201]:= !!d122.for

      f(1)= B0(mw2,me2,0D0)
      f(2)= B0(s,0D0,0D0)
      f(3)= B0(t,me2,me2)
      f(4)= C0(mw2,mw2,t,me2,0D0,me2)
      f(5)= C0(mw2,s,me2,me2,0D0,0D0)
      f(6)= C0(t,me2,me2,me2,me2,0D0)
      f(7)= D0(mw2,mw2,me2,me2,t,s,me2,0D0,me2,0D0)
      f(8)= mw2 + s
      f(9)= 4*mw2 - t
      f(10)= mw2**2 - s*u
      f(11)= mw2 - s
      f(12)= 4*mw2**5-5*mw2**4*s-16*mw2**3*s**2+
     & 4*mw2**2*s**3 + 4*mw2*s**4 - mw2**4*u -
     & 4*mw2**2*s**2*u+8*mw2*s**3*u+4*mw2**2*s*u**2+
     & s**3*u**2 + s**2*u**3
      f(13)= mw2**2 - 4*mw2*s + 2*s**2 + s*u
      f(14)= 4*mw2**3-9*mw2**2*s+2*s**3-mw2**2*u-
     & 4*mw2*s*u + 5*s**2*u + 3*s*u**2
      f(15)= 2*mw2**6-8*mw2**5*s+12*mw2**4*s**2-
     & 8*mw2**3*s**3+2*mw2**2*s**4-2*mw2**5*t+
     & 20*mw2**4*s*t-36*mw2**3*s**2*t+20*mw2**2*s**3*t-
     & 2*mw2*s**4*t-6*mw2**3*s*t**2+6*mw2**2*s**2*t**2-
     & 6*mw2*s**3*t**2+4*mw2*s**2*t**3-s**2*t**4
      f(16)= -f(8)/(2D0*f(9)*f(10)) -
     & (s**2*t**2*f(6)*f(11))/(2D0*f(10)**3) +
     & (s**3*t**2*f(7)*f(11))/(2D0*f(10)**3) +
     & (s**2*t*f(5)*f(11)**2)/f(10)**3 -
     & (f(1)*f(12))/(2D0*f(9)**2*f(10)**2*f(11)) +
     & (s*f(2)*f(13))/(2D0*f(10)**2*f(11)) +
     & (f(3)*f(8)*f(14))/(2D0*f(9)**2*f(10)**2) -
     & (f(4)*f(8)*f(15))/(2D0*f(9)**2*f(10)**3)
      d122res = f(16)
```

Delete the output file.

```
In[202]:= DeleteFile["d122.for"];
```

The Fortran code generated by **Write2** should be checked with care. All integer numbers (except 0 as argument of **B0**, **C0**, **D0**) are translated to integers. This causes problems when translating variables with rational powers and must be corrected in the Fortran output by hand.

5.2 Tables of Integrals

CheckDB[*expr*, *fil***]**	saves or retrieves *expr* to/from a file *fil*

Storing and retrieving results.

option name	*default value*	
Directory	**"fcdb"**	directory to use
NoSave	**False**	don't save to disk
ForceSave	**False**	force the expression to be evaluated even if the file exists. The expression is also saved if **NoSave** is set to **False**
Check	**False**	check if the evaluation of the expression agrees with what is loaded from disk and return **True** or **False**

Options of **CheckDB**.

6 Miscellaneous Functions

6.1 Low Level Dirac Algebra Functions

There are a number of lower level functions for doing Dirac algebra. Some are used by the functions described in section 4.2, some may be useful to perform more controlled calculations. For completeness, they are described here.

DiracGammaCombine[*expr***]**	combines sums, **DiracGamma[Momentum[***a***]] + DiracGamma[Momentum[***b***]] + ...**, in *expr* into **DiracGamma[Momentum[***a* **+** *b* **+ ..]]**
DiracGammaExpand[*expr***]**	expands all **DiracGamma[Momentum[***a* **+** *b* **+ ..]]** in *expr* into **DiracGamma[Momentum[***a***]] + DiracGamma[Momentum[***b***]] + ...**
DiracGammaT[*x***]**	the transpose of **DiracGamma[***x***]**
DiracSigma[*a***,** *b***]**	*i*/2 (*a b* -*b a*) in 4 dimensions. *a* and *b* must have head **DiracGamma**, **DiracMatrix** or **DiracSlash**
DiracSigmaExplicit[*expr***]**	inserts in *expr* the definition of **DiracSigma**
DiracSpinor[*p***,** *m***,** *ind***]**	a Dirac spinor for a fermion with momentum *p* and mass *m* and indices *ind*
EpsChisholm[*expr***]**	substitutes for a gamma matrix contracted with a Levi-Civita tensor (**Eps**) the Chisholm identity
ChisholmSpinor[*expr***]**	uses the Chisholm identity on a Dirac gamma matrix between spinors
ChisholmSpinor[*expr***,** *i***]**	uses the Chisholm identity on the first or second part of a spinor chain, depending on whether *i* is 1 or 2

Low level Dirac algebra functions.

6.2 Functions for Polynomial Manipulations

Unfortunately, at least with Mathematica 2.0, the built-in functions **Factor**, **Collect** and **Together** are either not general enough for the purposes needed in FeynCalc or consume too much CPU time for certain polynomials

with rational coefficients. The FeynCalc extensions **Factor2**, **Collect2** and **Combine** should be used instead when manipulating the rational polynomials emerging from **OneLoop** or **PaVeReduce**.

Collect2[*expr*, *x*$_1$, *x*$_2$, ...**]**	group together powers of terms containing $x_1, x_2, ...$
Collect2[*expr*, *x***]**	group together powers of terms containing *x*
Combine[*expr***]**	put terms in a sum over a common denominator
Factor2[*expr***]**	factor a polynomial in a canonical way

Modifications of **Collect**, **Together** and **Factor**.

This collects **f[x]** and **p[y]**.

```
In[203]:= Collect2[ (b - 2) d f[x] + f[x] + c p[y]
          + p[y], {f, y}]
```

$$\text{Out[203]} = (bd - 2d + 1) f(x) + (c + 1) p(y)$$

This puts terms over a common denominator without expanding the numerator.

```
In[204]:= Combine[(a - b) (c - d)/e + g]
```

$$\text{Out[204]} = \frac{(a - b)(c - d) + e\,g}{e}$$

Consider a generic polynomial.

```
In[205]:= test = (a - b) x + (b - a) y
```

$$\text{Out[205]} = (a - b)\,x + (-a + b)\,y$$

Under Mathematica 2.0, mapping **Factor** on the summands shows that no canonical factorization of integers in sums takes place.

```
In[206]:= Map[Factor, test]
```

$$\text{Out[206]} = (a - b)\,x + (-a + b)\,y$$

If a canonical factorization is desired, you may use **Factor2** instead.

```
In[207]:= Map[Factor2, test]
```

$$\text{Out[207]} = (a - b)\,x - (a - b)\,y$$

This is the overall factorization of **Factor**.

```
In[208]:= Factor[test]
Out[208]= (-a + b) (-x + y)
```

Here the convention used by **Factor2** becomes clear: The first term in a subsum gets a positive prefactor.

```
In[209]:= Factor2[test]
Out[209]= (a - b) (x - y)
```

option name	*default value*	
ProductExpand	**False**	expand products

An expansion controlling option for **Collect2** and **Combine**.

6.3 An Isolating Function for Automatically Introducing Abbreviations

Isolate[*expr***]**	if **Length[***expr***]** > 0, substitute an abbreviation **KK[***i***]** for *expr*
Isolate[*expr***, {***x*$_1$**, ***x*$_2$**, ...}]**	substitute abbreviations for subsums free of $x_1, x_2, ...$

A function for isolating variables by introducing abbreviations for common subexpressions.

option name	*default value*	
IsolateNames	**KK**	head of the abbreviations
IsolateSplit	**442**	a limit beyond which **Isolate** splits the expression into two sums

Options for **Isolate** and **Collect2**.

Isolate with one argument introduces a single **K[*i*]** as abbreviation.

```
In[210]:= Isolate[a + b]
```

```
Out[210]= KK(1)
```

Here f gets isolated with **KK[1]** and **KK[2]** replacing the bracketed subsums.

```
In[211]:= test = Isolate[(a + b) f + (c + d) f +
          e, f]
```

```
Out[211]= e + f KK(1) + f KK(2)
```

Looking at the **FullForm** reveals that the **KK[*i*]** are given in **HoldForm**.

```
In[212]:= FullForm[test]
```

```
Out[212]= Plus[e, Times[f, HoldForm[KK[1]]], Times[f, Hold-
          Form[KK[2]]]]
```

Asking for **KK[1]** returns its value, but in **test KK[1]** is held.

```
In[213]:= { KK[1], test, ReleaseHold[test] }
```

```
Out[213]= {a + b, e + f KK(1) + f KK(2) e + (a + b) f + (c + d) f}
```

For the term **(b + c (y + z))** a single abbreviation **g[2]** is returned.

```
In[214]:= Isolate[a[z] (b + c (y + z)) + d[z] (y +
          z), {a, d},
          IsolateNames → g]
```

```
Out[214]= d(z) g(1) + a(z) g(2)
```

With the help of an additional function it is easy to introduce identical abbreviations for each subsum.
This trick of selectively substituting functions for sums is also quite useful in special reordering of polynomials.

```
In[215]:= und[x__] := Isolate[Plus[x],
          IsolateNames → h]/;
          FreeQ2[{x}, {a, d}];
          (a[z] (b + c (y + z)) + d[z] (y + z))/.
          Plus → und /.  und → Plus
```

```
Out[215]= d(z) h(1) + a(z) h(2)
```

Here it is clear that **h[1]** = (y + z) is part of **h[2]**.

```
In[216]:= ReleaseHold[%]

Out[216]= (y + z) d(z) + a(z) (b + c h(1))
```

Decreasing the option **IsolateSplit** significantly forces **Isolate** to use more than one **l** for **a - b - c - d - e**.

```
In[217]:= Isolate[ a - b - c - d - e, IsolateNames
            → l, IsolateSplit → 15 ]

Out[217]= l(2)
```

This shows the values of the **l**.

```
In[218]:= {l[2], l[1]}

Out[218]= {-c - d - e + l(1), a - b}
```

The importance of **Isolate** is significant, since it gives you a means to handle very big expressions. The use of **Isolate** on big polynomials before writing them to a file is especially recommended.

A subtle issue of the option **IsolateSplit**, whose setting refers to the "size" of an expression as returned by **Length[Characters[ToString[FortranForm[***expr***]]]]**, is that it inhibits too many continuation lines in the Fortran file when writing out expressions involving **KK[i]** with **Write2**. See Section 6.5 for the usage of **Write2**.

You may want to change **IsolateSplit** when working with big rational polynomials in order to optimize the successive (**FixedPoint**) application of functions like **Combine[ReleaseHold[#]]&**.

6.4 An Extension of FreeQ and Two Other Useful Functions

FreeQ2[*expr***, {***f_1***, ***f_2***, ...}]**	yields **True** if *expr* does not contain any occurrence of f_1, f_2, ...
NumericalFactor[*expr***]**	gives the numerical factor of *expr*
PartitHead[*expr***, *h*]**	returns a list **{***a***, *h***[b]}** with *a* free of expressions with head *h*

Three useful functions.

FreeQ2 is an extension of **FreeQ**, allowing a list as second argument.

```
In[219]:= FreeQ2[w^2 + m^2 B0[pp, m1, m2], {w,
            B0}]

Out[219]= False
```

This gives the numerical factor of the expression.

```
In[220]:= NumericalFactor[-137 x]

Out[220]= -137
```

The action of **PartitHead** on a product is to split it apart.

```
In[221]:= PartitHead[f[m] (s - u), f]

Out[221]= {s - u, f(m)}
```

A sum gets separated into subsums.

```
In[222]:= PartitHead[s^2 + m^2 - f[m], f]

Out[222]= {m^2 + s^2, -f(m)}
```

6.5 Writing Out to Mathematica and Fortran

The Mathematica functions **Write** and **Save** may on rare occasions create files that cannot be read in again correctly. What sometimes happens is that, for example, a division operator is written out as the first character of a line. When the file is loaded again, Mathematica may simply ignore that part of the file before this character. You can easily work around this bug by editing the file and wrapping the expressions with brackets "( )". Since this is a cumbersome procedure for lots of expressions, it has been automatized in **Write2** for writing out expressions. If you use **Save**, you should check the output file for correctness.

An option can be given to **Write2** allowing the output to be written in **FortranForm**. This facility is elementary and mostly limited to the type of polynomials usually encountered in FeynCalc.

At least with Mathematica 2.0, notice: The **FortranForm** option should not be used if the expression to be written out contains a term $(-x)^n$, where n is a symbol, (such terms are never produced by FeynCalc, unless they already contained in the input). The Mathematica**FortranForm** is also incapable of recognizing some elementary functions like **Exp[x]**.

Write2[*fi***, *x* = *y*]**	write the setting $x = y$ into a file *fi*
Write2[*fi***, *x* = *y*, *a* = *b*, ...]**	write the setting $x = y$ into a file *fi*

A modification of **Write**.

option name	*default value*	
FormatType	**InputForm**	language in which to write the output
D0Convention	**0**	which convention to use for the scalar integrals

Options for **Write2**.

With the default setting **0** of **D0Convention** the arguments of the scalar integrals are not changed when written into a Fortran program. Another possible setting is **1**, which interchanges the fifth and sixth arguments of **D0** and writes the mass arguments of **A0**, **B0**, **C0** and **D0** (without squares).

InputForm	write out in Mathematica form
FortranForm	write out in Fortran form

The four possible settings of the option **FormatType** for **Write2**.

Create a test polynomial.

```
In[223]:= tpol = z + Isolate[Isolate[2 Pi I + f[x]
          (a - b), f]]

Out[223]= z + KK(4)
```

The default is to write out in Mathematica **InputForm**.

```
In[224]:= Write2["test.m", test = tpol];
```

Show the content of the file.

```
In[225]:= !!test.m

Out[225]= KK[3] = (a - b
          );
          KK[4] = ((2*I)*Pi + f[x]*HoldForm[KK[3]]
          );
          test = z + HoldForm[KK[4]]
```

Clean up.

```
In[226]:= DeleteFile["test.m"];
```

This writes a file in Fortran format.

```
In[227]:= Write2["test.for", test = tpol,
          FormatType → FortranForm];
```

Show the content of the Fortran file.

```
In[228]:= !!test.for

Out[228]= KK(3) = a - b
          KK(4) = (0,2)*Pi + f(x)*KK(3)
          test = z + KK(4)
```

Clean up.

```
In[229]:= DeleteFile /@ "test.m", "test.for";
```

6.6 More on Levi-Civita Tensors

EpsEvaluate[*expr***]** evaluate Levi-Civita **Eps**

A function for total antisymmetry and linearity with respect to **Momentum** .

This is $\varepsilon^{\mu\nu\rho\sigma}(p+q)_\sigma = \varepsilon^{\mu\nu\rho(p+q)}$.

```
In[230]:= Contract[LeviCivita[μ, ν, ρ, σ]
          FourVector[p + q, σ], EpsContract →
          False]

Out[230]= ϵ^{μ,ν,ρ,p+q}
```

In this way linearity is enforced: $\varepsilon^{\mu\nu\rho(p+q)} = \varepsilon^{\mu\nu\rho p} + \varepsilon^{\mu\nu\rho q}$.

```
In[231]:= EpsEvaluate[%]

Out[231]= ϵ^{μ,ν,ρ,p} + ϵ^{μ,ν,ρ,q}
```

EpsChisholm[*expr***]** utilize $\gamma_\mu \varepsilon^{\mu\nu\rho\sigma} = +i(\gamma^\nu\gamma^\rho\gamma^\sigma - g^{\nu\rho}\gamma^\sigma - g^{\rho\sigma}\gamma^\nu + g^{\nu\sigma}\gamma^\rho)\gamma^5$

A function for the Chisholm identity.

option name	*default value*	
LeviCivitaSign	-1	sign convention for the ε-tensor

An option for **EpsChisholm**.

Use the Chisholm identity for $\varepsilon^{abcd}\,\gamma^{\mu}\,\gamma^{a}$.

```
In[232]:= EpsChisholm[ LeviCivita[α, β, γ, δ]
          DiracMatrix[μ, α] // FCI]
Out[232]= iγ^μ.γ^β.γ^γ.γ^δ.γ^5 - iγ^μ.γ^δ.γ^5.g^βγ + iγ^μ.γ^γ.γ^5.g^βδ - iγ^μ.γ^β.γ^5.g^γδ
```

Eps[*a***, *b*, *c*, *d*]**	internal representation of Levi-Civita tensors, where the arguments must have head **LorentzIndex** or **Momentum**

The internal function for Levi-Civita tensors, see also section 3.

This shows the internal structure.

```
In[233]:= LeviCivita[μ, ν, ρ, σ] // StandardForm
Out[233]= Eps[LorentzIndex[μ], LorentzIndex[ν],LorentzIndex[ρ],
          LorentzIndex[σ]]
```

Contracting $\varepsilon^{\mu\nu\rho\sigma}\,p^{\sigma}$ yields $\varepsilon^{\mu\nu\rho p}$.
In the internal representation the p has the head **Momentum** wrapped around it.

```
In[234]:= Contract[% FourVector[p, s]] //
          StandardForm
Out[234]= Eps[LorentzIndex[μ], LorentzIndex[ν],LorentzIndex[ρ],
          Momentum[p]]
```

6.7 Simplifications of Expressions with Mandelstam Variables

TrickMandelstam[*expr*, {*s*, *t*, *u*, *m1*^2 + *m2*^2 + *m3*^2 + *m4*^2}]	simplifies *expr* to the shortest form removing *s*, *t* or *u* in each sum using $s + t + u = m_1^2 + m_2^2 + m_3^2 + m_4^2$

For tricky Mandelstam variable substitution.

This is an easy example of simplifications done by **TrickMandelstam**.

```
In[235]:= TrickMandelstam[(s + t - u) (2 mw2 - t -
            u), {s, t, u, 2 mw2}]
```

$$\text{Out[235]} = 2\,s\,(\text{mw2} - u)$$

The result is always given in a factorized form.

```
In[236]:= TrickMandelstam[m^2 s - s^2 + m^2 t - s t
            + m^2 u - s u, {s, t, u, 2 m^2}]
```

$$\text{Out[236]} = 2\,m^2(m^2 - s)$$

6.8 Manipulation of Propagators

FeynAmpDenominatorCombine[*expr***]**
expands *expr* w.r.t. to **FeynAmpDenominator** and combines products of **FeynAmpDenominator** into one **FeynAmpDenominator**

FeynAmpDenominatorSimplify[*expr***]**
simplifies each PropagatorDenominator in a canonical way

FeynAmpDenominatorSimplify[*expr***, sl q]**
includes some translation of momenta

FeynAmpDenominatorSimplify[*expr***, sl q1, sl q2]**
additionally removes 2-loop integrals with no mass scale

FeynAmpDenominatorSplit[*expr***]**
splits every **FeynAmpDenominator[a,b, ...]** into **FeynAmpDenominator[a]*FeynAmpDenominator[b] ...**

FeynAmpDenominatorSplit[*expr***, sl q]**
splits every **FeynAmpDenominator[a,b, ...]** into a product of two, one with sl q and other momenta, the other without sl q

PropagatorDenominatorExplicit[*expr***]**
writes out **FeynAmpDenominator[a,b, ...]** explicitly as a fraction

Functions for manipulating propagators.

6.9 Polarization Sums

PolarizationSum[*mu, nu***]**	$-g^{\mu\nu}$
PolarizationSum[*mu, nu, k***]**	$-g^{\mu\nu} + k^\mu\, k^\nu/k^2$
PolarizationSum[*mu, nu, k, n***]**	$-g^{\mu\nu} - k_\mu k_\nu n^2/(k\cdot n)^2 + (n_\mu k_\nu + n_\nu k_\mu)/(k\cdot n)$

Polarization sums.

This is the polarization sum for massive bosons: $\Sigma\varepsilon_\mu\varepsilon_\nu^* = -g^{\mu\nu} + k^\mu\, k^\nu/k^2$.

```
In[237]:= PolarizationSum[μ, ν, k]
```

$$\text{Out[237]} = \frac{k^\mu\, k^\nu}{k^2} - g^{\mu\nu}$$

Here the polarization sum for gluons is given; with external momentum $n = p_1 - p_2$.

```
In[238]:= PolarizationSum[μ, ν, k, p1 - p2]
```

$$\text{Out[238]} = -g^{\mu\nu} - \frac{k^\mu k^\nu\, \text{p1}^2}{(k\cdot p1 - k\cdot \text{p2})^2} + \frac{2k^\mu k^\nu\, \text{p1p2}}{(k\cdot p1 - k\cdot \text{p2})^2} - \frac{k^\mu k^\nu\, \text{p2}^2}{(k\cdot p1 - k\cdot \text{p2})^2} + \frac{(\text{p1} - \text{p2})^\mu\, k^\nu}{k\text{p1} - k\text{p2}} + \frac{k^\mu\, (\text{p1} - \text{p2})^\nu}{k\text{p1} - k\text{p2}}$$

6.10 Permuting the Arguments of the Four-Point Function

The arguments of the Passarino-Veltman four-point function **D0** are permuted by FeynCalc into an alphabetical order. If you want a specific permutation of the 24 possible ones, you can use the function **PaVeOrder**.

PaVeOrder[*expr***]**	order the arguments of **C0** and **D0** in *expr* canonically.

An ordering function for C_0 and D_0.

option name	*default value*	
PaVeOrderList	**{}**	order D_0 according to **{ . . . , a, b, . . . }**

An ordering option for **PaVeOrder**.

With the default setting of **PaVeOrder** a canonical ordering is chosen.

It is sufficient to supply a subset of the arguments.

```
In[239]:= PaVeOrder[D0[me2, me2, mw2, mw2, t, s,
            me2, 0, me2, 0],
            PaVeOrderList → {me2, me2, 0, 0}]
```

Out[239]= D_0(me2, s, mw2, t, mw2, me2, me2, 0, 0, me2)

This interchanges f and e.

```
In[240]:= PaVeOrder[D0[a, b, c, d, e, f, m12, m22,
            m32, m42],
            PaVeOrderList → {f, e}]
```

Out[240]= D_0(a, d, c, b, f, e, m22, m12, m42, m32)

This shows how to permute several D_0.

```
In[241]:= PaVeOrder[D0[a, b, c, d, e, f, m12, m22,
            m32, m42] +  D0[me2, me2, mw2, mw2, t,
            s, me2, 0, me2, 0], PaVeOrderList → {
            {me2, me2, 0, 0}, {f, e}}]
```

Out[241]= D_0(a, d, c, b, f, e, m22, m12, m42, m32) + D_0(me2, s, mw2, t, mw2, me2, me2, 0, 0, me2)

Chapter 2

Physics Applications

1 The Standard Model.

1.1 A Photon One-Loop Self Energy Diagram

The function **OneLoop** performs the algebraic simplifications of a given amplitude. The result is given in a polynomial of standard matrix elements, invariants of the process under consideration, and Passarino-Veltman integrals.

OneLoop[*q*, *amp*]	calculates the one-loop amplitude *amp* with *q* as loop momentum
OneLoop[*name*, *q*, *amp*]	calculates the one-loop amplitude *amp* and gives it a name

Calculating one-loop amplitudes.

`PropagatorDenominator[ Momentum[q], m]`	a factor of the denominator of a propagator
`PropagatorDenominator[ Momentum[q, D], m]`	a factor of the denominator of a propagator in D dimensions
`FeynAmpDenominator[ PropagatorDenominator[ ... ], PropagatorDenominator[ ... ]]`	a propagator

Representation of integrands.

The first argument to **OneLoop** is optional. It indicates a name for the amplitude for bookkeeping reasons. The second argument q is the loop momentum, i.e., the integration variable.

As last argument the analytical expression for the graph is given. It may be given in four dimensions. **OneLoop** performs the necessary extension to D dimensions automatically.

This is $A_0 = -i\,\pi^{-2}\,(2\,\pi\mu)^{4-D}\int d^D q\,(q^2-m^2)^{-1}$, corresponding to a tadpole diagram. The scaling variable μ is suppressed by FeynCalc.

```
In[242]:= -I/Pi^2 FeynAmpDenominator[
          PropagatorDenominator[q, m]]
```

$$\text{Out[242]} = -\frac{i}{\pi^2(q^2 - m^2)}$$

This calculates the tadpole diagram.

```
In[243]:= a0 = OneLoop[q, a0]
```

$$\text{Out[243]} = A_0(m^2)$$

Have **A0** written in terms of **B0**.

```
In[244]:= SetOptions[A0, A0ToB0 → True];
```

This calculates the tadpole diagram again. Now the result is given in terms of **B0**.

```
In[245]:= OneLoop[q, a0]
```

$$\text{Out[245]} = m^2 B_0(0,\ m^2,\ m^2) + m^2$$

Return to the default.

```
In[246]:= SetOptions[A0, A0ToB0 → False];
```

To obtain a more compact result, the factoring option of **OneLoop** is set. For a description of all options of **OneLoop** see section 1.3.

```
In[247]:= SetOptions[OneLoop, Factoring → True];
```

This is the transversal part of a photon self energy diagram with a fermion loop.
$ie^2/((2\pi)^4\,(1-D))$
$\int d^4q\,[q^2-m_f^2]^{-1}[(q-k)^2-m_f^2]^{-1}$
$\mathrm{tr}[\,(m_f+\not{q}-\not{k})\,\gamma^\nu\,(\not{q}+m_f)\,\gamma^\nu\,]$
$=$
$-[e^2(k^2+6m_f^2B_0(0,m_f^2,m_f^2)-$
$-3(k^2+$
$2m_f^2)B_0(k^2,m_f^2,m_f^2))]/(36\pi^2)$

```
In[248]:= OneLoop[ q, (I el^2)/(16 Pi^4)/(1 - D) *
          FeynAmpDenominator[
          PropagatorDenominator[q, mf],
          PropagatorDenominator[q - k, mf] ] *
          DiracTrace[(mf + DiracSlash[q - k]) .
          DiracMatrix[mu] .
          (mf + DiracSlash[q]) . DiracMatrix[mu]]
          ] /. ScalarProduct[k, k] → k2 /.
          (mf^2) → mf2
```

$$\text{Out[248]} = \frac{\text{el}^2\ (-\text{k2} + 6\,\text{mf2} - 6\,A_0(\text{mf2}) + 3\,(\text{k2} + 2\,\text{mf2})\,B_0(\text{k2},\ \text{mf2},\ \text{mf2}))}{36\,\pi^2}$$

Note that in this example, where the dimension is entered explicitly as a parameter (D), the option **Dimension** of **OneLoop** must also be set to D (this is the default).

1.2 Generic Diagrams for W → $f_i\bar{f}'_j$ with OneLoop

As an example for calculating triangle diagrams the result for two generic one-loop diagrams of the decay W → $f_i\bar{f}'_j$ for massless fermions given in [12] is verified with FeynCalc.

For the two diagrams different approaches are taken. In the first one FeynCalc introduces standard matrix elements for the part of the diagram containing polarization dependencies. In the other approach the set of standard matrix elements is defined by the user before FeynCalc calculates the diagrams. The last possibility is usually preferable, since the choices of FeynCalc for the standard matrix elements may have physical significance only by accident.

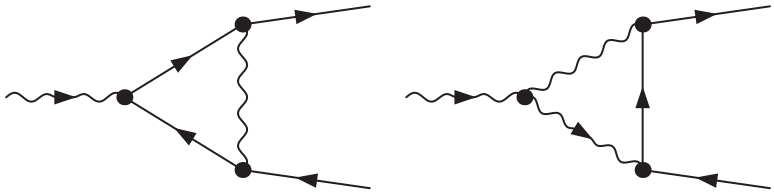


Figure 2.1: Two generic diagrams for the decay of $W \to f_i \bar{f}'_j$, generated by FeynArts.

This defines a function for abbreviation purposes: $g_i^c = g_i^- \omega_- + g_i^+ \omega_+$. Set the p_i on-shell and $(p_1 \cdot p_2) = k^2/2$, where k denotes the momentum of the W.

```
In[249]:= gc[i_] := g[i, "-"] DiracMatrix[7] +
          g[i, "+"] DiracMatrix[6];
          ScalarProduct[p1, p1] = 0;
          ScalarProduct[p2, p2] = 0;
          ScalarProduct[p1, p2] = k2/2;
          MakeBoxes[g[i_, j_], TraditionalForm] :=
          SubsuperscriptBox[g, i, j];
```

The analytical expression for the generic diagram is:

$$\delta m_1 = i(2\pi)^{-4} \int (2\pi\mu)^{4-D}\, d^D q$$
$$[(q^2 - m^2)\,(q + p_1)^2\,(q - p_2)^2]^{-1}$$
$$\overline{u}(p_1)\,\gamma^\nu\,(g_1^-\omega_- + g_1^+\omega_+)\,(\not{q} + \not{p}_1)$$
$$\not{\varepsilon}(g_3^-\omega_- + g_3^+\omega_+)\,(\not{q} -$$
$$\not{p}_2)\,\gamma^\nu\,(g_2^-\omega_- + g_2^+\omega_+)\,v(p_2)$$

Translated into the usual notation the result reads:

$$(1 - 2B_0(k^2, 0, 0) +$$
$$2k^2 C_0(0, 0, k^2, 0, m^2, 0) +$$
$$2k^2\, C_1(k^2, 0, 0, 0, 0, m^2) +$$
$$4C_{00}(k^2, 0, 0, 0, 0, m^2)$$
$$(\, g_1^+\, g_2^+\, g_3^+\, \overline{u}(p_1)\not{\varepsilon}\,\omega_+\, v(p_2) +$$
$$g_1^-\, g_2^-\, g_3^-\overline{u}(p_1)\not{\varepsilon}\,\omega_-\, v(p_2)))/(16\pi^2)$$

The remaining Dirac structure is wrapped with the head **StandardMatrixElement**, which is displayed like ‖ ⋯ ‖.

```
In[250]:= wff1 = OneLoop[ q, I/(2 Pi)^4
          FeynAmpDenominator[
          PropagatorDenominator[q, m],
          PropagatorDenominator[q + p1],
          PropagatorDenominator[q - p2]] *
          Spinor[p1] . DiracMatrix[nu] . gc[1] .
          DiracSlash[q + p1] .
          DiracSlash[Polarization[k]] .
          gc[3] . DiracSlash[q - p2] .
          DiracMatrix[nu] . gc[2] .
          Spinor[p2]  ] /. (m^2) → m2
```

$$\text{Out[250]} = -\frac{1}{16\,\pi^2}\Big((4\,B_0(0, 0, \text{m2}) - 2\,B_0(\text{k2}, 0, 0) -$$
$$2\,\text{k2}\,C_0(0,\ 0, \text{k2},\ 0,\ \text{m2},\ 0) - 2\,\text{m2}\,C_0(0,\ 0,\ \text{k2},\ 0,\ \text{m2},\ 0) -$$
$$4\,\text{PaVe}(0, 0, \{0, \text{k2}, 0\}, \{\text{m2}, 0, 0\}) - 1)$$
$$\Big(g_1^+\, g_2^+\, g_3^+\, \|\varphi(\text{p1})\,.\,(\gamma\cdot\varepsilon(k))\,.\,\gamma^6\,.\,\varphi(\text{p2})\| +$$
$$g_1^-\, g_2^-\, g_3^-\, \|\varphi(\text{p1})\,.\,(\gamma\cdot\varepsilon(k))\,.\,\gamma^7\,.\,\varphi(\text{p2})\|\Big)\Big)$$

This reduces the result to scalar integrals.

```
In[251]:= wff1a = PaVeReduce[wff1] // Simplify
```

$$\text{Out[251]} = \frac{1}{16\,\text{k2}\,\pi^2}$$
$$\Big(\big(-2\,(2\,\text{k2} + \text{m2})\,B_0(0, 0, \text{m2}) + (3\,\text{k2} + 2\,\text{m2})\,B_0(\text{k2}, 0, 0) +$$
$$2\,\big(C_0(\text{k2}, 0, 0, 0, 0, \text{m2})\,(\text{k2} + \text{m2})^2 + \text{k2}\big)\big)$$
$$\Big(g(1, +)\,g(2, +)\,g(3, +)\,\|\varphi(\text{p1})\,.\,(\gamma\cdot\varepsilon(k))\,.\,\gamma^6\,.\,\varphi(\text{p2})\| +$$
$$g(1, -)\,g(2, -)\,g(3, -)\,\|\varphi(\text{p1})\,.\,(\gamma\cdot\varepsilon(k))\,.\,\gamma^7\,.\,\varphi(\text{p2})\|\Big)\Big)$$

With this command you can extract the standard matrix elements.

```
In[252]:= var = Select[Variables[wff1a],
          (Head[#]===StandardMatrixElement)&]
```

$$\text{Out[252]} = \{\|\varphi(\text{p1})\,.\,(\gamma\cdot\varepsilon(k))\,.\,\gamma^6\,.\,\varphi(\text{p2})\|,$$
$$\|\varphi(\text{p1})\,.\,(\gamma\cdot\varepsilon(k))\,.\,\gamma^7\,.\,\varphi(\text{p2})\|\}$$

Here the **StandardMatrixElement**s are set to some abbreviations.

```
In[253]:= Set @@ {var, {ma[1], ma[2]} }
```

In this way you can generate a Fortran file.
With replacements you can adapt the result to your other Fortran code.

```
In[254]:= Write2["wff1a.for", vert = wff1a /.
          g[i_, "+"] → gp[i] /.
          g[i_, "-"] → gm[i], FormatType →
          FortranForm];
```

Show the content of the file.

```
In[255]:= !!wff1a.for

      vert = (((3*k2 + 2*m2)*B0(k2,Null,Null) -
     & 2*(2*k2 + m2)*B0(Null,m2,Null) +
     & 2*(k2 + (k2 + m2)**2*
     & C0(k2,Null,Null,Null,Null,m2)))*
     & (gp(1)*gp(2)*gp(3)*ma(1)+gm(1)*gm(2)*gm(3)*ma(2)))
     & /(16D1*k2*Pi**2)
```

Clean up.

```
In[256]:= DeleteFile["wff1a.for"];
          Clear[gc, g, wff1, wff1a, ma];
```

StandardMatrixElement[*expr***]**	a standard matrix element
SetStandardMatrixElements[{{*sm1* → *abb1***}, {***sm2* → *abb2***}, ...}]**	set abbreviations for standard matrix elements
SetStandardMatrixElements[{{*sm1* → *abb1***}, {***sm2* → *abb2***}, ...}**, $k_2 \to p_1 + p_2 - k_1$**]**	set abbreviations for standard matrix elements by using energy momentum conservation

A head for identifying standard matrix elements; *sm1*, *sm2* are the standard matrix elements, *abb1*, *abb2* the abbreviations.

The function **SetStandardMatrixElements** introduces **StandardMatrixElement[***abb1***]** for *sm1*. The abbreviations *abb1*, *abb2*, ... may be numbers or strings.

For calculating the generic triangle diagram with a non-abelian gauge coupling the standard matrix elements are set ahead using **SetStandardMatrixElements**.

The option **ReduceToScalars** is set to **True**; this will produce directly a result in terms of B_0 and C_0. The other definitions are convenient abbreviations: **r** for the right-handed projection operator γ_6, **l** for the left-handed projection operator γ_7, short mnemonic functions like **mt**, **fv** and **feynden** stand for metric tensors, four-vectors and denominators of propagators.

```
In[257]:= r = DiracMatrix[6]; l = DiracMatrix[7];
          ScalarProduct[p1, p1] =
          ScalarProduct[p2, p2] = 0;
          ScalarProduct[p1, p2] = k2/2 ;
          SetOptions[ OneLoop,
          Factoring → True, FormatType →
          FortranForm,
          ReduceToScalars → True, WriteOut → True,
          FinalSubstitutions → {g[i_, "+"] →
          gp[i], g[i_, "-"] → gm[i],
          StandardMatrixElement →  ma} ];
          mt = MetricTensor; fv = FourVector;
          feynden[x:{_, _}..] :=
          FeynAmpDenominator @@
          Map[Apply[PropagatorDenominator, #]&,
          {x}];
```

This sets the standard matrix elements:
$\mathcal{M}_1^+ = \overline{u}(p_1)\not{\epsilon}_1\,\omega_+\,v(p_2)$
$\mathcal{M}_1^- = \overline{u}(p_1)\not{\epsilon}_1\,\omega_-\,v(p_2)$

```
In[258]:= SetStandardMatrixElements[
          { (  Spinor[p1] . DiracSlash[Polarization[k]].
          r . Spinor[p2] ) → {1},
          (  Spinor[p1] . DiracSlash[Polarization[k]].
          l . Spinor[p2] ) → {2}}];
```

Here is the second generic diagram.
Note that in the result **StandardMatrixElement** is replaced by **ma**, as specified in the option **FinalSubstitutions** of **OneLoop** above.

```
In[259]:= OneLoop[q, I/(2 Pi)^4 *
          feynden[{q, 0}, {q + p1, m1}, {q - p2,
          m2}] *
          Spinor[p1] . DiracMatrix[nu] .
          (g[1, "-"] l + g[1, "+"] r) .
          DiracSlash[-q] . DiracMatrix[ro] .
          (g[2, "-"] l + g[2, "+"] r) .
          Spinor[p2] *
          g3 ( mt[ro, mu] fv[p1 + 2 p2 - q, nu] -
          mt[mu, nu] fv[2 p1 + p2 + q, ro] +
          mt[nu, ro] fv[2 q + p1 - p2, mu] ) *
          PolarizationVector[k, mu]
          ] /. (m1^2) → m12 /. (m2^2) → m22
```

$$\text{Out[259]= } -\frac{1}{(16\,k2\,\pi^2)}$$

$$(g3\,((2\,k2 + m12)\,B_0(0,\ 0,\ m12) +$$

$$(2\,k2 + m22)\,B_0(0,\ 0,\ m22) -$$

$$(k2 + m12 + m22)\,B_0(k2,\ m12,\ m22) + 2\,(k2\,m12 +$$

$$m22\,m12 + k2\,m22)\,C_0(k2,\ 0,\ 0,\ m12,\ m22,\ 0))$$

$$(gp(1)\,gp)(2)\,ma(1) + gm(1)\,gm(2)\,ma(2)))$$

As specified above in the options for **OneLoop**, Fortran output has been written to a file.

```
In[260]:= !!"wff2.for"

Out[260]= wff2 = -(g3*(-((k2 + m1**2 +
          m2**2)*B0(k2,m1**2,m2**2)) +
          & (2*k2 + m1**2)*B0(Null,m1**2,Null) +
          & (2*k2 + m2**2)*B0(Null,m2**2,Null) +
          & 2*(k2*m1**2 + k2*m2**2 + m1**2*m2**2)*
          & C0(k2,Null,Null,m1**2,m2**2,Null))*
          & (gp(1)*gp(2)*ma(1) + gm(1)*gm(2)*ma(2)))/
          & (16D1*k2*Pi**2)
```

Clean up.

```
In[261]:= DeleteFile["wff2.for"];
          DeleteFile /@ FileNames["PaVe*"];
          Clear[r, l, mt, fv, feynden, wff2];
```

1.3 The Options of OneLoop

Several options of **OneLoop** have already been introduced in the previous section. Here the full list of available options is given. The example in section 1.5 shows the use of some options.

In the automatic calculation of one-loop amplitudes it does not matter in which order the arguments of **FeynAmpDenominator** are given. Therefore the default setting of **DenomatorOrder** is **True**. In case you want to verify a result obtained by hand calculation, you can set this option to **False**, which will preserve the order of the propagators as entered. If you want to include the dimension D explicitly in the input, as in the example in section 1.1, you have to set **Dimension → D**.

With the default setting of **Dimension** you can enter four-dimensional objects to **OneLoop**, which are automatically extended to D dimensions inside **OneLoop**. In case you want to calculate a finite amplitude, you can set **Dimension → 4**.

The **Factoring** option should be used only for relatively small problems, since it may be very time consuming to factor the result. Unless the result of **OneLoop** is very short, only the coefficients of **StandardMatrixElement** are factored.

FormatType takes **InputForm**, **FortranForm**, **MacsymaForm** or **MapleForm** as settings. If the option **WriteOut** is set to **True**, the result is written out into a file using **Write2** with the setting of **FormatType**.

Replacements are done with **InitialSubstitutions** and **FinalSubstitutions**. Especially energy momentum conservation should be included, e.g., **InitialSubstitutions → {k2 → - k1 + p1 + p3}**. Note that the rules listed in **FinalSubstitutions** are not applied as one list of rules, but sequentially in a loop.

For more tuning, the option **IntermediateSubstitutions** can be used. Usually this should not be necessary. These rules are applied somewhere in the middle of the calculation.

If **IsolateNames** is set to **c**, for example, the result will be given as a **c[i]** in **HoldForm**. See **Isolate** for more information. The setting of **Mandelstam** may be, e.g., **Mandelstam → {s, t, u, m1^2 + m2^2 + m3^2 + m4^2}**, where $s + t + u = m_1^2 + m_2^2 + m_3^2 + m_4^2$.

The option **ReduceToScalars** should not be set to **True** when calculating several complicated diagrams involving $D_{\mu\nu\rho}$ or $D_{\mu\nu\rho\sigma}$. Depending on the computer you are using it may nevertheless work, but it is usually better to use **OneLoopSum** with the appropriate options. Note that depending on the setting of the option **BReduce** also

two-point coefficient functions may remain in the result.

For processes with light external fermions it is best not to neglect the fermion masses everywhere, but to keep them in the arguments of the scalar Passarino-Veltman functions. This set of masses should be supplied as a list to the option **`SmallVariables`**, see section 1.5.

If **`WriteOut`** is set to **`True`**, the result is written to a file composed of the first argument of **`OneLoop`**, i.e., the *name*. In which language, i.e., Mathematica, Fortran, Macsyma or Maple the result is written, depends on the setting of the option **`FormatType`**. You may also set **`WriteOut`** to a string, which denotes the directory in which to the result (actually this string is simply prepended to the file names).

option name	*default value*	
Apart2	**True**	use **Apart2** to partial fraction denominators
CancelQP	**True**	cancel $q \cdot p$ and q^2
DenominatorOrder	**False**	order the arguments of **FeynAmpDenominator** canonically
Dimension	**D**	number of space-time dimensions
Factoring	**False**	factor the result
FinalSubstitutions	**{}**	substitutions done at the end of the calculation
FormatType	**InputForm**	how to write out the result file
InitialSubstitutions	**{}**	substitutions done at the beginning of the calculation, e.g. energy-momentum conservation
IntermediateSubstitutions	**{}**	substitutions done at an intermediate stage of the calculation
IsolateNames	**False**	use **Isolate** on the result
Mandelstam	**{}**	indicate the Mandelstam relation
OneLoopSimplify	**False**	use **OneLoopSimplify** at the beginning of the calculation
Prefactor	**1**	extra prefactor of the amplitude
ReduceGamma	**False**	insert for γ_6 and γ_7 their definitions
ReduceToScalars	**False**	reduce to B_0, C_0, D_0
SmallVariables	**{}**	a list of masses, which will get wrapped with the head **SmallVariable**
WriteOut	**False**	write out a result file carrying the name of the optional first argument of **OneLoop**
WriteOutPaVe	**False**	store **PaVe**s in files
Sum	**True**	compatibility with FeynArts; sum terms multiplied with FeynArts' **SumOver**

Options of **OneLoop**.

1.4 OneLoopSum and Its Options

To sum a list of amplitudes two different methods are provided by **OneLoopSum**. Either the provided list of amplitudes is calculated and subsequently summed, or the summation is done partially before the calculation. This can be specified with the option **CombineGraphs**.

OneLoopSum[{FeynAmp[*N1*, *q*, *amp1*], **FeynAmp[** *N2*, *q*, *amp2*], ...}]	calculate a list of amplitudes
OneLoopSum[*expr***]**	sum already calculated amplitudes

A function for summing one-loop amplitudes.

The input of **OneLoopSum** is adapted to the output of FeynArts[1] . After saving a list of Feynman diagrams (that is, unintegrated amplitudes) created with FeynArts using the function **CreateFeynAmp**, you can start a new Mathematica session [2], load FeynCalc, get the amplitudes, calculate the amplitude with **OneLoopSum** and finally save the result e.g. as a Fortran file.

Below follows a sequence of commands to execute in order to generate a set of amplitudes from the insertions on the "particles" level and save it in a format understood be FeynCalc.

CreateFeynAmp[*ins***] // PickLevel[Particles] // ToFA1Conventions »**
filename

After quitting the kernel and loading FeynCalc, the amplitudes can be loaded by

amps **=** « *filename*

To actually use the Fortran file, obviously the constants and functions used, like masses and scalar integrals, have to be defined. This is discussed in section 1.6.

Instead of supplying a list of not yet calculated amplitudes you can also give the sum of already calculated ones as argument (*expr*) to **OneLoopSum**.

The output of **OneLoopSum** is typically given as a short expression wrapped in **HoldForm**. In order to get the full expression, the function **FRH** can be applied.

FRH[*expr***]**	removes all **HoldForm** and **Hold** in *expr*

A utility function.

[1] Actually to version 1 of FeynArts, but later versions provide a function translating to version 1 syntax (**ToFA1Conventions**). The examples given here all use FeynArts version 3.

[2] It is not possible to load FeynCalc and FeynArts simultaneously into one Mathematica session because some functions of FeynArts and FeynCalc have the same name but are in different contexts (name spaces). A more sophisticated approach is provided by the optional subpackage PHI, which patches FeynArts slightly, allowing the two packages to be loaded simultaneously.

option name	*default value*	
CombineGraphs	**{}**	combine amplitudes
Dimension	**True**	number of space-time dimensions
ExtraVariables	**{}**	list of variables which are bracketed out in the result like **B0**, **C0**, **D0** and **PaVe**
FinalFunction	**Identity**	function applied to the result
FinalSubstitutions	**{}**	substitutions done at the end of the calculation
FormatType	**InputForm**	format used when saving results
InitialSubstitutions	**{}**	substitutions done at the beginning of the calculation
IntermediateSubstitutions	**{}**	substitutions done at an intermediate stage in the calculation
IsolateNames	**KK**	**Isolate** the result
Mandelstam	**{}**	use the Mandelstam relation
Prefactor	**1**	multiply the result by a pre-factor
ReduceToScalars	**True**	reduce to scalar integrals
SelectGraphs	**All**	which graphs to select
WriteOutPaVe	**False**	write out the reduced **PaVe**

Options of **OneLoopSum**.

With the default options **OneLoopSum** calculates each amplitude separately by substituting **OneLoop** for **FeynAmp**. Then each single **PaVe** is reduced to scalar integrals. The hard final part consists in the simplification of the rational coefficients of the scalar integrals. This may involve thousands of factorizations and can therefore take hours of CPU time. But the algebraic simplifications achieved by putting all coefficients of the scalar integrals over a common denominator and to factor them, possibly cancelling factors and reducing the singularity structure, may be very significant. These calculations may need quite a lot of RAM space, therefore the options of **OneLoopSum** allow you to split up the task of summing lots of diagrams.

First you can select a certain subclass of diagrams with the option **SelectGraphs**. You may set, e.g., **SelectGraphs** → **{1, 2, 5, 8}**, which selects the amplitudes at positions 1, 2, 5 and 8 of the argument list of **OneLoopSum**. The setting **SelectGraphs** → **{1, 2, 5, 8, {10, 40}}** also includes the range of all amplitudes from position 10 to 40.

With the option **CombineGraphs** a possibility is given to sum the graphs before calculation. This is especially useful for combining a graph with its crossed counterpart. In general it makes sense to combine all graphs with the same propagators before calculation, but for very big sums this may reduce the performance considerably. The possible settings for **CombineGraphs** are the same as for **SelectGraphs**. If you use the FeynArts syntax

for the first argument of **FeynAmp**, i.e. **GraphName[..., N1]**, the last arguments of **GraphName** for combined graphs are concatenated and a new **GraphName** for the summed amplitude is created.

By setting the option **WriteOutPaVe** you can save the result of the reduction of each **PaVe** to a file for later use. The names of the corresponding files are generated automatically. In case you use **OneLoopSum** several times it recognizes previously saved reductions and loads these results automatically. This may save a considerable amount of time. Instead of setting the option **WriteOutPaVe** to an empty string (which means that the files are written in the current directory), you can specify another directory (a string prepended to the file names).

Note that these options together with the possibility of using **OneLoopSum** on already calculated graphs gives you a lot freedom to split up the calculation, which may be necessary in order to avoid memory overflow. It can also be a good idea to set **$VeryVerbose** to 1 or 2 for monitoring the calculation.

The option **Prefactor** may be used to extract global factors. You can, e.g., do **SetOptions[OneLoop, Prefactor → (2 SW^2)**, which causes each single amplitude to be multiplied by $2s_W^2$, and **SetOptions[OneLoopSum, Prefactor → 1/(2 SW^2)**, which causes the result of **OneLoopSum** to have $1/(2s_W^2)$ as a global factor.

1.5 Box Graphs of $e^+e^- \to ZH$

In this section it is shown how to calculate a sum of amplitudes with **OneLoopSum**. The input consists of a one page program with process-dependent definitions. This program reads in a file with unmodified output of FeynArts for the amplitudes. The Fortran file produced is obtained without any interactive action.

The six standard matrix elements are:

$$\begin{aligned}
\mathcal{M}_0^1 &= \overline{v}(p_1)\not{\epsilon}\,\omega_+\,u(p_2)\\
\mathcal{M}_0^2 &= \overline{v}(p_1)\not{\epsilon}\,\omega_-\,u(p_2)\\
\mathcal{M}_1^1 &= \overline{v}(p_1)\not{k}_1\,\omega_+\,u(p_2)\,\epsilon\,p_1\\
\mathcal{M}_1^2 &= \overline{v}(p_1)\not{k}_1\,\omega_-\,u(p_2)\,\epsilon\,p_1\\
\mathcal{M}_2^1 &= \overline{v}(p_1)\not{k}_1\,\omega_+\,u(p_2)\,\epsilon\,p_2\\
\mathcal{M}_2^2 &= \overline{v}(p_1)\not{k}_1\,\omega_-\,u(p_2)\,\epsilon\,p_2
\end{aligned}$$

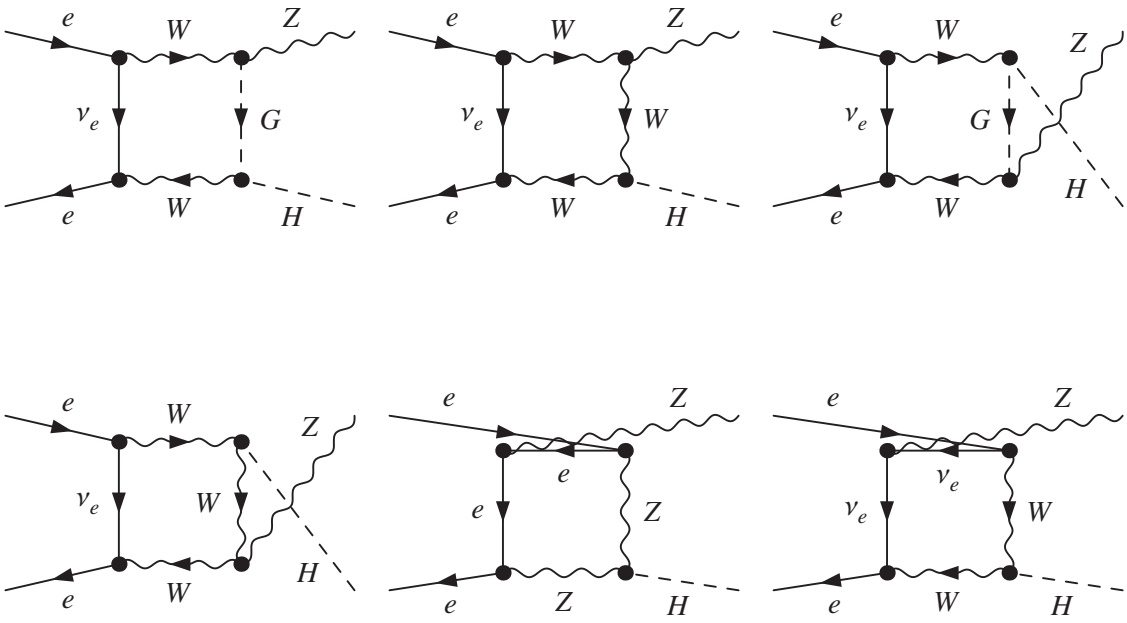


Figure 2.2: Box diagrams of $e^+e^- \to ZH$, generated by FeynArts. G denotes the unphysical charged Higgs.

```
(*Generate the box topologies*)
tops = CreateTopologies[1, 2 -> 2, Adjacencies -> {3},
      ExcludeTopologies -> {SelfEnergies, WFCorrections, Tadpoles,
          Boxes[3]}];

(*Check that the right topologies were generated*)
Paint[tops, ColumnsXRows -> {3, 1}];

(*Insert fields*)
inserttops =
    InsertFields[tops, {F[2, {1}], -F[2, {1}]} -> {V[2], S[1]}, Model -> "SM",
       GenericModel -> "Lorentz", InsertionLevel -> Particles,
      ExcludeFieldPoints -> {FieldPoint[F, -F, S]}];

(*Display the graphs*)
graphs = Paint[inserttops, PaintLevel -> {Particles}, AutoEdit -> False,
      SheetHeader -> False, Numbering -> False, ColumnsXRows -> {3, 2}];

(*Generate amplitude*)
eezhb = CreateFeynAmp[inserttops, AmplitudeLevel -> Particles];

(*Save amplitude in a format understood by FeynCalc*)
PickLevel[Particles][eezhb] // ToFA1Conventions >> "eezhb.amp";
```

For completeness here is the input program for generating boxes of $e^+e^- \to ZH$ with FeynArts.

```
(*Define the Mandelstam variables and put momenta on - shell*)
  SetMandelstam[s, t, u, p1, p2, -k1, -k2, SmallVariable[ME],
    SmallVariable[ME], MZ, MH];

(*Set the option for the ordering of D0's*)
SetOptions[PaVeOrder, PaVeOrderList -> {{s, t}, {s, u}, {t, u}}];

(*Set the options for OneLoop*)
SetOptions[OneLoop, Mandelstam -> {s, t, u, MH^2 + MZ^2},
    Prefactor -> 1/ALPHA2,
    InitialSubstitutions -> {k2 -> p1 + p2 - k1, CW -> MW/MZ,
        EL -> Sqrt[4 Pi Sqrt[ALPHA2]]}, SmallVariables -> {ME, ME2}];

(*The option for OneLoopSum introduces abbreviations at the end*)
SetOptions[OneLoopSum, Prefactor -> 2 ALP4PI FLUFAC,
    Mandelstam -> {s, t, u, MH^2 + MZ^2},
    FinalSubstitutions -> {SW -> Sqrt[SW2], ME -> Sqrt[ME2], MW -> Sqrt[MW2],
        MZ -> Sqrt[MZ2], MH -> Sqrt[MH2],
        ME2^n_ :> ME^(2 n) /; Head[n] =!= Integer,
        MZ2^n_ :> MZ^(2 n) /; Head[n] =!= Integer,
        MW2^n_ :> MW^(2 n) /; Head[n] =!= Integer,
        MH2^n_ :> MH^(2 n) /; Head[n] =!= Integer,
        SW2^n_ :> SW^(2 n) /; Head[n] =!= Integer,
        StandardMatrixElement -> MBM}, WriteOutPaVe -> ""];

(*Define the standard matrix elements*)
SetStandardMatrixElements[{Spinor[p1].DiracSlash[
           Conjugate[Polarization[k1]]].ChiralityProjector[+1].Spinor[
           p2] -> {0, 1},
      Spinor[p1].DiracSlash[
           Conjugate[Polarization[k1]]].ChiralityProjector[-1].Spinor[
           p2] -> {0, 2},
      ScalarProduct[Conjugate[Polarization[k1]], p1]*
          Spinor[p1].DiracSlash[k1].ChiralityProjector[+1].Spinor[p2] -> {1,
          1}, ScalarProduct[Conjugate[Polarization[k1]], p1]*
          Spinor[p1].DiracSlash[k1].ChiralityProjector[-1].Spinor[p2] -> {1,
          2}, ScalarProduct[Conjugate[Polarization[k1]], p2]*
          Spinor[p1].DiracSlash[k1].ChiralityProjector[+1].Spinor[p2] -> {2,
          1}, ScalarProduct[Conjugate[Polarization[k1]], p2]*
          Spinor[p1].DiracSlash[k1].ChiralityProjector[-1].Spinor[p2] -> {2,
          2}}, {k2 -> (p1 + p2 - k1)}];

(*get the amplitudes, which have been written to a file by FeynArts*)
eezhamp = << eezhb.amp;

(*This calculates the amplitudes and sums them up*)
eezhboxes = OneLoopSum[eezhamp, CombineGraphs -> {1, 2, 3, 4, 5, 6}];

(*Here the result is written into a Mathematica file*)
Write2["eezhb.m", EEZHBOXES = FRH[eezhboxes]];

(*Here the result is written into a Mathematica program*)
Write2["eezhb.s", EEZHBOXES = eezhboxes];

(*Here the result is written into a Fortran file*)
Write2["eezhb.for", EEZHBOXES = eezhboxes, FormatType -> FortranForm];
```

Input program for calculating boxes of $e^+e^- \to ZH$.

```
        KK(1)= B0(MH2,MZ2,MZ2)
        KK(2)= B0(MZ2,ME2,ME2)
        KK(3)= B0(MH2,MW2,MW2)
        KK(4)= B0(MZ2,0D0,0D0)
        KK(5)= B0(MZ2,MW2,MW2)
        KK(6)= D0(MH2,ME2,MZ2,ME2,t,u,MZ2,MZ2,ME2,ME2)
        KK(7)= D0(MH2,MZ2,ME2,ME2,s,t,MW2,MW2,MW2,0D0)
        KK(8)= D0(MH2,MZ2,ME2,ME2,s,u,MW2,MW2,MW2,0D0)
        KK(9)= D0(MH2,ME2,MZ2,ME2,t,u,MW2,MW2,0D0,0D0)
        KK(10)= C0(MH2,t,ME2,MZ2,MZ2,ME2)
        KK(11)= C0(MH2,u,ME2,MZ2,MZ2,ME2)
        KK(12)= C0(MZ2,t,ME2,ME2,ME2,MZ2)
        KK(13)= C0(MZ2,u,ME2,ME2,ME2,MZ2)
        KK(14)= C0(MH2,MZ2,s,MW2,MW2,MW2)
        KK(15)= C0(MH2,t,ME2,MW2,MW2,0D0)
        KK(16)= C0(MH2,u,ME2,MW2,MW2,0D0)
        KK(17)= C0(MZ2,t,ME2,0D0,0D0,MW2)
        KK(18)= C0(MZ2,t,ME2,MW2,MW2,0D0)
        KK(19)= C0(MZ2,u,ME2,0D0,0D0,MW2)
        KK(20)= C0(MZ2,u,ME2,MW2,MW2,0D0)
        KK(21)= C0(s,ME2,ME2,MW2,MW2,0D0)
        KK(22)= B0(t,ME2,MZ2)
        KK(23)= B0(u,ME2,MZ2)
        KK(24)= B0(t,MW2,0D0)
        KK(25)= B0(u,MW2,0D0)
        KK(26)= 2*MH2 - MZ2
        KK(27)= MH2 - 5*MZ2
        KK(28)= 1 - 2*SW2
        KK(29)= MW2 + MZ2
        KK(30)= 2*MH2 - MW2
        KK(31)= MW2 + 2*MZ2
        KK(32)= 4*MW2**2 + 3*MW2*MZ2 - MZ2**2*SW2
        KK(33)= 5*MH2*MW2 - 2*MW2**2 + 2*MW2*MZ2 - MH2*MZ2*SW2
        KK(34)= MH2 - MW2 + 2*MZ2
        KK(35)= 2*MW2**2 + MW2*MZ2 - MZ2**2*SW2
        KK(36)= MW2 - MZ2*SW2
        KK(37)= MW2 + MZ2*SW2
        KK(38)= MH2 + MZ2
        KK(39)= 3*MW2 + MZ2*SW2
        KK(40)= MH2*MW2 + 4*MW2**2 + 2*MW2*MZ2 - 2*MZ2**2 + MH2*MZ2*SW2
        KK(41)= 2*MH2*MW2 - 2*MW2**2 + 4*MW2*MZ2 + MZ2**2*SW2
        KK(42)= 4*MH2 - 2*MW2 + 3*MZ2
        KK(43)= 2*MW2 - MZ2
        KK(44)= MH2*MW2 - 2*MW2**2 - 3*MW2*MZ2 + 2*MZ2**2 - MH2*MZ2*SW2
        KK(45)= MH2*MW2 + 4*MW2**2 + 4*MW2*MZ2 + MH2*MZ2*SW2
        KK(46)= 5*MW2 + MZ2*SW2
        KK(47)= 2*MH2 - MW2 + 2*MZ2
        KK(48)= 3*MH2*MW2 + 2*MW2**2 + 6*MW2*MZ2 + MH2*MZ2*SW2
        KK(49)= 9*MW2 + MZ2*SW2
        KK(50)= 2*MW2 + MZ2
        KK(51)= 3*MW2 + 2*MZ2
        KK(52)= MW2**2 - MH2*MZ2 + 3*MW2*MZ2 + MZ2**2
        KK(53)= 6*MH2*MW2 - 8*MW2**2 - MH2*MZ2*SW2 + MZ2**2*SW2
        KK(54)= MH2 - 2*MW2 + MZ2
        KK(55)= 4*MH2*MW2 - 4*MW2**2 - 2*MW2*MZ2 - MH2*MZ2*SW2 + MZ2**2*SW2
        KK(56)= MH2 - MZ2
        KK(57)= 2*MH2 + MZ2
        KK(58)= 2*MH2**2 + 4*MH2*MZ2 + MZ2**2
        KK(59)= MH2 - 2*MZ2
---> ... 418 lines omitted ...
        KK(173)= ALP4PI*FLUFAC*KK(172)
        EEZHBOXES = 2*KK(173)
```

Fortran output file "eezhb.for" generated by **OneLoopSum** for $e^+e^- \to ZH$.

```
KK[1] = (B0[MH2, MZ2, MZ2]
       );
KK[2] = (B0[MZ2, SmallVariable[ME2], SmallVariable[ME2]]
       );
KK[3] = (B0[MH2, MW2, MW2]
       );
KK[4] = (B0[MZ2, 0, 0]
       );
KK[5] = (B0[MZ2, MW2, MW2]
       );
KK[6] = ( D0[MH2, SmallVariable[ME2], MZ2, SmallVariable[ME2], t, u, MZ2, MZ2,
 SmallVariable[ME2], SmallVariable[ME2]]
       );
KK[7] = ( D0[MH2, MZ2, SmallVariable[ME2], SmallVariable[ME2], s, t, MW2, MW2, MW2, 0]
       );
KK[8] = ( D0[MH2, MZ2, SmallVariable[ME2], SmallVariable[ME2], s, u, MW2, MW2, MW2, 0]
       );
KK[9] = ( D0[MH2, SmallVariable[ME2], MZ2, SmallVariable[ME2], t, u, MW2, MW2, 0, 0]
       );
KK[10] = (C0[MH2, t, SmallVariable[ME2], MZ2, MZ2, SmallVariable[ME2]]
       );
KK[11] = (C0[MH2, u, SmallVariable[ME2], MZ2, MZ2, SmallVariable[ME2]]
       );
KK[12] = ( C0[MZ2, t, SmallVariable[ME2], SmallVariable[ME2], SmallVariable[ME2], MZ2]
       );
KK[13] = ( C0[MZ2, u, SmallVariable[ME2], SmallVariable[ME2], SmallVariable[ME2], MZ2]
       );
KK[14] = (C0[MH2, MZ2, s, MW2, MW2, MW2]
       );
KK[15] = (C0[MH2, t, SmallVariable[ME2], MW2, MW2, 0]
       );
KK[16] = (C0[MH2, u, SmallVariable[ME2], MW2, MW2, 0]
       );
KK[17] = (C0[MZ2, t, SmallVariable[ME2], 0, 0, MW2]
       );
KK[18] = (C0[MZ2, t, SmallVariable[ME2], MW2, MW2, 0]
       );
KK[19] = (C0[MZ2, u, SmallVariable[ME2], 0, 0, MW2]
       );
KK[20] = (C0[MZ2, u, SmallVariable[ME2], MW2, MW2, 0]
       );
KK[21] = (C0[s, SmallVariable[ME2], SmallVariable[ME2], MW2, MW2, 0]
       );
KK[22] = (B0[t, MZ2, SmallVariable[ME2]]
       );
KK[23] = (B0[u, MZ2, SmallVariable[ME2]]
       );
KK[24] = (B0[t, 0, MW2]
       );
KK[25] = (B0[u, 0, MW2]
       );
KK[26] = (2*MH2 - MZ2
       );
KK[27] = (MH2 - 5*MZ2
       );
KK[28] = (1 - 2*SW2
       );
KK[29] = (MW2 + MZ2
       );
KK[30] = (2*MH2 - MW2
       );
(* ............... 564 lines omitted ..................... *)
KK[173] = (ALP4PI*FLUFAC*HoldForm[KK[172]]
       );
EEZHBOXES = 2*HoldForm[KK[173]]
```

Mathematica output file "eezhb.s" generated by **OneLoopSum** for $e^+e^- \to ZH$.

1.6 Processing Amplitudes

Calculating loop integrals will in many cases lead to very large expressions. The same goes for the application of other algorithms like Dirac tracing algorithms. The systematization and reduction of such expressions is a process which requires much more human planing, control and intervention than the process leading to the expressions. However, the computer is still of help. In this section shall be considered a few examples of how one may proceed, constructing small Mathematica programs taking advantage of tools provided by FeynCalc.

Load a stored amplitude (see section 1.5).

```
In[262]:= << "eezhb.m"
```

Check the 'size' of the expression.

```
In[263]:= EEZHBOXES // LeafCount
Out[263]= 10415
```

Expand the expressions, collect with respect to factors we know will be overall or "interesting", simplify what the non-overall factors multiply.

```
In[264]:= eezhoxes = Collect[EEZHBOXES // Expand,
            ALP4PI, FLUFAC, _MBM, _D0, _C0, _B0,
            If[FreeQ[#, _MBM | _D0 | _C0 | _B0 |
            _PaVe], FullSimplify[#], #] &];
```

Check the "size" of the resulting expression.

```
In[265]:= eezhoxes // LeafCount
Out[265]= 8881
```

Clean up.

```
In[266]:= DeleteFile /@ FileNames["eezhb*"];
          DeleteFile /@ FileNames["PaVe*"];
```

In the example above, notice that instead of simply applying **Simplify** to the whole expression, some grouping is first done and then **Simplify** is applied to individual terms. This is often advantageous because the performance of **Simplify** naturally scales very badly with the size of expressions. We remark that although **OneLoopSum** does a decent job in structuring the expression, it can still be reduced somewhat.

In the one-loop calculations considered thus far, amplitudes have been computed. The extra step of computing the (differential) cross section can also be done with FeynCalc. To demonstrate this we pick a very simple example, namely the famous Møller cross section. This example also demonstrates that a full calculation from Feynman rules to cross section can be carried out with FeynArts and FeynCalc[3].

[3] In fact, as we have seen in section 4.5, calculation of Feynman rules from lagrangians can also be automatized with FeynCalc.

```
(*Construction of topologies*)
tops = CreateTopologies[0, 2 -> 2, Adjacencies -> {3},
      ExcludeTopologies -> {SelfEnergies, WFCorrections}];

(*Check*)
Paint[tops, ColumnsXRows -> {3, 1}, AutoEdit -> False];

(*Field insertion*)
inserttops =
    InsertFields[tops, {F[1, {1}], F[1, {1}]} -> {F[1, {1}], F[1, {1}]},
      Model -> "QED", GenericModel -> "QED", InsertionLevel -> Particles];

(*Check*)
treegraphs =
    Paint[inserttops, PaintLevel -> {Particles}, AutoEdit -> False,
      SheetHeader -> False, Numbering -> False, ColumnsXRows -> {2, 1}];

(*Calculate the amplitudes*)
amps = CreateFeynAmp[inserttops, AmplitudeLevel -> Particles];

(*Save result*)
PickLevel[Classes][amps] // ToFA1Conventions >> "moelleramps.m"
```

For completeness here is the input program for generating the (leading order) diagrams of Møller scattering with FeynArts.

```
(*Define the Mandelstam variables and put momenta on - shell*)
  SetMandelstam[s, t, u, p1, p2, -k1, -k2, ME, ME, ME, ME];

(*get the amplitudes, which have been written to a file by FeynArts*)
amps = << moelleramps.m;

(*Sum the graphs and contract Lorentz indices*)
amp = (OneLoopSum[amps, CombineGraphs -> {1, 2}] // FRH) /. D :> Sequence[] //
     Contract

(*Square the amplitude,
  transform the spin sums into traces and evaluate the traces*)
squaredamp =
  FermionSpinSum[amp ComplexConjugate[amp /. li2 -> li1] // Expand] /.
          DiracTrace -> Tr // DiracSimplify //
      TrickMandelstam[#, {s, t, u, 4ME^2}] & // Simplify
squaredamp1 =
  squaredamp // Contract // PropagatorDenominatorExplicit // Simplify

(*The kinematical factor in the center of mass frame*)
kinfac = 1/(64 \[Pi]^2s);

(*The full differential cross section in the center of mass frame expressed \
in terms of Mandelstam variables*)
dcrosssection = 1/4*kinfac*squaredamp1 // Simplify
(*Shift to other variables : Scattering angle and half the CMS energy*)
dc = dcrosssection /. u -> 4ME^2 - s - t /. {s -> 4 \[Omega]^2,
          t -> -2 q2(1 - Sqrt[1 - sin[\[Theta]]^2])} /.
      q2 -> \[Omega]^2 - ME^2 // Simplify

(*Clean up*)
DeleteFile["moelleramps.m"];
```

Calculation of the Møller cross section.

Notice that the argument given to **`FermionspinSum`** is a product of two amplitudes. These both contain contracted indices; therefore, in one of them, these indices have to be renamed. The functions **`TrickMandelstam`** and **`PropagatorDenominatorExplicit`** are described in section 6.

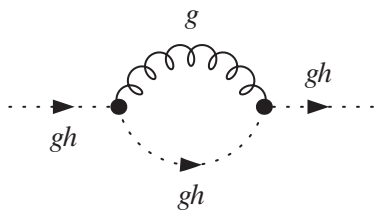
ComplexConjugate[*exp***]**	conjugates the expression *exp*, operating on fermion lines
FermionSpinSum[*exp***]**	constructs traces out of squared amplitudes

Calculation of squares of fermion amplitudes.

2 QCD

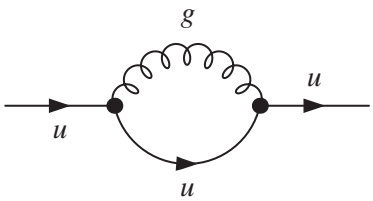
The modern text book literature on QCD relies to a large extent on one-loop results carried out over the last decades. Despite this, there are relatively few explicit 1-loop QCD calculations available. In this section, as part of the aim of this book (section 1), the 1-loop self energy calculation of the Appendix of [25] is redone with FeynCalc. Furthermore, examples of two-loop gluon and quark self energy calculations in a general covariant gauge are given. In those examples, in order to reduce all 2-loop integrals to master integrals, the recursion algorithm of Tarasov [9] as implemented in Tarcer [8] is employed.

2.1 The 1-loop ghost self energy



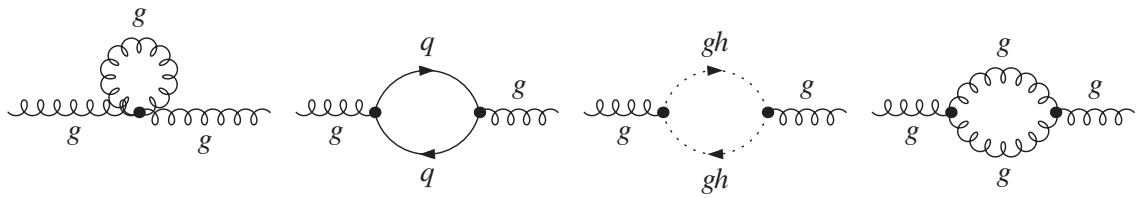
2.2 The 2-loop ghost self energy

2.3 The 1-loop quark self energy



2.4 The 2-loop quark self energy

2.5 The 1-loop gluon self energy



2.6 The 2-loop gluon self energy

The graphs

```
In[267]:= Paint[inserts = InsertFields[Rest@CreateTopologies[2, 1 → 1,
            ExcludeTopologies → {Internal}], {V[5]} → {V[5]},
            InsertionLevel → {Generic}, GenericModel- > "FCQCDLorentz",
            Model → "FCQCD"], ColumnsXRows → {3, 3}];
```

Calculating the bare 2-loop on-shell gluon self energy diagrams

In the Model file incoming momenta are specified by **p1**, **p2**, ... and all outgoing by **k1**, **k2**, .. .
We substitute p1 to p and k1 to -p and do some further substitutions.

```
In[268]:= amps = CreateFeynAmp[inserts, Truncated → True, PreFactor → 1]/.k1 → -p1/.p1 → p/.
            FeynAmpList[__] :⇒ List /. FeynAmp[_, _, x_] :⇒ x;

In[269]:= amps[[3]]
```

$$\text{Out[269]}= \frac{1}{2}\,\Pi_g^{\text{li9li10}}(p-q_1)\,\Pi_g^{\text{li3li4}}(q_1)\,\Pi_g^{\text{li11li12}}(-p-q_2)\,\Pi_g^{\text{li5li6}}(q_2)\,\Pi_g^{\text{li7li8}}(q_1+q_2)$$
$$V^{\text{li1li4li10}}(p,\ q_1,\ p-q_1)\,V^{\text{li2li6li12}}(p,\ q_2,\ -p-q_2)$$
$$V^{\text{li3li5li8}}(-q_1,\ -q_2,\ q_1+q_2)\,V^{\text{li7li9li11}}(-q_1-q_2,\ q_1-p,\ p+q_2)$$
$$f_{\text{ci1ci4ci10}}\,f_{\text{ci2ci5ci11}}\,f_{\text{ci4ci5ci8}}\,f_{\text{ci8ci10ci11}}$$

```
In[270]:= amp31 = amps[[3]]/.q2 → -q2
```

Out[270]=
$$\frac{1}{2}\,\Pi_g^{\text{li9li10}}(p-q_1)\,\Pi_g^{\text{li3li4}}(q_1)\,\Pi_g^{\text{li7li8}}(q_1-q_2)\,\Pi_g^{\text{li5li6}}(q_2)\,\Pi_g^{\text{li11li12}}(q_2-p)$$
$$V^{\text{li1li4li10}}(p,\ q_1,\ p-q_1)\,V^{\text{li2li6li12}}(p,\ -q_2,\ q_2-p)$$
$$V^{\text{li3li5li8}}(-q_1,\ q_2,\ q_1-q_2)\,V^{\text{li7li9li11}}(q_2-q_1,\ q_1-p,\ p-q_2)$$
$$f_{\text{ci1ci4ci10}}\,f_{\text{ci2ci5ci11}}\,f_{\text{ci4ci5ci8}}\,f_{\text{ci8ci10ci11}}$$

In[271]:=
```
amp31//InputForm
```
Out[271]=
```
(GluonPropagator[p - q1, {li9}, {li10}]*
 GluonPropagator[q1, {li3}, {li4}]*
 GluonPropagator[q1 - q2, {li7}, {li8}]*
 GluonPropagator[q2, {li5}, {li6}]*
 GluonPropagator[-p + q2, {li11}, {li12}]*
 GluonVertex[{p, li1}, {q1, li4}, {p - q1, li10}]*
 GluonVertex[{p, li2}, {-q2, li6}, {-p + q2, li12}]*
 GluonVertex[{-q1, li3}, {q2, li5}, {q1 - q2, li8}]*
 GluonVertex[{-q1 + q2, li7}, {-p + q1, li9},
 {p - q2, li11}] * SUNF[ci1, ci4, ci10]*
 SUNF[ci2, ci5, ci11] * SUNF[ci4, ci5, ci8]*
 SUNF[ci8, ci10, ci11])/2
```

In[272]:=
```
amp32 = SUNSimplify[Explicit@ amp31]
```
Out[272]=
$$-\left(i\,C_A^2\right)$$
$$g^{\text{li10li9}}\,g^{\text{li11li12}}\,g^{\text{li3li4}}\left(-p^{\text{li1}}\,g^{\text{li10li4}}+2\,q_1^{\text{li1}}\,g^{\text{li10li4}}+g^{\text{li1li4}}\,p^{\text{li10}}-g^{\text{li1li4}}\,q_1^{\text{li10}}-g^{\text{li1li10}}\,q_1^{\text{li4}}\right)$$
$$g^{\text{li5li6}}\left(p^{\text{li12}}\,g^{\text{li2li6}}+q_2^{\text{li12}}\,g^{\text{li2li6}}+g^{\text{li12li6}}\,p^{\text{li2}}-2\,g^{\text{li12li6}}\,q_2^{\text{li2}}-2\,g^{\text{li12li2}}\,p^{\text{li6}}+g^{\text{li12li2}}\,q_2^{\text{li6}}\right)$$
$$g^{\text{li7li8}}\left(-q_1^{\text{li3}}\,g^{\text{li5li8}}+2\,q_2^{\text{li3}}\,g^{\text{li5li8}}+2\,g^{\text{li3li8}}\,q_1^{\text{li5}}-g^{\text{li3li8}}\,q_2^{\text{li5}}-g^{\text{li3li5}}\,q_1^{\text{li8}}-g^{\text{li3li5}}\,q_2^{\text{li8}}\right)$$
$$\left(p^{\text{li11}}\,g^{\text{li7li9}}-2\,q_1^{\text{li11}}\,g^{\text{li7li9}}+q_2^{\text{li11}}\,g^{\text{li7li9}}-2\,g^{\text{li11li9}}\,p^{\text{li7}}+g^{\text{li11li9}}\,q_1^{\text{li7}}+g^{\text{li11li9}}\,q_2^{\text{li7}}+\right.$$
$$\left.g^{\text{li11li7}}\,p^{\text{li9}}+g^{\text{li11li7}}\,q_1^{\text{li9}}-2\,g^{\text{li11li7}}\,q_2^{\text{li9}}\right)\delta_{\text{ci1ci2}}\Big)\Big/$$
$$\left(4\,(p-q_1)^2\,q_1^2\,(q_1-q_2)^2\,q_2^2\,(q_2-p)^2\right)$$

In[273]:=
```
Timing[
  amp33 =
    Collect2[FeynAmpDenominatorCombine@
        Contract3[Explicit[amp32]/.li1 → li2/.SUNDelta[__] :→ 1], q1, q2]; ]
```
Out[273]=
```
{0.71 Second, Null}
```

In[274]:=
```
amp33
```

Out[274]=
$$-\frac{i\,C_A^2\,(11-5\,D)\,p\cdot q_1{}^2}{2\,q_1^2.q_2^2.(p-q_1)^2.(q_2-p)^2.(q_1-q_2)^2}-\frac{i\,C_A^2\,(5-2\,D)\,q_1\cdot q_2{}^2}{2\,q_1^2.q_2^2.(p-q_1)^2.(q_2-p)^2.(q_1-q_2)^2}-$$
$$\frac{3\,i\,C_A^2\,(1-D)\,p^2\,p\cdot q_1}{2\,q_1^2.q_2^2.(p-q_1)^2.(q_2-p)^2.(q_1-q_2)^2}+\frac{i\,C_A^2\,(34-19\,D)\,p\cdot q_1\,p\cdot q_2}{2\,q_1^2.q_2^2.(p-q_1)^2.(q_2-p)^2.(q_1-q_2)^2}+$$
$$\frac{i\,C_A^2\,(7-4\,D)\,p^2\,q_1^2}{q_1^2.q_2^2.(p-q_1)^2.(q_2-p)^2.(q_1-q_2)^2}-\frac{i\,C_A^2\,(14-11\,D)\,p\cdot q_2\,q_1^2}{2\,q_1^2.q_2^2.(p-q_1)^2.(q_2-p)^2.(q_1-q_2)^2}-$$
$$\frac{5\,i\,C_A^2\,(5-2\,D)\,p^2\,q_1\cdot q_2}{2\,q_1^2.q_2^2.(p-q_1)^2.(q_2-p)^2.(q_1-q_2)^2}+\frac{i\,C_A^2\,(5-2\,D)\,p\cdot q_1\,q_1\cdot q_2}{2\,q_1^2.q_2^2.(p-q_1)^2.(q_2-p)^2.(q_1-q_2)^2}+$$
$$\frac{i\,C_A^2\,(5-2\,D)\,p\cdot q_2\,q_1\cdot q_2}{2\,q_1^2.q_2^2.(p-q_1)^2.(q_2-p)^2.(q_1-q_2)^2}-\frac{i\,C_A^2\,(14-11\,D)\,p\cdot q_1\,q_2^2}{2\,q_1^2.q_2^2.(p-q_1)^2.(q_2-p)^2.(q_1-q_2)^2}+$$
$$\frac{i\,C_A^2\,(14-11\,D)\,q_1^2\,q_2^2}{2\,q_1^2.q_2^2.(p-q_1)^2.(q_2-p)^2.(q_1-q_2)^2}$$

In[275]:=
```
amp34 = ToTFi[amp33, q1, q2, p]//FCE
```

Out[275]=
$$-\frac{5}{2}\,i\,(5-2\,D)\,p^2\,F^{(D)\,00001}_{\{1,0\}\{1,0\}\{1,0\}\{1,0\}\{1,0\}}\,C_A^2-\frac{1}{2}\,i\,(5-2\,D)\,F^{(D)\,00002}_{\{1,0\}\{1,0\}\{1,0\}\{1,0\}\{1,0\}}\,C_A^2+$$
$$\frac{1}{2}\,i\,(5-2\,D)\,F^{(D)\,00011}_{\{1,0\}\{1,0\}\{1,0\}\{1,0\}\{1,0\}}\,C_A^2-\frac{3}{2}\,i\,(1-D)\,p^2\,F^{(D)\,00100}_{\{1,0\}\{1,0\}\{1,0\}\{1,0\}\{1,0\}}\,C_A^2+$$
$$\frac{1}{2}\,i\,(5-2\,D)\,F^{(D)\,00101}_{\{1,0\}\{1,0\}\{1,0\}\{1,0\}\{1,0\}}\,C_A^2+\frac{1}{2}\,i\,(34-19\,D)\,F^{(D)\,00110}_{\{1,0\}\{1,0\}\{1,0\}\{1,0\}\{1,0\}}\,C_A^2-$$
$$\frac{1}{2}\,i\,(11-5\,D)\,F^{(D)\,00200}_{\{1,0\}\{1,0\}\{1,0\}\{1,0\}\{1,0\}}\,C_A^2-\frac{1}{2}\,i\,(14-11\,D)\,F^{(D)\,01100}_{\{1,0\}\{1,0\}\{1,0\}\{1,0\}\{1,0\}}\,C_A^2+$$
$$i\,(7-4\,D)\,p^2\,F^{(D)\,10000}_{\{1,0\}\{1,0\}\{1,0\}\{1,0\}\{1,0\}}\,C_A^2-\frac{1}{2}\,i\,(14-11\,D)\,F^{(D)\,10010}_{\{1,0\}\{1,0\}\{1,0\}\{1,0\}\{1,0\}}\,C_A^2+$$
$$\frac{1}{2}\,i\,(14-11\,D)\,F^{(D)\,11000}_{\{1,0\}\{1,0\}\{1,0\}\{1,0\}\{1,0\}}\,C_A^2$$

In[276]:=
```
amp35 = TarcerRecurse[amp34]
```

Out[276]=
$$-\frac{i\,(2\,D^2-57\,D+136)\,p^2\,B^{(D)}_{\{1,0\}\{1,0\}}{}^2\,C_A^2}{16\,(D-4)}-\frac{i\,(2\,D^3+19\,D^2-124\,D+184)\,J^{(D)}_{\{1,0\}\{1,0\}\{1,0\}}\,C_A^2}{4\,(D-4)^2}$$

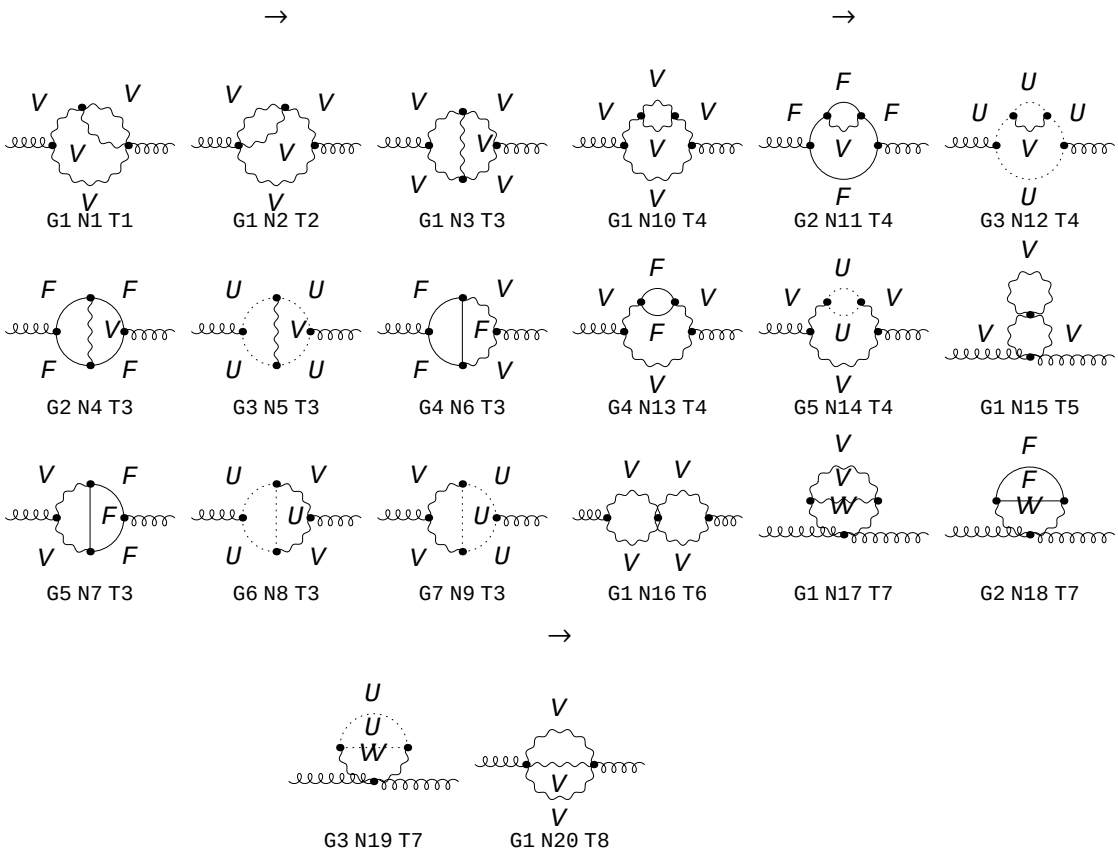


Figure 2.3: 2-loop gluon self-energy diagrams.

3 Chiral Perturbation Theory

3.1 Introduction

The package PHI (Phenomenology of Hadronic Interactions) provides utilities for computations in the Chiral Perturbation Theory (ChPT). The general idea is to allow writing up a calculation starting with a lagrangian and ending up with an amplitude, all within the framework of FeynCalc. For this to be possible, many functions on a rather general level were implemented, as was a method of interacting seamlessly with FeynArts. The following sections cover the more important features of the package. A fuller description can be found in [22].

The motivation for writing PHI was a lack of software for implementing an effective model like one of the various ChPT models, which have a more complicated power counting than models which simply expand Green's functions perturbatively in a coupling constant. However, the package is general enough that other models can be defined, as has been demonstrated with the simple example of QED.

The main features of PHI are:

- A set of basic objects are provided that can be composed and manipulated to form ChPT lagrangians.
- The most common ChPT lagrangians are included and new ones can easily be defined.
- The lagrangians can be expanded in terms of pion (meson) fields, with SU(2) (SU(3)) flavor traces being done automatically.
- External sources can be switched off and on.
- FeynArts is used for generating Feynman diagrams and amplitudes, including counter-terms. Power counting and storing of Feynman rules is systematized.

3.2 Models and Lagrangians

PHI comes with a number of predefined chiral models (or configurations). These define lagrangians etc. in terms of quantum fields. Which configuration and lagrangians are loaded should be defined before startup of FeynCalc. When loading a langrangian, it is added to the database of lagrangians, **Lagrangian**. In section 3.5 is discussed how to construct such lagrangians.

Quit the kernel and reload FeynCalc with the packages PHI and FeynArts enabled. Have PHI load "ChPTVirtualPhotons2" and its two lagrangians.

```
In[277]:= Quit[];

          $LoadPhi = True;
          $LoadFeynArts = True;
          $Configuration = "ChPTVirtualPhotons2";
          $Lagrangians = "ChPTVirtualPhotons2"[2],
            "ChPTVirtualPhotons2"[4];

          Get["HighEnergyPhysics`FeynCalc`"];
```

Inspect the leading order lagrangian.

```
In[278]:= Lagrangian[ChPTVirtualPhotons2[2]]
Out[278]= 1/4 (⟨χ✱U†⟩ + ⟨χ†✱ U⟩ + ⟨D_μ(U)✱D_μ(U)†⟩) (f_π)^2 −
          1/2 λ ∂_μ(γ_μ) ∂_ν(γ_ν) − 1/4 (γ_μν✱γ_μν) +
          C ⟨Q_R✱ U✱Q_L✱ U†⟩
```

$Configuration is a short name denoting some physics model, in the case above, ChPT with virtual photons in SU(2). $Lagrangians is then a list of the lagrangians defined with this model, in the case above the lagrangians of leading and next-to-leading order (2 and 4) in the chiral expansion.

$Configuration	string valued variable determining which configuration is loaded at startup
$Lagrangians	string valued variable determining which lagrangians are loaded at startup

PHI startup settings.

The list of available configurations and lagrangians can be found by browsing the directory "HighEnergyPhysics/Phi/Configurations" and "HighEnergyPhysics/Phi/Lagrangians". Currently the list reads

"ChPT2"	ChPT with in SU(2)
"ChPT3"	ChPT in SU(3)
"ChPTVirtualPhotons2"	ChPT in SU(2) with virtual photons
"ChPTVirtualPhotons3"	ChPT in SU(3) with virtual photons
"ChPTPhotonsLeptons3"	ChPT in SU(3) with virtual photons and leptons
"ChPTW3"	ChPT in SU(3) with ΔI=1 weak interaction

PHI Possible settings of **$Configuration**.

3.3 Quantum Fields and their Space-Time Derivatives

The field argument of **QuantumField** can be any symbol. Notice however, that the various configurations files for PHI each use some conventions; e.g. for a pion the symbol **Particle[Pion]** is used. The Dirac bar can be applied to fermionic **QuantumField**s with the function **DiracBar**. Since partial derivatives and covariant derivatives may be performed on **QuantumField**s as well as on polynomials of **QuantumField**s, the **QuantumField**s usually carry a space-time argument, e.g. x. For the derivatives, the functions, **FieldDerivative** and **CovariantFieldDerivative** are provided. As already seen, an equivalent way of calculating space-time derivatives is using "." and the operators of page 55. The "." operator is, however, also used for multiplying Dirac matrices, so to avoid confusion **FieldDerivative** may be used.

A single derivative of some field p.

```
In[279]:= FieldDerivative[QuantumField[p][x], x,
           LorentzIndex[μ]]
Out[279]= ∂μπ
```

The result is expressed using the symbol **PartialD**.

```
In[280]:= % // StandardForm
Out[280]= QuantumField[PartialD[LorentzIndex[μ]], p][x]
```

The covariant derivative of the squared pion field (defined by the loaded model), $D_\mu \phi_\pi^2$.

```
In[281]:= CovariantFieldDerivative[QuantumField[Particle[
            Pion]][x]^2, x, LorentzIndex[μ]]//
          CommutatorReduce
Out[281]= iπ^2 (V⃗μ·σ⃗ − A⃗μ·σ⃗) − i (A⃗μ·σ⃗ + V⃗μ·σ⃗) π^2 +
          iγμ QL π^2 − iγμ QR π^2 + 2π ∂μπ
```

CommutatorReduce pulls out abelian quantities of non-abelian products (see below).

FieldDerivative[g[x], x, {l1, l2, ...}]	calculates $\frac{\partial}{\partial x_{l_1}} \frac{\partial}{\partial x_{l_2}} \ldots$ on the expression **g[x]**
CovariantFieldFieldDerivative[g[x], x, {l1, l2, ...}]	as above but with the vector and axial-vector source field terms included. This is defined in the chosen configuration file

Derivatives acting on **QuantumField**s.

Many operations can be performed on expressions involving **QuantumField**s. We shall see some in the examples to come. But first we turn to some of the objects that PHI adds to the basic framework of FeynCalc. Notice that in ChPT, a condensed notation is used and many objects like e.g. the basic **MM[x]**, which is the PHI notation

for the U of ChPT, actually contain **QuantumField**s [4].

3.4 Vectors, Matrices and Structure Constants in SU(2) and SU(3)

QuantumFields may be grouped in flavor SU(N) matrices or vectors[5] with the functions **IsoVector**, **UMatrix** and **UVector**. When this is done, these functions take over the space-time argument.

IsoVector[*v***]**	represents an SU(2) or SU(3) multiplet with number of entries corresponding to the number of generators (3 or 8)
UVector[*v***]**	represents an SU(2) or SU(3) vector with number of entries corresponding to the dimension of the representation (2 or 3)
UMatrix[*m***]**	is recognized as a square matrix of dimension 2 or 3

SU(N) vectors and matrices.

option name	*default value*	
SUNN	**2**	number of quark flavors. Either 2 or 3
UDimension	**Automatic**	dimension of the representation

Options determining the dimension of SU(N) vectors and matrices.

The SU(N) matrices and vectors may be multiplied with each other using some new functions; **NM** ("✱" in **OutputForm**) is used for multiplication of SU(N) matrices, **IsoDot** ("·") is used for the dot product of SU(N) iso-vectors. The Mathematica**Dot** (".") is used for multiplication of SU(N) matrices with SU(N) vectors. SU(N) matrices may be adjoined, conjugated and transposed.

Dot product of two iso-vectors.

```
In[282]:= IsoDot[IsoVector[a], IsoVector[b]]
Out[282]= a⃗ · b⃗
```

[4] Remember that an explanation of the various symbols can be obtained by using the **?** operator.
[5] with SU(N) we shall understand SU(N), where N=2 or 3.

A more complicated structure. Iso-vectors of SU(N) matrices are allowed.

```
In[283]:= UVector[Ψ] . NM[ IsoDot[IsoVector[a],
            IsoVector[UMatrix[b]]],
            Adjoint[UMatrix[c]]] . UVector[Ψ]
```

$$\text{Out[283]=}\ \vec{\Psi}.\left(\vec{a}\cdot\vec{b}\star c^{\dagger}\right).\vec{\Psi}$$

NM[*a*, *b*, ...]	non-commutative multiplication for matrices and/or fields *a*, *b*, ...
IsoDot[*a*, *b*]	dot product for iso-vectors *a*, *b*
IsoCross[*a*, *b*]	anti-symmetric cross product for isospin vectors *a*, *b*
IsoSymmetricCross	symmetric cross product for isospin vectors *a*, *b*
Dot[*a*, *b*]	the usual dot product, "**.**", is used for multiplication of **UVectors** with **UVectors** or **UMatrices**

Multiplication operators for SU(N) vectors and matrices.

Adjoint[*m*]	the adjoint $m^{\dagger}$ of a matrix *m*
Conjugate[*m*]	the conjugate m^{c} of a matrix *m*
Transpose[*m*]	the transpose m^{t} of a matrix *m*

Operations on matrices.

The matrices generating flavor SU(N) are denoted **UGenerator**.

UGenerator[*i*]	**UMatrix[UGenerator[*i*]]** is the *i*'th generator of SU(*N*)

Generators of flavor SU(2) or SU(3).

SU(*N*) vectors and matrices may be written out explicitly using **WriteOutUMatrices** and/or **WriteOutIsoVectors**.

WriteOutUMatrices[*exp***]**	write out SU(*N*) vectors in *exp*
WriteOutIsoVectors[*exp***]**	write out SU(*N*) matrices in *exp*

Writing out explicit components of SU(*N*) vectors and matrices.

The dot product of two iso-vectors, *a* and *b*. Notice that the default is to assume non-commutativity and use **NM** for multiplication.

```
In[284]:= IsoDot[IsoVector[a], IsoVector[b]] //
          WriteOutIsoVectors
Out[284]= a(1)^c*b(1) + a(2)^c*b(2) + a(3)^c*b(3)
```

Some matrix, *a*, written out in components.

```
In[285]:= UMatrix[a] // WriteOutUMatrices
Out[285]= ( a(1, 1)  a(1, 2) )
          ( a(2, 1)  a(2, 2) )
```

The first generator of SU(*N*) written out in components. In this case, the first of the triplet of generators of SU(2), because the default value of the option **SUNN** of **UMatrix** has been set to 2 by the active model file.

```
In[286]:= UMatrix[UGenerator[1]] //
          WriteOutUMatrices
Out[286]= ( 0   -1 )
          ( -1   0 )
```

The triplet of generators of SU(2).

```
In[287]:= IsoVector[UMatrix[UGenerator[]]]
Out[287]= σ⃗
In[288]:= % // WriteOutIsoVectors
Out[288]= ⟨σ^1, σ^2, σ^3⟩
```

```
In[289]:= % // WriteOutUMatrices
```

$$\text{Out[289]=}\ \langle\left(\begin{matrix}0 & -1\\ -1 & 0\end{matrix}\right), \left(\begin{matrix}0 & -i\\ i & 0\end{matrix}\right), \left(\begin{matrix}-1 & 0\\ 0 & 1\end{matrix}\right)\rangle$$

Products using the multiplication operators for SU(N) vectors and matrices may be written out using contracted symbolic indices with the function **IsoIndicesSupply**.

IsoIndicesSupply[*exp***]**	replaces dot and cross products of SU(N) vectors with contracted indices

Writing out components of SU(N) vectors and matrices using symbolic indices.

Write out a dot-product with a contracted index.

```
In[290]:= IsoDot[IsoVector[a], IsoVector[b]] //
          IsoIndicesSupply
```

$$\text{Out[290]=}\ a(i_1)^c \star b(i_1)$$

Applying **IsoIndicesSupply** on expressions using **IsoDot**, etc. will typically result in expressions involving the Kronecker delta function and the structure constants of SU(N), with $N = 2$ or $N = 3$, depending on the setting of the option **SUNN**.

SU2Delta[*i***, ***j***]**	the Kronecker delta function δ_{ij} in SU(2)
SU2F[*i***, ***j***, ***k***]**	the antisymmetric structure constants d_{ijk} in SU(2)
SU3Delta[*i***, ***j***]**	the Kronecker delta function δ_{ij} in SU(3)
SU3F[*i***, ***j***, ***k***]**	the antisymmetric structure constants d_{ijk} in SU(3)
SU3D[*i***, ***j***, ***k***]**	the symmetric structure constants d_{ijk} in SU(3)

The Kronecker delta function and the structure constants of SU(2) and SU(3).

To reduce or otherwise manipulate expressions expressions involving SU(N) indices and/or non-commutative

multiplication of SU(N) matrices, the three functions listed below are provided.

IndicesCleanup[*exp***]**	renames Lorentz and SU(N) indices in a systematic way
SUNReduce[*exp***]**	does some reduction on expressions involving SU(N) indices
CommutatorReduce[*exp***]**	does some reduction on expressions containing non-abelian SU(N) products in *exp*

SU(N) utilities.

To calculate the trace of flavor SU(N) matrices, the function **Utrace** is used. This function distributes over sums. If the trace of a term is not known, **Utrace** is simply replaced with **Utrace1**, which displays in **TraditionalForm** as $\langle\cdots\rangle$.

UTrace[*m***]**	calculate the flavor SU(N) trace of the matrix m
UTrace1[*m***]**	return value if the trace of m is not known

The trace of flavor SU(N) matrices.

A few abbreviations

```
In[291]:= id = IsoDot;
          iv = IsoVector;
          ug = IsoVector[UMatrix[UGenerator[]]];
```

A deceptively complicated example, $\langle\left(\vec{a}\cdot\vec{\sigma}+\vec{d}\cdot\vec{\sigma}\right)\vec{b}\cdot\vec{\sigma}\vec{c}\cdot\vec{\sigma}\rangle-\langle\vec{b}\cdot\vec{\sigma}\vec{c}\cdot\vec{\sigma}\vec{a}\cdot\vec{\sigma}\rangle$

```
In[292]:= UTrace[
          NM[id[iv[a], ug] + id[iv[d], ug],
            id[iv[b], ug], id[iv[c], ug]] -
          NM[id[iv[b], ug], id[iv[c], ug],
            id[iv[a], ug]]]
```

$$\text{Out[292]}= \langle\left(\vec{a}\cdot\vec{\sigma}+\vec{d}\cdot\vec{\sigma}\right)\ast\vec{b}\cdot\vec{\sigma}\ast\vec{c}\cdot\vec{\sigma}\rangle-\langle\vec{b}\cdot\vec{\sigma}\ast\vec{c}\cdot\vec{\sigma}\ast\vec{a}\cdot\vec{\sigma}\rangle$$

Expanding the non-commutative products causes the trace to distribute over the sum. Cycling the trace to a unique form can often cause some cancellation.

```
In[293]:= % // NMExpand // CycleUTraces
```

$$\text{Out[293]}= \langle\vec{b}\cdot\vec{\sigma}\ast\vec{c}\cdot\vec{\sigma}\ast\vec{d}\cdot\vec{\sigma}\rangle$$

When the monomials of generator matrices are reduced to order 1 or 0, the trace is known.

```
In[294]:= % // ExpandU
Out[294]= 2 i b^c × c . d^c
```

Indices are supplied. Vector products are thereby written in terms of the SU(N) structure constants.

```
In[295]:= % // IsoIndicesSupply //
          CommutatorReduce
Out[295]= 2 i b(i2)^c c(i3)^c d(i4)^c f(i2 i3 i4)
```

NMExpand[*exp***]**	expand any **NM** products in *exp* using distributivity
CycleUTraces[*exp***]**	Use cyclicity to write all traces in a canonical form
ExpandU[*exp***]**	expand **IsoDot** products involving generator matrices into products containing at most one generator matrix.

Utilities for manipulating non-commutative products of falvour SU(N) matrices.

These functions each take a number of options, allowing to tune their behaviour. In particular, **SUNReduce** takes the option **FullReduce**, which, when set to **True** triggers some extra reduction and iterates until the result no longer changes. By default it is set to **False**, so, iterated application on an expression can cause the result to change on each iteration.

option name	*default value*	
FullReduce	**False**	iterate until the result no longer changes
Explicit	**False**	perform explicit sums over repeated indices
HoldSums	**True**	when **Explicit** is set to **True**, only write the summation symbols, don't do the summations
SUNN	**2**	dimension of the representation

Options of **SUNReduce**.

Another deceptively complicated example.

```
In[296]:= k(IsoDot[IsoCross[IsoVector[a],
          IsoVector[b]], IsoCross[IsoVector[a],
          IsoVector[b]]] + IsoDot[IsoCross[IsoVector[a],
          IsoVector[c]], IsoCross[IsoVector[a],
          IsoVector[c]]])
```

$$\text{Out[296]=}\ k\left(\vec{a}\times\vec{b}\cdot\vec{a}\times\vec{b}+\vec{a}\times\vec{c}\cdot\vec{a}\times\vec{c}\right)$$

Write out with contracted indices.

```
In[297]:= % // IsoIndicesSupply
```

$$\text{Out[297]=}\ k\left((a(i_2)\star b(i_3)^c\star a(i_4)^c\star b(i_5))\ f_{i_{10}i_2i_3}\ f_{i_{10}i_4i_5}+(a(i_6)\star c(i_7)^c\star a(i_8)^c\star c(i_9))\ f_{i_{11}i_6i_7}\ f_{i_{11}i_8i_9}\right)$$

If variables have not been explicitly declared as non-commutative, **CommutatorReduce** pulls them out of the non-commutative product **NM**.

```
In[298]:= % // CommutatorReduce
```

$$\text{Out[298]=}\ k\left(a(i_2)\ b(i_5)\ a(i_4)^c\ b(i_3)^c\ f_{i_{10}i_2i_3}\ f_{i_{10}i_4i_5}+a(i_6)\ c(i_9)\ a(i_8)^c\ c(i_7)^c\ f_{i_{11}i_6i_7}\ f_{i_{11}i_8i_9}\right)$$

IsoIndicesSupply increments the index number. To have indices renamed in a unique and minimal way, use **IndicesCleanup**.

```
In[299]:= % // IndicesCleanup
```

$$\text{Out[299]=}\ -k\ a(k_1)\ b(k_2)\ a(k_5)^c\ b(k_4)^c\ f_{k_1k_3k_4}\ f_{k_2k_3k_5}-k\ a(k_1)\ c(k_2)\ a(k_5)^c\ c(k_4)^c\ f_{k_1k_3k_4}\ f_{k_2k_3k_5}$$

By default, **SUNReduce** just applies its reduction rules once.

```
In[300]:= % // SUNReduce // Simplify
```

$$\text{Out[300]=}\ k\,a(k_1)\,a(k_5)^c\,(b(k_2)\,b(k_4)^c+c(k_2)\,c(k_4)^c)\left(\delta^{(2)}_{k_1k_5}\,\delta^{(2)}_{k_2k_4}-\delta^{(2)}_{k_1k_2}\,\delta^{(2)}_{k_4k_5}\right)$$

Tell **SUNReduce** to fully reduce the expression.

```
In[301]:= SUNReduce[%, FullReduce → True] //
          Simplify
```

$$\text{Out[301]=}\ k\,a(k_1)\,(b(k_2)\,a(k_1)^c\,b(k_2)^c-b(k_1)\,a(k_4)^c\,b(k_4)^c+c(k_2)\,a(k_1)^c\,c(k_2)^c-c(k_1)\,a(k_4)^c\,c(k_4)^c)$$

Now, **IndicesCleanup** can bring some further simplification.

```
In[302]:= % // IndicesCleanup // Simplify
```

$$\text{Out[302]=}\ k\,a(k_1)\,(b(k_1)\,(a(k_1)^c\,b(k_1)^c-a(k_4)^c\,b(k_4)^c)+c(k_1)\,(a(k_1)^c\,c(k_1)^c-a(k_4)^c\,c(k_4)^c))$$

Clean up.

```
In[303]:= Clear[a,b,c,d];
```

DeclareUMatrix[*a***]**	causes the symbol *a* to be treated as a flavor SU(*N*) matrix
DeclareUScalar[*a***]**	causes the symbol *a* to be treated as a flavor SU(*N*) scalar
UndeclareUMatrix[*a***]**	causes the symbol *a* to seize being treated as a flavor SU(*N*) matrix
UndeclareUScalar[*a***]**	causes the symbol *a* to seize being treated as a flavor SU(*N*) scalar

Matrices and scalars of SU(2) or SU(3).

3.5 Model and Lagrangian Definitions

In ChPT one finds a number of symbols representing matrices of fields with various transformation properties under SU(*N*)×SU(*N*). Some of these symbols are common to all ChPT models and are provided by the package PHI itself, others are defined in the model files and are only available upon loading the right configuration.

The constants of the various models usually have head **ParticleMass**, **CouplingConstant** or **DecayConstant**.

ParticleMass[*p***]**	mass of the particle *p*
CouplingConstant[*c***]**	some coupling constant *c*
DecayConstant[*p***]**	decay constant of the particle *p*

Heads for physical constants.

As we have seen, the convention used for the symbols representing quantum fields used in ChPT is to give them the head **Particle**. This is in order to be able to have an extra argument specifying whether a quantity is renormalized or not.

Particle[*p***]**	represents a particle or source *p* should be a member of the list **$Particles**
$Particles	a list of available particles and sources of the active configuration (**$Configuration**)

Particle names.

RenormalizationState[*i***]**	tag a quantity as unrenormalized (*i*=0) or renormalized (*i*=1)

Option of **Particle**.

A number of matrices and constants, commonly used in ChPT are predefined by the main package PHI and listed below. Others are defined by the individual model definition files. As always, to get information about these, use the **?** operator.

UQuarkMass[]	**UMatrix[UQuarkMass[]]** is the quark mass matrix
UChi[]	**UMatrix[UChi[]]** is the χ of ChPT
MM[*x***]**	the U of ChPT. Notice that PHI displays this as $\mathcal{U}$
SMM[*x***]**	the u of ChPT ($u^2 = U$). Notice that PHI displays this as u
QuarkCondensate[]	the quark condensate $B_0 = \langle 0\|q\bar{q}\|0\rangle$

ChPT symbols available under all configurations.

The predefined matrices are typically used when building up the lagrangians corresponding to the model. The definition of a lagrangian should be in a file in the directory "HighEnergyPhysics/Phi/Lagrangians" the

name of which should indicate the model to which it belongs (defined by a file in the directory "HighEnergyPhysics/Phi/Configurations"). To keep the definition compact and to allow inspection of the unexpanded form, the convention is to not give the meson matrix U (**[**MM]) and other similar matrices any space-time argument nor any option specifying the expansion order or renormalization state. These arguments will be supplied automatically by the function **ArgumentsSupply**.

ArgumentsSupply[*exp***,** *x***,** *opts***]**	supply space-time argument *x* and options *opts* to matrix functions in *exp*

Writing out predefined lagrangians.

option name	*default value*	
ExpansionOrder	**4**	order of expansion in the meson field
DropOrder	**4**	power of the meson field, higher orders than which will be dropped

Controlling the expansion in the meson field.

Here is a piece of the leading order lagrangian. Notice that space-time arguments have not been supplied.

```
In[304]:= UTrace[NM[UChiMatrix, Adjoint[MM]]]
```

$$\text{Out[304]}= \langle \chi * \mathcal{U}^{\dagger} \rangle$$

If space-time arguments are supplied, the matrices are expanded.

```
In[305]:= UTrace[NM[UChiMatrix[x],
            Adjoint[MM[x]]]]
```

$$\text{Out[305]}= 2\,\mathcal{B}_0 \left\langle \left(\frac{\mathbb{d}\,(m_\pi)^2}{2\,\mathcal{B}_0} + \vec{s}\cdot\vec{\sigma} + \mathbb{d}\,s^0 \right) * \left(-\frac{i\,\vec{\pi}\cdot\vec{\sigma}}{f_\pi} - \frac{\vec{\pi}\cdot\vec{\sigma} * \vec{\pi}\cdot\vec{\sigma}}{2\,(f_\pi)^2} + \mathbb{d} \right) \right\rangle$$

Here is an example, involving also **IsoVector**, **UMatrix**. Notice that **IsoVector** takes over the x-dependence.

```
In[306]:= CovariantFieldDerivative[IsoDot[
          IsoVector[QuantumField[Particle[Pion]]][x],
          IsoVector[UMatrix[UGenerator[]]],x,
          LorentzIndex[μ]] // CommutatorReduce
```

$$\begin{aligned}\text{Out[306]}= \;& \partial_\mu\left(\vec{\pi}\right)\cdot\vec{\sigma} + \\ & i\left(\vec{\pi}\cdot\vec{\sigma} \ast \left(\vec{V}_\mu\cdot\vec{\sigma} - \vec{A}_\mu\cdot\vec{\sigma}\right)\right) - i\left(\left(\vec{V}_\mu\cdot\vec{\sigma} + \vec{A}_\mu\cdot\vec{\sigma}\right)\ast\vec{\pi}\cdot\vec{\sigma}\right) + \\ & i\left(\vec{\pi}\cdot\vec{\sigma}\ast Q_{\mathbf{L}}\right)\gamma_\mu - i\left(Q_{\mathbf{R}}\ast\vec{\pi}\cdot\vec{\sigma}\right)\gamma_\mu\end{aligned}$$

The previously inspected lagrangian may be expanded in the meson field. For readability we expand only to order 0.

```
In[307]:= ArgumentsSupply[Lagrangian[ChPTVirtualPhotons2[2]],
          x, RenormalizationState[0],
          ExpansionOrder → 0,
          DropOrder → 0]
```

$$\begin{aligned}\text{Out[307]}= \;& \tfrac{1}{4}\Big(\langle\Big(i\left(\vec{V}_\mu\cdot\vec{\sigma} - \vec{A}_\mu\cdot\vec{\sigma}\right) - i\left(\vec{A}_\mu\cdot\vec{\sigma} + \vec{V}_\mu\cdot\vec{\sigma}\right) + \\ & i\left(Q_L\ast\gamma_\mu\right) - i\left(Q_R\ast\gamma_\mu\right)\Big)\ast\Big(\\ & - i\left(\vec{V}_\mu\cdot\vec{\sigma} - \vec{A}_\mu\cdot\vec{\sigma}\right) + i\left(\vec{A}_\mu\cdot\vec{\sigma} + \vec{V}_\mu\cdot\vec{\sigma}\right) - \\ & i\left(\gamma_\mu\ast{Q_L}^\dagger\right) + i\left(\gamma_\mu\ast{Q_R}^\dagger\right)\Big)\rangle + 4\,\mathcal{B}_0\left(\tfrac{(m_\pi)^2}{\mathcal{B}_0} + 2\,s^0\right)\Big)\,(f_\pi)^2 - \\ & \tfrac{1}{4}\left(\left(\partial_\mu\gamma_\nu - \partial_\nu\gamma_\mu\right)\ast\left(\partial_\mu\gamma_\nu - \partial_\nu\gamma_\mu\right)\right) + C\,\langle Q_R\ast Q_L\rangle - \\ & \tfrac{1}{2}\,\lambda\,\partial_\mu\gamma_\mu\,\partial_\nu\gamma_\nu\end{aligned}$$

3.6 Power Counting with FeynArts

To fully automatize the calculation of amplitudes, a way of generating Feynman diagrams and corresponding expressions is needed. The package FeynArts by H. Eck, J. Küblbeck and T. Hahn is a convenient choice, since it is written in Mathematica and is general enough to allow the definition of ChPT models and any number of legs in the coupling vertices. FeynArts is documented in [24].

Two features of ChPT complicates the application of FeynArts: 1) The expansion is not just in a coupling constant, but in meson masses and momenta and typically also in a coupling constant. This makes the "generic" and "classes" coupling concept of FeynArts a complicating irrelevance. 2) The coupling expressions can be very large. Therefore, the FeynArts convention of having all coupling definitions in one file is not practical.

The strategy chosen to deal with these complications is as follows:

- The functions **MomentaCollect**, **GenericCoupling** and **ClassesCoupling** are provided, to facilitate the splitting of Feynman rules generated with **FeynRule** (see section 4.5) into "generic" and "classes" coupling "vectors" understood by FeynArts.
- For a calculation to order p^{2n}, each vertex will in general have contributions of order p^2, p^4, ... p^{2n}. Ignoring coupling constants for the moment, the generic vector must then have $2n$ entries. The classes "vector" consists of $2n$ $2n$-dimensional lists, where the $2n$th list will have zeros for all but the $2n$th entry, that is, the classes "vector" is a diagonal matrix. E.g. for $n = 2$, $\{\{a, 0\}, \{0, b\}\}$, where a and b are the $O(p^2)$ and $O(p^4)$ couplings.

- The generic and classes "vectors" are each stored in separate files with extension ".Gen" and ".Mod" respectively, in the directory "HighEnergyPhysics/Phi/CouplingVectors". A naming convention is followed for these files and a function, **XName**, is provided for generating the correct name. Storing and retrieving from the directory "HighEnergyPhysics/Phi/CouplingVectors" can then be done with the function **CheckDB**.
- The FeynArts model files "Automatic.gen" and "Automatic.mod" load the saved coupling "vectors" and generate correct FeynArts model definitions using the information of the active PHI model.

Generating and storing FeynArts coupling "vectors" usually follows a standard procedure; this is exemplified below for the $\gamma\gamma\pi^+\pi^-$ coupling.

The lagrangian is now expanded to second order in the pion field.

```
In[308]:= l = ArgumentsSupply[
              Lagrangian[ChPTVirtualPhotons2[2]],
              x, RenormalizationState[0],
              DiagonalToU → True,
              ExpansionOrder → 2,
              DropOrder → 2];
```

Terms of order in the pion field different from 2 are discarded. Default substitutions are applied, cyclicity of the trace is used, some expansion is done.

```
In[309]:= DiscardTerms[l,
          Retain →
          ParticleField[Pion ,
            RenormalizationState[0]] → 2,
          ParticleField[Photon,
            RenormalizationState[0]] → 2,
          CommutatorReduce → True,
          Method → Expand] /.
          $Substitutions // NMExpand //
          CycleUTraces // Expand
```

$$\text{Out[309]= } \tfrac{1}{8}(e)^2\langle\vec{\pi}\cdot\vec{\sigma}\ast\vec{\pi}\cdot\vec{\sigma}\ast\sigma^3\ast\sigma^3\rangle\gamma_\mu^2-\tfrac{1}{8}(e)^2\langle\vec{\pi}\cdot\vec{\sigma}\ast\sigma^3\ast\vec{\pi}\cdot\vec{\sigma}\ast\sigma^3\rangle\gamma_\mu^2$$

Monomials in $\vec{\pi}\cdot\vec{\sigma}$ are reduced to monomials of order 0 or 1, using the commutation and anti-commutation relations of the generator matrices.

```
In[310]:= ExpandU[%, CommutatorReduce → True] //
          Simplify
```

$$\text{Out[310]= } \tfrac{1}{4}(e)^2\left(-\left(\mathbb{p}(\vec{3})\cdot\vec{\pi}\right)^2+\mathbb{p}(\vec{3})\times\vec{\pi}\cdot\mathbb{p}(\vec{3})\times\vec{\pi}+\vec{\pi}\cdot\vec{\pi}\right)\gamma_\mu^2$$

The vector products are written in terms of contracted indices. Some simplification is done.

```
In[311]:= ll = % // IsoIndicesSupply //
          SUNReduce // IndicesCleanup //
          CommutatorReduce // Simplify
```

$$\text{Out[311]= } \tfrac{1}{4}(e)^2\,\pi^{k_1}\left(-\delta^{(2)}_{3k_1}\,\pi^3+\pi^{k_1}-\delta^{(2)}_{3k_3}\,f^{(2)}_{3k_1k_4}\,f^{(2)}_{k_2k_3k_4}\,\pi^{k_2}\right)\gamma_{\rho 1}^2$$

The fields of the vertex are defined.

```
In[312]:= fields = QuantumField[Particle[Pion,
          RenormalizationState[0]],
          SUNIndex[I1]][p1], QuantumField[Particle[Pion,
          RenormalizationState[0]],
          SUNIndex[I2]][p2], QuantumField[Particle[Photon,
          RenormalizationState[0]],
          LorentzIndex[μ3]][p3],
          QuantumField[Particle[Photon,
          RenormalizationState[0]],
          LorentzIndex[μ4]][p4]
```

$$\text{Out[312]} = \left\{\pi^{I_1}, \pi^{I_2}, \gamma_{\mu_3}, \gamma_{\mu_4}\right\}$$

The Feynman rule is calculated and reduced.

```
In[313]:= m = FeynRule[ll, fields] //
          SUNReduce[#, FullReduce → True] & //
          IndicesCleanup // CommutatorReduce //
          Simplify
```

$$\text{Out[313]} = -2\,i\,(e)^2\,g^{\mu_3\mu_4}\left(\delta^{(2)}_{3I_1}\,\delta^{(2)}_{3I_2} - \delta^{(2)}_{I_1I_2}\right)$$

The terms are collected according to powers of the chiral expansion parameters. In this particular case this step is redundant.

```
In[314]:= mfa = MomentaCollect[m // Expand,
          PerturbationOrder → 2]
```

$$\text{Out[314]} = (e)^2\,g^{\mu_3\mu_4}\left(2\,i\,\delta^{(2)}_{I_1I_2} - 2\,i\,\delta^{(2)}_{3I_1}\,\delta^{(2)}_{3I_2}\right)$$

The generic coupling vector is generated.

```
In[315]:= gencoup = GenericCoupling[mfa]
```

$$\text{Out[315]} = \left\{(e)^2\,g^{\mu_3\mu_4}\right\}$$

The classes coupling "vector" is generated.

```
In[316]:= classcoup = ClassesCoupling[mfa] //
          Together; classcoup // StandardForm
```

```
Out[316]= {{-2 i (SUNDelta[3, I1] SUNDelta[3, I2] - SUNDelta[I1, I2])}}
```

The generic coupling vector is saved.

```
In[317]:= CheckDB[gencoup, XName[PhiModel →
          ChPTVirtualPhotons2, VertexFields →
          Pion[0], Pion[0], Photon[0], Photon[0],
          PerturbationOrder → 2] <> ".Gen"];
```

The classes coupling "vector" is saved.

```
In[318]:= CheckDB[classcoup, XName[PhiModel →
          ChPTVirtualPhotons2, VertexFields →
          Pion[0], Pion[0], Photon[0], Photon[0],
          PerturbationOrder → 2] <> ".Mod"];
```

Clean up.

```
In[319]:= Clear[l,ll,m,mfa];
```

$Substitutions	a list of substitution rules, defined by the active PHI configuration file, used to write out predefined lagrangians beyond what is done by **ArgumentsSupply**

Expansion of predefined lagrangians.

When the $\gamma\gamma\pi^+\pi^-$ coupling has been calculated, transformed to the right format and stored in the right place, it is ready for use by FeynArts through the model files "Automatic.gen" and "Automatic.mod". Also, when PHI is loaded, the database of amplitudes accessible through the function **Amplitude** is enlarged with all amplitudes calculated and stored as above.

We may, for example, inspect the $\gamma\gamma\pi^+\pi^-$ coupling.

```
In[320]:= Amplitude["ChPTVirtualPhotons2P20P20V10V10o2"]
```

$$\text{Out[320]} = -2\,i\,(e)^2\,g^{\mu_3\mu_4}\,(\delta_{3I1}\,\delta_{3I2} - \delta_{I1I2})$$

MomentaCollect[*l***]**	collects terms of *l* according to their order in the perturbation expansion
GenericCoupling[*m***]**	constructs the kinematical coupling vector to be used in a generic model file for FeynArts from the matrix element *m*
ClassesCoupling[*l***]**	constructs the kinematical coupling "vector" to be used in a classes model file for FeynArts from the matrix element *m*
XName[*opts***]**	generates a name for a coupling file, using the options *opts*
CheckDB[*exp***, *fil*]**	checks if *fil* exists. If it does, gets *fil* and returns the loaded expressions. If *fil* does not exist, evaluates *exp*, saves it to *fil* and returns the evaluated *exp*

Utilities for generating and storing FeynArts coupling "vectors".

option name	*default value*	
ParticlesNumber	**4**	number of particles in the vertex
PerturbationOrder	**2**	order in the perturbation expansion
MomentumVariables String	**"p"**	base string of the momenta names, e.g. **p1**, **p2**, ...
VertexFields	**{Pion[0], Pion[0], Pion[0], Pion[0]}**	fields in the vertex
PhiModel	**ChPT2**	PHI model
Directory	**"Phi/Storage"**	storage directory
ForceSave	**False**	evaluate and save even if the file already exists

Options controlling the utilities for generating FeynArts coupling vectors.

3.7 Example: Radiative Pion Decay

$\pi^{\pm} \to \mu^{\pm} \nu_{\mu} \gamma$

Chapter 3

Reference Guide

Abbreviation

Description

Abbreviation is a function used by OneLoop and PaVeReduce for generating smaller files when saving results to the hard disk. The convention is that a definition like GP = GluonPropagator should be accompanied by the definition Abbreviation[GluonPropagator] = HoldForm[GP].

See also: $Abbreviations, OneLoop, PaVeReduce, WriteOut, WriteOutPaVe, GluonPropagator, GluonVertex, QuarkPropagator.

Examples

```
In[321]:= GP[p, {μ, a}, {ν, b}]
Out[321]= Π^{μν}_{ab}(p)
```

Amplitude

Description

Amplitude is a database of Feynman amplitudes. Amplitude["name"] returns the amplitude corresponding to the string "name". A list of all defined names is obtained with Amplitude[]. New amplitudes can be added to the file "Amplitude.m". It is strongly recommended to use names that reflect the process. The option Gauge → 1 means 't Hooft Feynman gauge; Polarization → 0 gives unpolarized OPE-type amplitudes, Polarization → 1 the polarized ones.

```
In[322]:= Options[Amplitude]
Out[322]= {Dimension → D, Gauge → 1, QuarkMass → 0, Polarization → 1}
```

See also: FeynAmp.

Examples

```
In[323]:= Amplitude[]//Length
Out[323]= 206
```

This is the amplitude of a gluon self-energy diagram.

```
In[324]:= Amplitude["se1g1"]
```

$$\text{Out[324]= } \Pi^{\alpha\rho}_{cd}(p-q)\,\Pi^{\beta\sigma}_{ef}(q)\,V^{\nu\rho\sigma}(-p,\,p-q,\,q)\,V^{\mu\alpha\beta}(p,\,q-p,\,-q)\,f_{ace}\,f_{bdf}$$

```
In[325]:= Explicit[%]
```

$$\text{Out[325]= } -\frac{1}{(p-q)^2\,q^2}(g^{\alpha\rho}\,g^{\beta\sigma}\,(-g_s\,p^\alpha\,g^{\beta\mu}-g_s\,q^\alpha\,g^{\beta\mu}+2\,g_s\,g^{\alpha\mu}\,p^\beta-g_s\,g^{\alpha\mu}\,q^\beta-g_s\,g^{\alpha\beta}\,p^\mu+2\,g_s\,g^{\alpha\beta}\,q^\mu)$$
$$(g_s\,p^\nu\,g^{\rho\sigma}-2\,g_s\,q^\nu\,g^{\rho\sigma}+g_s\,g^{\nu\sigma}\,p^\rho+g_s\,g^{\nu\sigma}\,q^\rho-2\,g_s\,g^{\nu\rho}\,p^\sigma+g_s\,g^{\nu\rho}\,q^\sigma)\,\delta_{cd}\,\delta_{ef}\,f_{ace}\,f_{bdf})$$

This is the amplitude for graph 6.2 from the paper Z.Phys C **70:**637-654, 1996**.**

```
In[326]:= FeynAmp[q1, q2, EpsEvaluate[Trick[Explicit[Amplitude["gg2"]]]]]
```

$$\text{Out[326]= } \int d^Dq_1\int d^Dq_2\,(\frac{1}{q_1^2\cdot q_2^2\cdot q_2^2\cdot(q_2-p)^2\cdot(q_1-q_2)^2\cdot(q_1-p)^2}$$
$$\left(2\,i\,\epsilon^{\lambda_1\lambda_5\Delta q_2}\left(-g_s\,p^\nu\,g^{\lambda_7\lambda_{10}}+2\,g_s\,q_1^\nu\,g^{\lambda_7\lambda_{10}}+2\,g_s\,g^{\nu\lambda_{10}}\,p^{\lambda_7}-g_s\,g^{\nu\lambda_{10}}\,q_1^{\lambda_7}-g_s\,g^{\nu\lambda_7}\,p^{\lambda_{10}}-g_s\,g^{\nu\lambda_7}\,q_1^{\lambda_{10}}\right)\right.$$
$$(g_s\,p^\mu\,g^{\lambda_1\lambda_{11}}-2\,g_s\,q_2^\mu\,g^{\lambda_1\lambda_{11}}-2\,g_s\,g^{\mu\lambda_{11}}\,p^{\lambda_1}+g_s\,g^{\mu\lambda_{11}}\,q_2^{\lambda_1}+g_s\,g^{\mu\lambda_1}\,p^{\lambda_{11}}+g_s\,g^{\mu\lambda_1}\,q_2^{\lambda_{11}})$$
$$\left(-2\,g_s\,q_1^{\lambda_5}\,g^{\lambda_7\lambda_{12}}+g_s\,q_2^{\lambda_5}\,g^{\lambda_7\lambda_{12}}+g_s\,g^{\lambda_5\lambda_{12}}\,q_1^{\lambda_7}-2\,g_s\,g^{\lambda_5\lambda_{12}}\,q_2^{\lambda_7}+g_s\,g^{\lambda_5\lambda_7}\,q_1^{\lambda_{12}}+g_s\,g^{\lambda_5\lambda_7}\,q_2^{\lambda_{12}}\right)$$
$$\left(-g_s\,p^{\lambda_{10}}\,g^{\lambda_{11}\lambda_{12}}-g_s\,q_1^{\lambda_{10}}\,g^{\lambda_{11}\lambda_{12}}+2\,g_s\,q_2^{\lambda_{10}}\,g^{\lambda_{11}\lambda_{12}}-g_s\,g^{\lambda_{10}\lambda_{12}}\,p^{\lambda_{11}}+2\,g_s\,g^{\lambda_{10}\lambda_{12}}\,q_1^{\lambda_{11}}-g_s\,g^{\lambda_{10}\lambda_{12}}\,q_2^{\lambda_{11}}+2\,g_s\right.$$
$$\left.\left.g^{\lambda_{10}\lambda_{11}}\,p^{\lambda_{12}}-g_s\,g^{\lambda_{10}\lambda_{11}}\,q_1^{\lambda_{12}}-g_s\,g^{\lambda_{10}\lambda_{11}}\,q_2^{\lambda_{12}}\right)(1-(-1)^m)\,(\Delta\cdot q_2)^{m-1}\,f_{ac_5c_{11}}\,f_{bc_7c_{10}}\,f_{c_5c_7c_{12}}\,f_{c_{10}c_{11}c_{12}}\right))$$

Amputate

Description

Amputate[exp,q1,q2,...] amputates Eps and DiracGamma. Amputate[exp,q1,q2,Pair→{p}] amputates also p.q1 and p.q2; Pair→All amputates all except OPEDelta.

```
In[327]:= Options[Amputate]
Out[327]= {Dimension → D, Pair → {}, Unique → True}
```

See also: DiracGamma, DiracMatrix, DiracSimplify, DiracSlash, DiracTrick.

Examples

```
In[328]:= DiracSlash[p].DiracSlash[q]
```

$$\text{Out[328]= } (\gamma\cdot p).(\gamma\cdot q)$$

```
In[329]:= Amputate[%, q]
```

$$\text{Out[329]= } (\gamma\cdot p).\gamma^{\$AL\$27(1)}\,q^{\$AL\$27(1)}$$

AnomalousDimension

Description

AnomalousDimension[name] is a database of anomalous dimensions of twist 2 operators.

```
In[330]:= Options[AnomalousDimension]
Out[330]= {Polarization → 1, Simplify → FullSimplify}
```

See also: SplittingFunction, SumS, SumT.

Examples

Polarized case

```
In[331]:= SetOptions[ AnomalousDimension, Polarization → 1]
Out[331]= {Polarization → 1, Simplify → FullSimplify}
```

$\gamma^{(0)}_{NS,qq}$ polarized

```
In[332]:= AnomalousDimension[gnsqq0]
```

$$\text{Out}[332]= C_F \left(8 S_1(m-1) + \frac{4}{m} + \frac{4}{m+1} - 6\right)$$

$\gamma^{(0)}_{S,qg}$ polarized

```
In[333]:= AnomalousDimension[gsqg0]
```

$$\text{Out}[333]= \left(\frac{8}{m} - \frac{16}{m+1}\right) T_f$$

$\gamma^{(0)}_{S,gq}$ polarized

```
In[334]:= AnomalousDimension[gsgq0]
```

$$\text{Out}[334]= C_F \left(\frac{4}{m+1} - \frac{8}{m}\right)$$

$\gamma^{(0)}_{S,gg}$ polarized

```
In[335]:= AnomalousDimension[gsgg0]
```

$$\text{Out}[335]= \frac{8 T_f}{3} + C_A \left(8 S_1(m-1) - \frac{8}{m} + \frac{16}{m+1} - \frac{22}{3}\right)$$

$\gamma^{(0)}_{PS,qq}$ polarized

```
In[336]:= AnomalousDimension[gpsqq1]
```

$$\text{Out}[336]= 16 C_F \left(\frac{1}{m+1} + \frac{3}{(m+1)^2} + \frac{2}{(m+1)^3} - \frac{1}{m} - \frac{1}{m^2} + \frac{2}{m^3}\right) T_f$$

$\gamma^{(1)}_{NS,qq}$ polarized

```
In[337]:= AnomalousDimension[gnsqq1]
```

```
Out[337]= -( 16 S1(m-1)/m^2 + 16 S1(m-1)/(m+1)^2 + 16 S2(m-1)/m + 16 S2(m-1)/(m+1) -
      24 S2(m-1) + 32 S12(m-1) + 32 S21(m-1) + 32 (S~)2(m-1)/m + 32 (S~)2(m-1)/(m+1) -
      32 (S~)3(m-1) + 64 (S~)12(m-1) - 40/m + 40/(m+1) + 16/(m+1)^2 + 8/m^3 + 40/(m+1)^3 + 3 ) CF^2 -
   Nf ( 80/9 S1(m-1) - 16/3 S2(m-1) - 8/(9 m) + 88/(9 (m+1)) - 8/(3 m^2) - 8/(3 (m+1)^2) - 2/3 ) CF -
   CA ( -536/9 S1(m-1) + 88/3 S2(m-1) - 16 S3(m-1) - 16 (S~)2(m-1)/m - 16 (S~)2(m-1)/(m+1) +
      16 (S~)3(m-1) - 32 (S~)12(m-1) + 212/(9 m) - 748/(9 (m+1)) + 44/(3 m^2) - 4/(3 (m+1)^2) - 16/(m+1)^3 + 17/3 ) CF
```

$\gamma_{S,qg}^{(1)}$ polarized

```
In[338]:= AnomalousDimension[gsqg1]
Out[338]= 8 CF Tf ( 2 S1^2(m-1)/m - 4 S1^2(m-1)/(m+1) - 2 S2(m-1)/m +
      4 S2(m-1)/(m+1) + 14/m - 19/(m+1) - 1/m^2 - 8/(m+1)^2 - 2/m^3 + 4/(m+1)^3 ) +
   16 CA Tf ( -S1^2(m-1)/m + 2 S1^2(m-1)/(m+1) - 2 S1(m-1)/m^2 + 4 S1(m-1)/(m+1)^2 - S2(m-1)/m + 2 S2(m-1)/(m+1) -
      2 (S~)2(m-1)/m + 4 (S~)2(m-1)/(m+1) - 4/m + 3/(m+1) - 3/m^2 + 8/(m+1)^2 + 2/m^3 + 12/(m+1)^3 )
```

$\gamma_{S,gq}^{(1)}$ polarized

```
In[339]:= AnomalousDimension[gsgq1]
Out[339]= 4 ( 4 S1^2(m-1)/m - 2 S1^2(m-1)/(m+1) - 8 S1(m-1)/m + 2 S1(m-1)/(m+1) + 8 S1(m-1)/m^2 - 4 S1(m-1)/(m+1)^2 +
      4 S2(m-1)/m - 2 S2(m-1)/(m+1) + 15/m - 6/(m+1) - 12/m^2 + 3/(m+1)^2 + 4/m^3 - 2/(m+1)^3 ) CF^2 +
   32 Tf ( -2 S1(m-1)/(3 m) + S1(m-1)/(3 (m+1)) + 7/(9 m) - 2/(9 (m+1)) - 2/(3 m^2) + 1/(3 (m+1)^2) ) CF +
   8 CA ( -2 S1^2(m-1)/m + S1^2(m-1)/(m+1) + 16 S1(m-1)/(3 m) - 5 S1(m-1)/(3 (m+1)) + 2 S2(m-1)/m - S2(m-1)/(m+1) +
      4 (S~)2(m-1)/m - 2 (S~)2(m-1)/(m+1) - 56/(9 m) - 20/(9 (m+1)) + 28/(3 m^2) - 38/(3 (m+1)^2) - 4/m^3 - 6/(m+1)^3 ) CF
```

$\gamma_{S,gg}^{(1)}$ polarized

```
In[340]:= AnomalousDimension[gsgg1]
```

```
Out[340]= 4 ( 8 S1(m-1)/m^2 - 16 S1(m-1)/(m+1)^2 + 134/9 S1(m-1) + 8 S2(m-1)/m - 16 S2(m-1)/(m+1) +
      4 S3(m-1) - 8 S12(m-1) - 8 S21(m-1) + 8 (S~)2(m-1)/m - 16 (S~)2(m-1)/(m+1) + 4 (S~)3(m-1) -
      8 (S~)12(m-1) - 107/(9 m) + 241/(9 (m+1)) + 58/(3 m^2) - 86/(3 (m+1)^2) - 8/m^3 - 48/(m+1)^3 - 16/3 ) CA^2 +
   32 Tf ( - 5/9 S1(m-1) + 14/(9 m) - 19/(9 (m+1)) - 1/(3 m^2) - 1/(3 (m+1)^2) + 1/3 ) CA +
   8 CF ( - 10/(m+1) + 2/(m+1)^2 + 4/(m+1)^3 + 1 + 10/m - 10/m^2 + 4/m^3 ) Tf
```

$\gamma_{S,gg}^{(1)}$ polarized (different representation)

```
In[341]:= AnomalousDimension[GSGG1]
Out[341]= 4 ( - (m (m (m (m (48 m (m+3) + 469) + 698) + 7) + 258) + 144)/(9 m^3 (m+1)^3) + 8 S2'(m/2)/(m (m+1)) -
      S3'(m/2) + (2 (m (67 m (m+1)^2 + 144) + 72) S1(m))/(9 m^2 (m+1)^2) - 4 S2'(m/2) S1(m) + 8 S~(m) ) CA^2 +
   32 Tf ( m (m+1) (3 m (m+1) + 13) - 3)/(9 m^2 (m+1)^2) - 5 S1(m)/9 ) CA + 8 CF (m (m (m (m (m (m+3) + 5) + 1) - 8) + 2) + 4) Tf/(m^3 (m+1)^3)
```

Check that all odd moments give the same for the two representations of $\gamma_{S,gg}^{(1)}$.

```
In[342]:= Table[% - %%/.OPEm → ij, {ij, 1, 17, 2}]
Out[342]= {0, 0, 0, 0, 0, 0, 0, 0, 0}
```

AntiCommutator

Description

AntiCommutator[x, y] = c defines the anti-commutator of the non commuting objects x and y.
See also: Commutator, CommutatorExplicit, DeclareNonCommutative, DotSimplify.

Examples

This declares a and b as noncommutative variables.

```
In[343]:= DeclareNonCommutative[a, b]

In[344]:= AntiCommutator[a, b]
Out[344]= {a, b}

In[345]:= CommutatorExplicit[%]
Out[345]= a.b + b.a
```

```
In[346]:= CommutatorExplicit[AntiCommutator[a + b, a - 2b ]]
Out[346]= (a - 2 b).(a + b) + (a + b).(a - 2 b)

In[347]:= DotSimplify[AntiCommutator[a + b, a - 2b ]]
Out[347]= 2 a.a - a.b - b.a - 4 b.b

In[348]:= DeclareNonCommutative[c, d, ~c, ~d]
```

Defining {c,d} = z results in replacements of c.d by z-d.c.

```
In[349]:= AntiCommutator[c, d] = z
Out[349]= z

In[350]:= DotSimplify[ d . c . d ]
Out[350]= d z - d.d.c

In[351]:= AntiCommutator[~d, ~c] = ~z
Out[351]= ~z

In[352]:= DotSimplify[ ~d . ~c . ~d ]
Out[352]= ~d ~z - ~c.~d.~d

In[353]:= UnDeclareNonCommutative[a, b, c, d, ~c, ~d]

In[354]:= Unset[AntiCommutator[c, d]]

In[355]:= Unset[AntiCommutator[~d, ~c]]
```

AntiQuarkField

Description

AntiQuarkField is the name of a fermionic field. AntiQuarkField is just a name with no functional properties. Only typeset rules are attached.

See also: QuantumField, QuarkField.

Examples

```
In[356]:= AntiQuarkField
Out[356]= $\overline{\psi}$
```

FCAntiSymmetrize

Description

FCAntiSymmetrize[expr, {a1, a2, ...}] antisymmetrizes expr with respect to the variables a1,a2, ...
See also: FCSymmetrize.

Examples

```
In[357]:= FCAntiSymmetrize[f[a, b], {a, b}]
Out[357]= 1/2 (f(a, b) - f(b, a))

In[358]:= FCAntiSymmetrize[f[x, y, z], {x, y, z}]
Out[358]= 1/6 (f(x, y, z) - f(x, z, y) - f(y, x, z) + f(y, z, x) + f(z, x, y) - f(z, y, x))
```

Anti5

Description

Anti5[exp] anticommutes all γ^5in exp to the right. Anti5[exp, n] anticommutes all γ^5 n times to the right. Anti5[exp, -n] anticommutes all γ^5 n times to the left.
The naive γ^5scheme is used.
See also: DiracOrder, DiracSimplify, DiracTrick.

Examples

```
In[359]:= DiracMatrix[5, μ]
Out[359]= γ⁵γ^μ

In[360]:= Anti5[%]
Out[360]= -γ^μ.γ⁵

In[361]:= Anti5[%, -1]
Out[361]= γ⁵.γ^μ

In[362]:= DiracMatrix[5, α, β, γ, δ]
Out[362]= γ⁵γ^αγ^βγ^γγ^δ

In[363]:= Anti5[%, 2]
Out[363]= γ^α.γ^β.γ⁵.γ^γ.γ^δ

In[364]:= Anti5[%%, ∞]
Out[364]= γ^α.γ^β.γ^γ.γ^δ.γ⁵

In[365]:= Anti5[%, -∞]
```

```
Out[365]= γ⁵.γ^α.γ^β.γ^γ.γ^δ
```

In the naive γ^5- scheme D-dimensional γ-matrices anticommute with γ^5.

```
In[366]:= GAD[5, μ]
Out[366]= γ⁵.γ^μ

In[367]:= Anti5[%]
Out[367]= -γ^μ.γ⁵
```

Apart2

Description

Apart2[expr] partial fractions FeynAmpDenominators (and FAD's).
See also: FAD, FeynAmpDenominator.

Examples

```
In[368]:= FAD[{q, m}, {q, M}, q - p]
                                    1
Out[368]= ------------------------------------------------
          ([(q - p)^2]) ([q^2 - m^2]) ([q^2 - M^2])

In[369]:= Apart2[%]
          1                  1
          ---------------- - ----------------
          (q^2-m^2).(q-p)^2  (q^2-M^2).(q-p)^2
Out[369]= ----------------------------------
                        2    2
                       m  - M

In[370]:= StandardForm[FCE[%]]
          FAD[{q, m}, -p + q] - FAD[{q, M}, -p + q]
Out[370]= -----------------------------------------
                          2    2
                         m  - M
```

A0

Description

A0[m^2] isthePassarino−

Veltmanone − point integral A_0.

```
In[371]:= Options[A0]
Out[371]= {A0ToB0 → False}
```

See also: B0, C0, D0, PaVe.

Examples

By default A_0 is not expressed in terms of B_0.

```
In[372]:= A0[m^2]
Out[372]= A0(m^2)

In[373]:= SetOptions[A0, A0ToB0 → True];

In[374]:= A0[m^2]
Out[374]= B0(0, m^2, m^2) m^2 + m^2

In[375]:= SetOptions[A0, A0ToB0 → False];
```

According to the rules of dimensional regularization $A_0(0)$ is set to 0.

```
In[376]:= A0[0]
Out[376]= 0

In[377]:= A0[SmallVariable[M^2]]
Out[377]= 0
```

A0ToB0

Description

A0ToB0 is an option for A0. If set to True, A0[m^2] is expressed by (1 + B0[0, m^2, m^2]) m^2.

```
In[378]:= Options[A0]
Out[378]= {A0ToB0 → False}
```

See also: A0, B0, C0, D0, PaVe.

BackgroundGluonVertex

Description

BackgroundGluonVertex[{p,mu,a}, {q,nu,b}, {k,la,c}] or BackgroundGluonVertex[p,mu,a , q,nu,b , k,la,c] yields the 3-gluon vertex in the background field gauge, where the first set of arguments corresponds to the external background field. BackgroundGluonVertex[{p,mu,a}, {q,nu,b}, {k,la,c}, {s,si,d}] or BackgroundGluonVertex[{mu,a}, {nu,b}, {la,c}, {si,d}] or BackgroundGluonVertex[p,mu,a , q,nu,b , k,la,c , s,si,d] or BackgroundGluonVertex[mu,a , nu,b , la,c , si,d] yields the 4-gluon vertex, with {p,mu,a} and {k,la,c} denoting the external background fields. The gauge, dimension and the name of the coupling constant are determined by the options Gauge, Dimension and CouplingConstant. The Feynman rules are taken from L.Abbot **NPB** 185 (1981), 189-203; except that all momenta are incoming. Note that Abbots coupling constant convention is consistent with the default setting of GluonVertex.

```
In[379]:= Options[BackgroundGluonVertex]
```

```
Out[379]= {Dimension → D, CouplingConstant → g_s, Gauge → 1}
```

See also: GluonVertex.

Examples

```
In[380]:= BackgroundGluonVertex[{p, μ, a}, {q, ν, b}, {k, λ, c}]
Out[380]= g_s ((-k + p - q)^λ g^μν + g^λν (q - k)^μ + g^λμ (k - p + q)^ν) f_abc

In[381]:= BackgroundGluonVertex[{p, μ, a}, {q, ν, b}, {k, λ, c}, {s, σ, d}]
Out[381]= -i g_s^2 ((g^λσ g^μν - g^λν g^μσ - g^λμ g^νσ) f_adu6 f_bcu6 +
                    (g^λσ g^μν - g^λν g^μσ) f_acu6 f_bdu6 + (g^λσ g^μν - g^λν g^μσ + g^λμ g^νσ) f_abu6 f_cdu6)

In[382]:= BackgroundGluonVertex[{p, μ, a}, {q, ν, b}, {k, λ, c}, Gauge → α]
Out[382]= g_s ((p - q - k/α)^λ g^μν + g^λν (q - k)^μ + g^λμ (k - p + q/α)^ν) f_abc

In[383]:= BackgroundGluonVertex[{p, μ, a}, {q, ν, b}, {k, λ, c}, {s, σ, d}, Gauge → α]
Out[383]= -i g_s^2 ((g^λσ g^μν - g^λν g^μσ/α - g^λμ g^νσ) f_adu7 f_bcu7 +
                    (g^λσ g^μν - g^λν g^μσ) f_acu7 f_bdu7 + (g^λσ g^μν/α - g^λν g^μσ + g^λμ g^νσ) f_abu7 f_cdu7)
```

Bracket

Description

Bracket is an option of Convolute.

See also: Convolute.

BReduce

Description

BReduce is an option for B0, B00, B1, B11 determining whether reductions to A0 and B0 will be done.

See also: A0, B0, B00, B1, B11.

Examples

By default B_0 is not expressed in terms of A_0.

```
In[384]:= B0[0, s, s]
Out[384]= B_0(0, s, s)

In[385]:= B0//Options
```

```
Out[385]= {BReduce → False, B0Unique → True, B0Real → False}
```

With BReduce→True, transformation is done.

```
In[386]:= B0[0, s, s, BReduce → True]
Out[386]= A0(s)/s - 1
```

B0

Description

B0[pp, ma^2, mb^2] is the Passarino-Veltman two-point integral B_0. All arguments are scalars and have dimension mass^2. If the option BReduce is set to True certain B0's are reduced to A0's. Setting the option B0Unique to True simplifies B0[a,0,a] and B0[0,0,a].

```
In[387]:= Options[B0]
Out[387]= {BReduce → False, B0Unique → True, B0Real → False}
```

See also: B1, B00, B11, PaVe.

Examples

```
In[388]:= B0[SP[p, p], m^2, m^2]
Out[388]= B0(p^2, m^2, m^2)

In[389]:= SetOptions[B0, B0Unique → True, B0Real → True];

In[390]:= B0[0, 0, m^2]
Out[390]= B0(0, m^2, m^2) + 1

In[391]:= B0[m^2, 0, m^2]
Out[391]= B0(0, m^2, m^2) + 2

In[392]:= B0[0, m^2, m^2]
Out[392]= B0(0, m^2, m^2)
```

B0Real

Description

B0Real is an option of B0 (default False). If set to True, B0 is assumed to be real and the relation B0[a,0,a] = 2 + B0[0,a,a] is applied.

See also: B0.

Examples

By default the arguments are not assumed real.

```
In[393]:= B0[s, 0, s]
Out[393]= B0(0, s, s) + 2
```

```
In[394]:= B0//Options
Out[394]= {BReduce → False, B0Unique → True, B0Real → True}
```

With B0Real→True, transformation is done.

```
In[395]:= B0[s, 0, s, B0Real → True]
Out[395]= B0(0, s, s) + 2
```

B0Unique

Description

B0Unique is an option of B0. If set to True, B0[0,0,m2] is replaced with (B0[0,m2,m2]+1) and B0[m2,0,m2] simplifies to (B0[0,m2,m2]+2).
See also: B0.

Examples

By default transformation is done.

```
In[396]:= B0[0, 0, s]
Out[396]= B0(0, s, s) + 1
```

```
In[397]:= B0//Options
Out[397]= {BReduce → False, B0Unique → True, B0Real → True}
```

With B0Real→False, nothing happens.

```
In[398]:= B0[0, 0, s, B0Unique → False]
Out[398]= B0(0, 0, s)
```

B00

Description

B00[pp, ma^2,mb^2] is the Passarino-Veltman B_0-function, i.e., the coefficient function of the metric tensor. All arguments are scalars and have dimension mass^2.

```
In[399]:= Options[B00]
```

```
Out[399]= {BReduce → True}
```

See also: B0, B1, PaVe.

Examples

Remember that SP[p] is a short hand input for ScalarProduct[p,p], i.e., p^2.

```
In[400]:= SP[p]
```

$$\text{Out[400]= } p^2$$

```
In[401]:= B00[SP[p], m^2, M^2]
```

$$\text{Out[401]= } \frac{1}{3} B_0(p^2, m^2, M^2)\, m^2 + \frac{1}{18}(3m^2 + 3M^2 - p^2) + \frac{1}{6}\left(A_0(M^2) + \left(\frac{(M^2 - m^2)(B_0(p^2, m^2, M^2) - B_0(0, m^2, M^2))}{2p^2} - \frac{1}{2} B_0(p^2, m^2, M^2)\right)(m^2 - M^2 + p^2)\right)$$

```
In[402]:= B00[SP[p], m^2, m^2]
```

$$\text{Out[402]= } \frac{1}{3} B_0(p^2, m^2, m^2)\, m^2 + \frac{1}{18}(6m^2 - p^2) + \frac{1}{6}\left(A_0(m^2) - \frac{1}{2} B_0(p^2, m^2, m^2)\, p^2\right)$$

```
In[403]:= B00[SP[p], m^2, M^2, BReduce → False]
```

$$\text{Out[403]= } B_0(p^2, m^2, M^2)$$

```
In[404]:= B00[0, m^2, m^2]
```

$$\text{Out[404]= } \frac{1}{3} B_0(0, m^2, m^2)\, m^2 + \frac{m^2}{3} + \frac{1}{6} A_0(m^2)$$

```
In[405]:= B00[SmallVariable[M^2], m^2, m^2]
```

$$\text{Out[405]= } \frac{1}{3} B_0(M^2, m^2, m^2)\, m^2 + \frac{m^2}{3} + \frac{1}{6} A_0(m^2)$$

B1

Description

B1[pp, maˆ2, mbˆ2] the Passarino-Veltman B_1-function. All arguments are scalars and have dimension massˆ2.

```
In[406]:= Options[B1]
Out[406]= {BReduce → True}
```

See also: B0, B00, B11, PaVe, PaVeReduce.

Examples

```
In[407]:= B1[SP[p], m^2, M^2]
```

$$\text{Out[407]}= \frac{(M^2 - m^2)\,(B_0(p^2, m^2, M^2) - B_0(0, m^2, M^2))}{2\,p^2} - \frac{1}{2} B_0(p^2, m^2, M^2)$$

```
In[408]:= B1[SP[p], m^2, M^2, BReduce → False]
```

$$\text{Out[408]}= B_1(p^2, m^2, M^2)$$

```
In[409]:= SetOptions[B1, BReduce → True];
```

```
In[410]:= B1[SP[p], m^2, m^2]
```

$$\text{Out[410]}= -\frac{1}{2} B_0(p^2, m^2, m^2)$$

```
In[411]:= B1[m^2, m^2, 0]
```

$$\text{Out[411]}= \frac{1}{2}\,(-B_0(0, m^2, m^2) - 2) - \frac{1}{2}$$

```
In[412]:= B1[0, 0, m^2]
```

$$\text{Out[412]}= \frac{1}{2}\,(-B_0(0, m^2, m^2) - 1) + \frac{1}{4}$$

```
In[413]:= B1[pp, SmallVariable[m_e^2], m_2^2]
```

$$\text{Out[413]}= \frac{(B_0(pp, m_e^2, m_2^2) - B_0(0, m_e^2, m_2^2))\,m_2^2}{2\,pp} - \frac{1}{2} B_0(pp, m_e^2, m_2^2)$$

```
In[414]:= B1[SmallVariable[m_e^2], SmallVariable[m_e^2], 0]
```

$$\text{Out[414]}= \frac{1}{2}\,(-B_0(0, m_e^2, m_e^2) - 2) - \frac{1}{2}$$

B11

Description

B11[pp, ma^2, mb^2] is the Passarino-Veltman B_{11}-function, i.e., the coefficient function of $p^\mu\, p^\nu$. All arguments are scalars and have dimension mass^2.

```
In[415]:= Options[B11]
```

$$\text{Out[415]}= \{\text{BReduce} \to \text{True}\}$$

See also: B0, B00, B1, PaVe.

Examples

Remember that SP[p] is a short hand input for ScalarProduct[p,p], i.e. p^2.

```
In[416]:= SP[p]
Out[416]= p^2

In[417]:= B11[SP[p], m^2, M^2]
Out[417]= 1/(3 p^2) ( - B0(p^2, m^2, M^2) m^2 + A0(M^2) + 1/6 (-3 m^2 - 3 M^2 + p^2) -
            2 ( (M^2 - m^2) (B0(p^2, m^2, M^2) - B0(0, m^2, M^2)) / (2 p^2) - 1/2 B0(p^2, m^2, M^2) ) (m^2 - M^2 + p^2) )

In[418]:= B11[SP[p], m^2, m^2]
Out[418]= (A0(m^2) + 1/6 (p^2 - 6 m^2) + B0(p^2, m^2, m^2) (p^2 - m^2)) / (3 p^2)

In[419]:= SetOptions[B11, BReduce -> False]
Out[419]= {BReduce -> False}

In[420]:= B11[SP[p], m^2, M^2]
Out[420]= B11(p^2, m^2, M^2)

In[421]:= SetOptions[B11, BReduce -> True]
Out[421]= {BReduce -> True}

In[422]:= B11[0, m^2, m^2]
Out[422]= 1/3 B0(0, m^2, m^2)

In[423]:= B11[SmallVariable[M^2], m^2, m^2]
Out[423]= 1/3 B0(M^2, m^2, m^2)
```

CA

Description

CA is one of the Casimir operator eigenvalues of SU(N) (CA = N).
See also: CF, SUNSimplify.

Examples

```
In[424]:= CA
Out[424]= CA

In[425]:= SUNSimplify[CA, SUNNToCACF -> False]
Out[425]= N
```

```
In[426]:= SUNN
Out[426]= N
```

Calc

Description

Calc[exp] performs several basic simplifications. Calc[exp] is the same as DotSimplify[DiracSimplify[Contract[DiracSimpli
SUNSimplify[Trick[exp], Explicit→False]]]]].
See also: DiracSimplify, DiracTrick, Trick.

Examples

This calculates $\gamma^\mu \gamma_\mu$ in 4 dimensions and g^ν_ν in D dimensions.

```
In[427]:= Calc[{GA[μ, μ], MTD[ν, ν]}]
Out[427]= {D, D}
```

This simplifies $f_{\text{abc}}\, f_{\text{abe}}$.

```
In[428]:= Calc[SUNF[a, b, c] SUNF[a, b, e]]
Out[428]= C_A δ_ce
```

```
In[429]:= FV[p + r, μ] MT[μ, ν] FV[q - p, ν]
Out[429]= (q - p^ν) (p + r^μ) g^μν
```

```
In[430]:= Calc[%]
Out[430]= -p^2 + p · q - p · r + q · r
```

```
In[431]:= GluonVertex[p, 1, q, 2, -p - q, 3]
Out[431]= V^li1li2li3(p, q, -p - q) f_ci1ci2ci3
```

```
In[432]:= Calc[% FVD[p, li1] FVD[q, li2] FVD[-p - q, li3]]
Out[432]= 0
```

CalculateCounterTerm ***unfinished***

Description

CalculateCounterTerm[exp, k] calculates the residue of exp.

Examples

```
In[433]:= ?CalculateCounterTerm
CalculateCounterTerm[exp, k] calculates the residue of exp.
```

CancelQP

Description

CancelQP is an option for OneLoop. If set to True, cancelation of all $q.p$'s and q^2 is performed.

Examples

```
In[434]:= t = SPD[p + 2q2, p + q2] FAD[{q2 - p, m}, {q2, 0}, {q2, M}]//FCI
```

$$\text{Out[434]}= \frac{(p+q_2)\cdot(p+2\,q_2)}{((q_2-p)^2-m^2).q_2^2.(q_2^2-M^2)}$$

```
In[435]:= OneLoop[q2, t, CancelQP → True]
```

$$\text{Out[435]}= i\,\pi^2\left(\frac{3\,B_0(p^2,0,m^2)\,m^2}{2\,M^2}-\frac{3\,A_0(M^2)}{2\,M^2}-\right.$$
$$\left.\frac{(3\,m^2-7\,M^2)\,B_0(p^2,m^2,M^2)}{2\,M^2}-\frac{5\,B_0(p^2,0,m^2)\,p^2}{2\,M^2}+\frac{5\,B_0(p^2,m^2,M^2)\,p^2}{2\,M^2}\right)$$

```
In[436]:= OneLoop[q2, t, CancelQP → False]
```

$$\text{Out[436]}= i\,\pi^2\left(-\frac{3\,B_0(0,m^2,m^2)\,m^2}{2\,M^2}+\frac{7\,B_0(0,m^2,M^2)\,m^2}{2\,M^2}+\frac{3\,B_0(p^2,0,m^2)\,m^2}{2\,M^2}-\frac{3\,m^2}{2\,M^2}-\frac{2\,A_0(m^2)}{M^2}+\frac{2\,A_0(M^2)}{M^2}-\right.$$
$$\left.\frac{7}{2}B_0(0,m^2,M^2)-\frac{(3\,m^2-7\,M^2)\,B_0(p^2,m^2,M^2)}{2\,M^2}-\frac{5\,B_0(p^2,0,m^2)\,p^2}{2\,M^2}+\frac{5\,B_0(p^2,m^2,M^2)\,p^2}{2\,M^2}\right)$$

```
In[437]:= t = SPD[q2, p]SPD[q1, p] FAD[{q1, m}, {q2, m}, q1 - p, q2 - p, q2 - q1]//FCI
```

$$\text{Out[437]}= \frac{p\cdot q_1\,p\cdot q_2}{(q_1^2-m^2).(q_2^2-m^2).(q_1-p)^2.(q_2-p)^2.(q_2-q_1)^2}$$

```
In[438]:= OneLoop[q2, t, CancelQP → True]
```

$$\text{Out[438]}= i\,\pi^2\left(-\frac{C_0(p^2,q_1^2,p^2-2\,p\cdot q_1+q_1^2,0,m^2,0)\,p\cdot q_1\,m^2}{2\,(m^2-q_1^2)\,(p^2-2\,p\cdot q_1+q_1^2)}+\frac{B_0(q_1^2,0,m^2)\,p\cdot q_1}{2\,(m^2-q_1^2)\,(p^2-2\,p\cdot q_1+q_1^2)}-\right.$$
$$\left.\frac{B_0(p^2-2\,p\cdot q_1+q_1^2,0,0)\,p\cdot q_1}{2\,(m^2-q_1^2)\,(p^2-2\,p\cdot q_1+q_1^2)}-\frac{C_0(p^2,q_1^2,p^2-2\,p\cdot q_1+q_1^2,0,m^2,0)\,p^2\,p\cdot q_1}{2\,(m^2-q_1^2)\,(p^2-2\,p\cdot q_1+q_1^2)}\right)$$

```
In[439]:= OneLoop[q2, t, CancelQP → False]
```

$$\text{Out[439]}= i\,\pi^2\left(\frac{PaVe(2,\{p^2,q_1^2,p^2-2\,p\cdot q_1+q_1^2\},\{0,m^2,0\})\,p\cdot q_1^2}{(m^2-q_1^2)\,(p^2-2\,p\cdot q_1+q_1^2)}-\right.$$
$$\frac{p^2\,PaVe(1,\{p^2,q_1^2,p^2-2\,p\cdot q_1+q_1^2\},\{0,m^2,0\})\,p\cdot q_1}{(m^2-q_1^2)\,(p^2-2\,p\cdot q_1+q_1^2)}-$$
$$\frac{p^2\,PaVe(2,\{p^2,q_1^2,p^2-2\,p\cdot q_1+q_1^2\},\{0,m^2,0\})\,p\cdot q_1}{(m^2-q_1^2)\,(p^2-2\,p\cdot q_1+q_1^2)}-$$
$$\left.\frac{C_0(p^2,q_1^2,p^2-2\,p\cdot q_1+q_1^2,0,m^2,0)\,p^2\,p\cdot q_1}{(m^2-q_1^2)\,(p^2-2\,p\cdot q_1+q_1^2)}\right)$$

```
In[440]:= Clear[t]
```

Cases2

Description

Cases2[expr, f] returns a list of all objects in expr with head f. Cases2[expr,f] is equivalent to Cases2[{expr},f[___],Infinity]//Union. Cases2[expr, f, g, ...] or Cases2[expr, {f,g, ...}] is equivalent to Cases[{expr},f[___] | g[___] ...] .

Examples

```
In[441]:= Cases2[f[a] + f[b]^2 + f[c, d], f]
Out[441]= {f(a), f(b), f(c, d)}

In[442]:= Cases2[Sin[x] Sin[y - z] + g[y], Sin, g]
Out[442]= {g(y), Sin(x), Sin(y - z)}

In[443]:= Cases2[Sin[x] Sin[y - z] + g[x] + g[a, b, c], {Sin, g}]
Out[443]= {g(x), g(a, b, c), Sin(x), Sin(y - z)}

In[444]:= Cases2[DiracSlash[p].DiracSlash[q] + ScalarProduct[p, p], Dot]
Out[444]= {(γ · p).(γ · q)}
```

CF

Description

CF is one of the Casimir operator eigenvalues of SU(*N*) (CF = (N^2-1)/(2 *N*)).
See also: CA, SUNSimplify.

Examples

```
In[445]:= CF
Out[445]= C_F

In[446]:= SUNSimplify[CF, SUNNToCACF → False]
Out[446]= (N^2 - 1)/(2 N)

In[447]:= %//InputForm
Out[447]= (-1 + SUNN^2)/(2 * SUNN)

In[448]:= SUNN
Out[448]= N

In[449]:= SUNSimplify[SUNN^2 - 1, SUNNToCACF → True]
Out[449]= 2 C_A C_F
```

ChangeDimension

Description

ChangeDimension[exp, dim] changes all LorentzIndex and Momenta in exp to dimension dim (and also Levi-Civita-tensors, Dirac slashes and Dirac matrices).

See also: LorentzIndex, Momentum, DiracGamma, Eps.

Examples

Remember that LorentzIndex[mu, 4] is simplified to LorentzIndex[mu] and Momentum[p, 4] to Momentum[p]. Thus the fullowing objects are defined in four dimensions.

```
In[450]:= {LorentzIndex[μ], Momentum[p]}
Out[450]= {μ, p}

In[451]:= ChangeDimension[%, D]
Out[451]= {μ, p}

In[452]:= %//StandardForm
Out[452]= {LorentzIndex[μ, D], Momentum[p, D]}
```

This changes all non-4-dimensional objects to 4-dimensional ones.

```
In[453]:= ChangeDimension[%%, 4] // StandardForm
Out[453]= {LorentzIndex[μ], Momentum[p]}
```

Consider the following list of 4- and D-dimensional object.

```
In[454]:= {GA[μ, ν] MT[μ, ν], GAD[μ, ν] MTD[μ, ν] f[D]}
Out[454]= {γ^μ.γ^ν g^μν, γ^μ.γ^ν f(D) g^μν}

In[455]:= DiracTrick[Contract[%]]
Out[455]= {4, D f(D)}

In[456]:= DiracTrick[Contract[ChangeDimension[%%, n]]]
Out[456]= {n, n f(D)}
```

Any explicit occurence of D (like in f(D)) is not replaced by ChangeDimension.

```
In[457]:= SetOptions[LeviCivita, Dimension → D];
```

The option Dimension of Eps must be changed too, since with the default setting Dimension→4 the arguments of Eps are automatically changed to 4 dimensions.

```
In[458]:= SetOptions[Eps, Dimension → D];

In[459]:= a1 = LeviCivita[μ, ν, ρ, σ]
Out[459]= ϵ^μνρσ

In[460]:= a2 = ChangeDimension[a1, 4]
```

```
Out[460]= ϵ^μνρσ

In[461]:= Factor2[Contract[a1^2]]
Out[461]= (1 - D) (2 - D) (3 - D) D

In[462]:= Contract[a2^2]
Out[462]= -24

In[463]:= SetOptions[Eps, Dimension → 4];

In[464]:= Clear[a1, a2];
```

ChargeConjugationMatrix

Description

ChargeConjugationMatrix denotes the charge conjugation matrix C.
See also: ChargeConjugationMatrixInv.

Examples

```
In[465]:= ChargeConjugationMatrix.DiracMatrix[μ].ChargeConjugationMatrixInv
Out[465]= C.γ^μ.(-C)

In[466]:= Calc[%]
Out[466]= -C.γ^μ.C

In[467]:= ChargeConjugationMatrix.DiracGamma[5].ChargeConjugationMatrixInv
Out[467]= C.γ^5.(-C)

In[468]:= Calc[%]
Out[468]= γ_5^T

In[469]:= ChargeConjugationMatrix^2
Out[469]= -1
```

ChargeConjugationMatrixInv

Description

ChargeConjugationMatrixInv is the inverse of ChargeConjugationMatrix. It is substituted immediately by -ChargeConjugationMatrix.
See also: ChargeConjugationMatrix.

Examples

```
In[470]:= ChargeConjugationMatrix.DiracMatrix[μ].ChargeConjugationMatrixInv
Out[470]= C.γ^μ.(-C)

In[471]:= Calc[%]
Out[471]= -C.γ^μ.C

In[472]:= ChargeConjugationMatrix.DiracGamma[5].ChargeConjugationMatrixInv
Out[472]= C.γ^5.(-C)

In[473]:= Calc[%]
Out[473]= γ5^T

In[474]:= ChargeConjugationMatrix.ChargeConjugationMatrixInv
Out[474]= C.(-C)

In[475]:= %//Calc
Out[475]= -C.C

In[476]:= ChargeConjugationMatrixInv^2
Out[476]= -1
```

CheckContext

Description

CheckContext[string] yields True if the packaged associated with string is already loaded, and False otherwise. See also: MakeContext, $FeynCalcStuff.

Examples

If a FeynCalc symbol has not been used, the corresponding subpackage has not been loaded:

```
In[477]:= CheckContext["CompleteSquare"]
Out[477]= False
```

Using it makes FeynCalc load the subpackage:

```
In[478]:= CompleteSquare
Out[478]= CompleteSquare

In[479]:= CheckContext["CompleteSquare"]
Out[479]= True
```

ChiralityProjector

Description

ChiralityProjector[+1] denotes 1/2(1 + γ^5). ChiralityProjector[-1] denotes 1/2(1 + γ^5).
See also: DiracGamma, DiracMatrix, FCI.

Examples

```
In[480]:= {ChiralityProjector[+1], ChiralityProjector[-1]}
Out[480]= {γ^6, γ^7}

In[481]:= FCI[%]
Out[481]= {γ^6, γ^7}

In[482]:= DiracSimplify[%, DiracSubstitute67 → True]
Out[482]= {γ^5/2 + 1/2, 1/2 - γ^5/2}
```

Chisholm

Description

Chisholm[x] substitutes products of three Dirac matrices or slashes by the Chisholm identity.
See also: EpsChisholm, ChisholmSpinor.

Examples

```
In[483]:= Chisholm[GA[α, β, μ, ν]]
Out[483]= -I γ^$MU$79 . γ^ν . γ^5 ε^αβμ$MU$79 + γ^μ . γ^ν g^αβ - γ^β . γ^ν g^αμ + γ^α . γ^ν g^βμ

In[484]:= t1 = DiracMatrix[μ, ν, ρ]
Out[484]= γ^μ γ^ν γ^ρ

In[485]:= t2 = Chisholm[t1]
Out[485]= I γ^$MU$84 . γ^5 ε^μνρ$MU$84 + γ^ρ g^μν - γ^ν g^μρ + γ^μ g^νρ
```

The MU variables are unique indices.

```
In[486]:= Calc[t1.t1]
Out[486]= -D^3 + 6 D^2 - 4 D

In[487]:= t3 = Chisholm[t1]
Out[487]= I γ^$MU$91 . γ^5 ε^μνρ$MU$91 + γ^ρ g^μν - γ^ν g^μρ + γ^μ g^νρ

In[488]:= Calc[t2.t3]
Out[488]= 3 D^2 - 2 D - 24
```

```
In[489]:= t4 = DiracSlash[a, b, c]
Out[489]= (γ · a).(γ · b).(γ · c)

In[490]:= Chisholm[t4]
Out[490]= -i γ$MU$100.γ^5 ϵ$MU$100abc + γ · c a · b - γ · b a · c + γ · a b · c

In[491]:= a1 = GA[μ, ν, ρ, σ, τ, κ]
Out[491]= γ^μ.γ^ν.γ^ρ.γ^σ.γ^τ.γ^κ

In[492]:= a2 = Chisholm[a1]
Out[492]= -i γ^τ.γ^κ.γ^5 ϵ^μνρσ + i γ^σ.γ^κ.γ^5 ϵ^μνρτ - i γ$MU$111.γ^κ.γ^5 ϵ^ρστ$MU$111 g^μν +
            i γ$MU$114.γ^κ.γ^5 ϵ^νστ$MU$114 g^μρ - i γ$MU$117.γ^κ.γ^5 ϵ^μστ$MU$117 g^νρ + γ^τ.γ^κ g^μσ g^νρ -
            γ^σ.γ^κ g^μτ g^νρ - γ^τ.γ^κ g^μρ g^νσ + γ^ρ.γ^κ g^μτ g^νσ + γ^σ.γ^κ g^μρ g^ντ - γ^ρ.γ^κ g^μσ g^ντ +
            γ^τ.γ^κ g^μν g^ρσ - γ^ν.γ^κ g^μτ g^ρσ + γ^μ.γ^κ g^ντ g^ρσ - γ^σ.γ^κ g^μν g^ρτ + γ^ν.γ^κ g^μσ g^ρτ - γ^μ.γ^κ g^νσ g^ρτ -
            i γ$MU$105.γ^κ.γ^5 ϵ^νρ$MU$105 g^στ + γ^ρ.γ^κ g^μν g^στ - γ^ν.γ^κ g^μρ g^στ + γ^μ.γ^κ g^νρ g^στ

In[493]:= a3 = Chisholm[a1]
Out[493]= -i γ^τ.γ^κ.γ^5 ϵ^μνρσ + i γ^σ.γ^κ.γ^5 ϵ^μνρτ - i γ$MU$128.γ^κ.γ^5 ϵ^ρστ$MU$128 g^μν +
            i γ$MU$131.γ^κ.γ^5 ϵ^νστ$MU$131 g^μρ - i γ$MU$134.γ^κ.γ^5 ϵ^μστ$MU$134 g^νρ + γ^τ.γ^κ g^μσ g^νρ -
            γ^σ.γ^κ g^μτ g^νρ - γ^τ.γ^κ g^μρ g^νσ + γ^ρ.γ^κ g^μτ g^νσ + γ^σ.γ^κ g^μρ g^ντ - γ^ρ.γ^κ g^μσ g^ντ +
            γ^τ.γ^κ g^μν g^ρσ - γ^ν.γ^κ g^μτ g^ρσ + γ^μ.γ^κ g^ντ g^ρσ - γ^σ.γ^κ g^μν g^ρτ + γ^ν.γ^κ g^μσ g^ρτ - γ^μ.γ^κ g^νσ g^ρτ -
            i γ$MU$122.γ^κ.γ^5 ϵ^νρ$MU$122 g^στ + γ^ρ.γ^κ g^μν g^στ - γ^ν.γ^κ g^μρ g^στ + γ^μ.γ^κ g^νρ g^στ
```

Check that both a1.a1 and a2.a3 give the same.

```
In[494]:= Calc[a1.a1]
Out[494]= -D^6 + 30 D^5 - 260 D^4 + 840 D^3 - 1120 D^2 + 512 D

In[495]:= Calc[a2.a3]
Out[495]= -15 D^4 + 60 D^3 - 214 D^2 + 380 D - 144

In[496]:= Clear[t1, t2, t3, t4, a1, a2, a3]
```

ChisholmSpinor

Description

ChisholmSpinor[x] uses the Chisholm identity on a DiraGamma between spinors. As an optional second argument 1 or 2 may be given, indicating that ChisholmSpinor should only act on the first resp. second part of a product of spinor chains.

See also: EpsChisholm, Chisholm.

Examples

```
In[497]:= Spinor[p1, m1].DiracGamma[LorentzIndex[μ]].Spinor[p2, m2]
Out[497]= φ(p1, m1).γ^μ.φ(p2, m2)

In[498]:= %//ChisholmSpinor
Out[498]= -(m1 m2 φ(p1, m1).γ^μ.φ(p2, m2))/(p1 · p2) + (i φ(p1, m1).γ^alpha1.γ^5.φ(p2, m2) ϵ^alpha1μp1p2)/(p1 · p2) +
  (m2 φ(p1, m1).φ(p2, m2) p1^μ)/(p1 · p2) + (m1 φ(p1, m1).φ(p2, m2) p2^μ)/(p1 · p2)
```

ClearScalarProducts

Description

ClearScalarProducts removes all user-performed specific settings for ScalarProduct's.
See also: ScalarProduct, Pair, SP, SPD.

Examples

```
In[499]:= ScalarProduct[p, p] = m^2
Out[499]= m^2

In[500]:= Pair[Momentum[p], Momentum[p]]
Out[500]= m^2

In[501]:= ClearScalarProducts

In[502]:= Pair[Momentum[p], Momentum[p]]
Out[502]= p^2

In[503]:= ScalarProduct[p, p]
Out[503]= p^2
```

Collecting

Description

Collecting is an option of ScalarProductCancel, Series2, TID and related functions. Setting it to True will trigger some kind of collecting of the result.
See also: ScalarProductCancel, Series2, TID.

Collect2

Description

Collect2[expr, x] collects together terms which are not free of any occurrence of x. Collect2[expr, {x1, x2, ...}] (or also Collect2[expr, x1, x2, ...]) collects together terms which are not free of any occurrence of x1, x2, ... The coefficients are put over a common denominator. If expr is expanded before collecting depends on the option Factoring, which may be set to Factor, Factor2, or any other function, which is applied to the coefficients. If expr is already expanded with respect to x (x1,x2, ...), the option Expanding can be set to False.

```
In[504]:= Options[Collect2]
Out[504]= {Denominator → False, Dot → False, Expanding → True, Factoring → Factor2, IsolateNames →
False}
```

See also: Isolate.

Examples

```
In[505]:= Collect2[t1 = a + r a + k^2 f[a] - k f[a] + x/2 - y/w, a]
Out[505]= a (r + 1) + (w x - 2 y)/(2 w) - (1 - k) k f(a)

In[506]:= Collect2[t1, a, Factoring → False]
Out[506]= a (r + 1) + x/2 - y/w + (k^2 - k) f(a)

In[507]:= Collect2[t1, a, Factoring → Factor]
Out[507]= a (r + 1) + (w x - 2 y)/(2 w) + (k - 1) k f(a)

In[508]:= Collect2[2 a (b - a) (h - 1) - b^2 (e a - c) + b^2, {a, b}]
Out[508]= 2 (1 - h) a^2 - b^2 e a - 2 b (1 - h) a + b^2 (c + 1)

In[509]:= Collect2[Expand[(a - b - c - d)^5], a, IsolateNames → L]

 L1 :: shdw : Symbol L1 appears in multiple contexts {Global`, HighEnergyPhysics`Phi`Objects`};

     definitions in context Global` may shadow or be shadowed by other definitions.
Out[509]= a^5 - 5 L(1) a^4 + 10 L(1)^2 a^3 - 10 L(1)^3 a^2 + 5 L(1)^4 a - L(1)^5

In[510]:= ReleaseHold[%]
Out[510]= a^5 - 5 (b + c + d) a^4 + 10 (b + c + d)^2 a^3 - 10 (b + c + d)^3 a^2 + 5 (b + c + d)^4 a - (b + c + d)^5

In[511]:= Clear[t1, L]

In[512]:= Collect2[Expand[(a - b - c)^3], a, Factoring → fun]
Out[512]= fun(1) a^3 + fun(-3 b - 3 c) a^2 + fun(3 b^2 + 6 c b + 3 c^2) a + fun(-b^3 - 3 c b^2 - 3 c^2 b - c^3)

In[513]:= % /. fun → FactorTerms
Out[513]= a^3 - 3 (b + c) a^2 + 3 (b^2 + 2 c b + c^2) a - b^3 - c^3 - 3 b c^2 - 3 b^2 c
```

Collect3

Description

Collect3[expr, {x, y, ...}] collects terms involving the same powers of monomials $x^{n_1} y^{n_2}$ ... An option Factor → True/False can be given, which factors the coefficients. The option Head (default Plus) determines the applied function to the list of monomials mulitplied by their coefficients.

```
In[514]:= Options[Collect3]
Out[514]= {Factor → False, Head → Plus}
```

See also: Collect2, Isolate.

Examples

```
In[515]:= Collect3[2 a (b - a) (h - 1) - b^2 (e a - c) + b^2, {a, b}]
Out[515]= (2 - 2 h) a^2 - b^2 e a + b (2 h - 2) a + b^2 (c + 1)

In[516]:= Collect3[Expand[(a - b - c - d)^5], {a}]
Out[516]= a^5 + (-5 b - 5 c - 5 d) a^4 + (10 b^2 + 20 c b + 20 d b + 10 c^2 + 10 d^2 + 20 c d) a^3 +
            (-10 b^3 - 30 c b^2 - 30 d b^2 - 30 c^2 b - 30 d^2 b - 60 c d b - 10 c^3 - 10 d^3 - 30 c d^2 - 30 c^2 d) a^2 +
            (5 b^4 + 20 c b^3 + 20 d b^3 + 30 c^2 b^2 + 30 d^2 b^2 + 60 c d b^2 + 20 c^3 b + 20 d^3 b +
              60 c d^2 b + 60 c^2 d b + 5 c^4 + 5 d^4 + 20 c d^3 + 30 c^2 d^2 + 20 c^3 d) a - b^5 - c^5 - d^5 - 5 b c^4 -
            5 b d^4 - 5 c d^4 - 10 b^2 c^3 - 10 b^2 d^3 - 10 c^2 d^3 - 20 b c d^3 - 10 b^3 c^2 - 10 b^3 d^2 - 10 c^3 d^2 -
            30 b c^2 d^2 - 30 b^2 c d^2 - 5 b^4 c - 5 b^4 d - 5 c^4 d - 20 b c^3 d - 30 b^2 c^2 d - 20 b^3 c d
```

Combinations

Description

Combinations[l, n] returns a list of all possible sets containing n elements from the list l. (this function is probably in the combinatorics package, but we have enough in memory already).

Examples

```
In[517]:= Combinations[{a, b, c, d}, 3]//TableForm
Out[517]= Combinations({a, b, c, d}, 3)
```

Combine

Description

Combine[expr]puts terms in a sum over a common denominator,and cancels factors in the result. Combine is similar to Together, but accepts the option Expanding and works usually better than Together for polynomials involving rationals with sums in the denominator.

```
In[518]:= Options[Combine]
Out[518]= {Expanding → False}
```

See also: Factor2.

Examples

```
In[519]:= Combine[(a - b) (c - d)/e + g]
Out[519]= (a - b) (c - d) + e g
          ---------------------
                    e
```

Here the result from Together where the numerator is automatically expanded.

```
In[520]:= Together[(a - b) (c - d)/e + g]
Out[520]= a c - b c - a d + b d + e g
          ---------------------------
                       e
```

If the option Expanding is set to True, the result of Combine is the same as Together, but uses a slightly different algorithm.

```
In[521]:= Combine[(a - b) (c - d)/e + g, Expanding → True]
Out[521]= a c - b c - a d + b d + e g
          ---------------------------
                       e
```

CombineGraphs

Description

CombineGraphs is an option for OneLoopSum.

See also: OneLoopSum.

Commutator

Description

Commutator[x, y] = c defines the commutator between the non-commuting objects x and y.

See also: AntiCommutator, CommutatorExplicit, DeclareNonCommutative, DotSimplify.

Examples

```
In[522]:= DeclareNonCommutative[a, b, c, d]

In[523]:= Commutator[a, b]
Out[523]= [a, b]

In[524]:= CommutatorExplicit[%]
Out[524]= a.b - b.a

In[525]:= DotSimplify[Commutator[a + b, c + d]]
Out[525]= a.c + a.d + b.c + b.d - c.a - c.b - d.a - d.b

In[526]:= UnDeclareNonCommutative[a, b, c, d]
```

Verify the Jacobi identity.

```
In[527]:= χ = Commutator; DeclareNonCommutative[x, y, z];

In[528]:= χ[x, χ[y, z]] + χ[y, χ[z, x]] + χ[z, χ[x, y]]
Out[528]= [x, [y, z]] + [y, [z, x]] + [z, [x, y]]

In[529]:= DotSimplify[%]
Out[529]= 0

In[530]:= Clear[χ]

In[531]:= UnDeclareNonCommutative[x, y, z]
```

CommutatorExplicit

Description

CommutatorExplicit[exp] substitutes any Commutator and AntiCommutator in exp by their definitions.
See also: Calc, DotSimplify.

Examples

```
In[532]:= DeclareNonCommutative[a, b, c, d]

In[533]:= Commutator[a, b]
Out[533]= [a, b]

In[534]:= CommutatorExplicit[%]
Out[534]= a.b - b.a

In[535]:= AntiCommutator[a - c, b - d]
Out[535]= {a - c, b - d}

In[536]:= CommutatorExplicit[%]
```

```
Out[536]= (a - c).(b - d) + (b - d).(a - c)

In[537]:= CommutatorExplicit[%%]//DotSimplify
Out[537]= a.b - a.d + b.a - b.c - c.b + c.d - d.a + d.c

In[538]:= UnDeclareNonCommutative[a, b, c, d]
```

CompleteSquare

Description

Completes the square of a second order polynomial in the momentum x. CompleteSquare[a p^2+b p+c, p] → -b^2/(4 a)+c+a (b/(2 a)+x)^2. CompleteSquare[a p^2+b p+c, p, q] → {-b^2/(4 a)+c+a q^2, q→b/(2 a)+p}.

Examples

```
In[539]:= t1 = 5SP[2p + 3r, p + r]//FCI
Out[539]= 5 (p + r) · (2 p + 3 r)

In[540]:= t2 = CompleteSquare[t1, p]
Out[540]= 10 (p + 5 r/4) . (p + 5 r/4) - 5 r^2/8

In[541]:= t1 - t2//ScalarProductExpand//Expand
Out[541]= 0

In[542]:= CompleteSquare[t1, p, q]
Out[542]= {10 q^2 - 5 r^2/8, q → p + 5 r/4}

In[543]:= Clear[t1, t2]
```

ComplexConjugate

Description

ComplexConjugate[expr] complex conjugates expr. It operates on fermion lines, i.e., products of Spinor[..] .DiracMatrix[..] . Spinor[..], and changes all occuring LorentzIndex[mu] into LorentzIndex[ComplexIndex[mu]]. For taking the spin sum (i.e. constructing the traces) use FermionSpinSum. WARNING: In expr should be NO explicit I's in denominators!

See also: ComplexIndex, FermionSpinSum, LorentzIndex.

Examples

```
In[544]:= ComplexConjugate[MetricTensor[μ, ν]]
Out[544]= g^(μν)

In[545]:= StandardForm[%]
Out[545]= Pair[LorentzIndex[μ], LorentzIndex[ν]]

In[546]:= GA[μ, ν, 5]
Out[546]= γ^μ.γ^ν.γ^5

In[547]:= ComplexConjugate[%]
Out[547]= -γ^5.γ^ν.γ^μ

In[548]:= SUNTrace[SUNT[a, b, c]]
Out[548]= tr(T_a.T_b.T_c)

In[549]:= ComplexConjugate[%]
Out[549]= tr(T_c.T_b.T_a)

In[550]:= ComplexConjugate[SUNF[a, b, c]]
Out[550]= f_abc

In[551]:= StandardForm[FCE[ComplexConjugate[MetricTensor[μ, ν]]]]
Out[551]= MT[μ, ν]

In[552]:= SpinorUBar[k1, m].GA[λ].SpinorU[p1, m]
Out[552]= ū(k1, m).γ^λ.u(p_1, m)

In[553]:= ComplexConjugate[%]
Out[553]= φ(p_1, m).γ^λ.φ(k1, m)
```

Notice that SpinorUBar and SpinorU are only input functions. Internally they are converted to Spinor objects.

Contract

Description

Contract[expr] contracts pairs of Lorentz indices of metric tensors, four-vectors and (depending on the optionEpsContract) of Levi-Civita tensors in expr. For the contraction of Dirac matrices with each other use DiracSimplify. Contract[exp1, exp2] contracts (exp1*exp2), where exp1 and exp2 may be larger products of sums of metric tensors and 4-vectors.

```
In[554]:= Options[Contract]
Out[554]= {Collecting → True, Contract3 → False, EpsContract → True, Expanding → True, Factoring → False,

            FeynCalcInternal → False, MomentumCombine → False, Rename → False, Schouten → 0}
```

The option setting Contract3 can be set to True resulting in a faster algorithm to contract products of tensors.

See also: Pair, DiracSimplify, MomentumCombine.

Examples

```
In[555]:= MetricTensor[μ, ν] FourVector[p, μ]
Out[555]= p_μ g^μν

In[556]:= Contract[%]
Out[556]= p^ν

In[557]:= FourVector[p, μ]DiracMatrix[μ]
Out[557]= γ^μ p_μ

In[558]:= Contract[%]
Out[558]= γ · p

In[559]:= MetricTensor[μ, μ]
Out[559]= g^μμ
```

The default dimension for MetricTensor is 4.

```
In[560]:= Contract[%]
Out[560]= 4
```

A short way to enter D-dimensional metric tensors is given by MTD. The "." as multiplication operator is not necessary but just convenient for typesetting.

```
In[561]:= MTD[μ, ν] . MTD[μ, ν]
Out[561]= g^μν . g^μν

In[562]:= Contract[%]
Out[562]= D

In[563]:= MTD[μ, ν] . MTD[μ, ν]
Out[563]= g^μν . g^μν

In[564]:= FourVector[p, μ] FourVector[q, μ]
Out[564]= p_μ q_μ

In[565]:= Contract[% ]
Out[565]= p · q

In[566]:= FourVector[p - q, μ] FourVector[a - b, μ]
Out[566]= (a - b)_μ (p - q)_μ

In[567]:= Contract[%]
Out[567]= a · p - a · q - b · p + b · q

In[568]:= LeviCivita[μ, ν, α, σ] FourVector[p, σ]
Out[568]= ϵ^μνασ p_σ

In[569]:= Contract[%]
Out[569]= ϵ^αμνp

In[570]:= LeviCivita[μ, ν, α, β] LeviCivita[μ, ν, α, σ]
```

```
Out[570]= ϵ^{μναβ} ϵ^{μνασ}

In[571]:= Contract[%]
Out[571]= -6 g^{βσ}

In[572]:= SetOptions[Eps, Dimension → D]; LCD[μ, ν, α, β] LCD[μ, ν, α, σ]
Out[572]= ϵ^{μναβ} ϵ^{μνασ}

In[573]:= Contract[%]//Factor2
Out[573]= (1 - D) (2 - D) (3 - D) g^{βσ}

In[574]:= SetOptions[Eps, Dimension → 4];
```

Contract1

Description

Contract1[exp] contracts Upper and Lower indices. Neither Upper and Upper nor Lower and Lower indices are contracted.

See also: LorentzIndex, Lower, Upper.

Examples

```
In[575]:= FV[p, Lower[μ]]FV[q, Upper[μ]]//FCI//Contract1
Out[575]= p · q

In[576]:= FV[p, Upper[μ]]FV[q, Upper[μ]]//FCI//Contract1
Out[576]= p^{Upper(μ)} q^{Upper(μ)}

In[577]:= MT[Lower[μ], Upper[μ]]//FCI//Contract1
Out[577]= 4

In[578]:= MT[Upper[μ], Upper[μ]]//FCI//Contract1
Out[578]= g^{Upper(μ)Upper(μ)}
```

Convolute

Description

Convolute[f, g, x] convolutes $f(x)$ and $g(x)$, i.e., $\int_0^1 dx_1 \int_0^1 dx_2 \delta(x - x_1 x_2) f(x_1) g(x_2)$. Convolute[f, g] is equivalent to Convolute[f, g, x]. Convolute[exp, {x1, x2}] assumes that exp is polynomial in x1 and x2. Convolute uses table-look-up and does not do any integral calculations, only linear algebra.

```
In[579]:= Options[Convolute]
Out[579]= {Bracket → {ε}, FinalSubstitutions → {PlusDistribution → Identity}}
```

See also: PlusDistribution, ConvoluteTable.

Examples

```
In[580]:= Convolute[1, 1]
Out[580]= -Log(x)

In[581]:= Convolute[x, x]
Out[581]= -x Log(x)

In[582]:= Convolute[1, x]
Out[582]= 1 - x

In[583]:= Convolute[1, 1/(1 - x)]
Out[583]= Log(1 - x) - Log(x)

In[584]:= Convolute[1, PlusDistribution[1/(1 - x)]]
Out[584]= Log(1 - x) - Log(x)

In[585]:= Convolute[1/(1 - x), x]
Out[585]= Log(1 - x) x - Log(x) x - x + 1

In[586]:= Convolute[1/(1 - x), 1/(1 - x)]
Out[586]= -ζ(2) δ(1 - x) + 2 Log(1 - x)/(1 - x) - Log(x)/(1 - x)

In[587]:= Convolute[1, Log[1 - x]]
Out[587]= -Log(1 - x) Log(x) - Li_2(1 - x)

In[588]:= Convolute[1, x Log[1 - x]]
Out[588]= x + (1 - x) Log(1 - x) - 1

In[589]:= Convolute[1/(1 - x), Log[1 - x]]
Out[589]= Log^2(1 - x) - Log(x) Log(1 - x) - ζ(2)

In[590]:= Convolute[1/(1 - x), x Log[1 - x]]
Out[590]= x Log^2(1 - x) + (1 - x) Log(1 - x) - x Log(x) Log(1 - x) + x - x ζ(2) - 1

In[591]:= Convolute[Log[1 - x]/(1 - x), x]
Out[591]= 1/2 x Log^2(1 - x) + (1 - x) Log(1 - x) - x Log(x) Log(1 - x) + x Log(x) - x Li_2(1 - x)

In[592]:= Convolute[1, x Log[x]]
Out[592]= -Log(x) x + x - 1

In[593]:= Convolute[Log[1 - x], x]
Out[593]= (1 - x) Log(1 - x) + x Log(x)

In[594]:= Convolute[1/(1 - x), Log[x]/(1 - x)]
Out[594]= Log(1 - x) Log(x)/(1 - x) - Log^2(x)/(2 (1 - x))
```

```
In[595]:= Convolute[1, Log[x]]
Out[595]= -1/2 Log^2(x)

In[596]:= Convolute[x, x Log[x]]
Out[596]= -1/2 x Log^2(x)

In[597]:= Convolute[1/(1 - x), Log[x]]
Out[597]= -1/2 Log^2(x) + Log(1 - x) Log(x) + Li_2(1 - x)

In[598]:= Convolute[1, Log[x]/(1 - x)]
Out[598]= -1/2 Log^2(x) - Li_2(1 - x)

In[599]:= Convolute[1/(1 - x), x Log[x]]
Out[599]= -1/2 x Log^2(x) - x Log(x) + x Log(1 - x) Log(x) + x + x Li_2(1 - x) - 1

In[600]:= Convolute[Log[x]/(1 - x), x]
Out[600]= -1/2 x Log^2(x) + Log(x) - x - x Li_2(1 - x) + 1

In[601]:= Convolute[1, x Log[x]]
Out[601]= - Log(x) x + x - 1

In[602]:= Convolute[Log[x], x]
Out[602]= -x + Log(x) + 1

In[603]:= Convolute[1/(1 - x), Log[1 - x]/(1 - x)]
Out[603]= (3 Log^2(1 - x))/(2 (1 - x)) - (Log(x) Log(1 - x))/(1 - x) - ζ(2)/(1 - x) + δ(1 - x) ζ(3)
```

ConvoluteTable ***unfinished***

Description

ConvoluteTable[f, g, x] yields the convolution of f and g.

```
In[604]:= Options[Convolute]
Out[604]= {Bracket → {ε}, FinalSubstitutions → {PlusDistribution → Identity}}
```

See also: PlusDistribution, Convolute.

Examples

```
In[605]:= ConvoluteTable[1, 1]
Out[605]= -Log(x)

In[606]:= ConvoluteTable[x, x]
Out[606]= -x Log(x)

In[607]:= ConvoluteTable[1, x]
Out[607]= 1 - x
```

CovariantFieldDerivative

Description

CovariantFieldDerivative[f[x],x,{li1,li2,...},opts] is a covariant derivative of f[x] with respect to space-time variables x and with Lorentz indices li1, li2,... CovariantFieldDerivative has only typesetting definitions by default. The user is must supply his/her own definition of the actual function.

See also: CovariantD, ExpandPartialD, FieldDerivative.

Examples

```
In[608]:= CovariantFieldDerivative[QuantumField[A, {μ}][x], x, {μ}]
```

$$\text{Out[608]}= \frac{1}{2}\,\mathrm{i}\left(\left(\vec{A}_\mu\cdot\vec{\sigma}+\vec{V}_\mu\cdot\vec{\sigma}\right)\text{✱}A_\mu\right)-\frac{1}{2}\,\mathrm{i}\left(A_\mu\text{✱}\left(\vec{V}_\mu\cdot\vec{\sigma}-\vec{A}_\mu\cdot\vec{\sigma}\right)\right)+\partial_\mu A_\mu$$

CounterT

Description

CounterT is a factor used by GluonPropagator and QuarkPropagator when CounterTerms is set to All.

See also: CounterTerm, GluonPropagator, QuarkPropagator.

Examples

```
In[609]:= GluonPropagator[p, μ, a, ν, b, Explicit → True, CounterTerm → All]
```

$$\text{Out[609]}= \text{CounterT}\left(\frac{\mathrm{i}\,C_A\,S_n\left(\frac{10\,p^\mu\,p^\nu}{3}-\frac{10}{3}\,g^{\mu\nu}\,p^2\right)\delta_{ab}\,g_s^2}{\varepsilon}+\frac{\mathrm{i}\,S_n\,T_f\left(\frac{4}{3}\,g^{\mu\nu}\,p^2-\frac{4\,p^\mu\,p^\nu}{3}\right)\delta_{ab}\,g_s^2}{\varepsilon}\right)-\frac{\mathrm{i}\,g^{\mu\nu}\,\delta_{ab}}{p^2}$$

CounterTerm

Description

CounterTerm[name] is a database of counter terms. CounterTerm is also an option for the Feynman rule functions QuarkGluonVertex, GluonPropagator, QuarkPropagator.

See also: CounterT, QuarkGluonVertex, GluonPropagator, QuarkPropagator.

Examples

```
In[610]:= CounterTerm[Zm]
```

$$\text{Out[610]}= \frac{C_F\left(\frac{4\left(\frac{11 C_A}{2}+\frac{9 C_F}{2}-2 N_f T_f\right)}{\varepsilon^2}+\frac{2\left(\frac{97 C_A}{12}+\frac{3 C_F}{4}-\frac{5 N_f T_f}{3}\right)}{\varepsilon}\right) g_s^4}{256\,\pi^4}+\frac{3 C_F g_s^2}{8\,\varepsilon\,\pi^2}+1$$

CouplingConstant

Description

In general, CouplingConstant is an option for several Feynman rule functions and for CovariantD and FieldStrength.
In the convention of the subpackage PHI, CouplingConstant is also the head of coupling constants. CouplingConstant takes three extra optional arguments, with head RenormalizationState, RenormalizationScheme and ExpansionState respectively. E.g. CouplingConstant[QED[1]] is the unit charge, CouplingConstant[ChPT2[4],1] is the first of the coupling constants of the lagrangian ChPT2[4]. CouplingConstant[a_,b_,c___][i_] := CouplingConstant[a,b,RenormalizationState[i],c].

See also: CovariantD, FieldStrength.

CovariantD

Description

CovariantD[mu] is a generic covariant derivative with Lorentz index mu. With the option-setting Explicit → True, an explicit expression for a fermionic field is returned, depending on the setting on the other options.
CovariantD[x, mu] is a generic covariant derivative with respect to xˆmu.
CovariantD[mu, a, b] is a covariant derivative for a bosonic field; acting on QuantumField[f,{},{a,b}], where f is some field name and a and b are two SU(N) indices. Again, with the option-setting Explicit → True, an explicit expression is returned, depending on the setting on the other options.
CovariantD[OPEDelta, a, b] is a short form for CovariantD[mu,a,b]*FourVector[OPEDelta, mu]. CovariantD[{OPEDelta, a, b}, {n}] yields the product of n operators, where n is an integer. CovariantD[OPEDelta, a, b, {m, n}] gives the expanded form of CovariantD[OPEDelta, a, b]ˆm up to order gˆn for the gluon, where

n is an integer and g the coupling constant indicated by the setting of the option CouplingConstant. CovariantD[OPEDelta, {m, n}] gives the expanded form of CovariantD[OPEDelta]^m up to order g^n of the fermionic field.

```
In[611]:= Options[CovariantD]
Out[611]= {CouplingConstant → g_s, DummyIndex → Automatic,
             Explicit → False, PartialD → RightPartialD, QuantumField → A}
```

Possible settings of PartialD are: LeftPartialD, LeftRigthPartialD, RightPartialD. The default setting of QuantumField is GaugeField.

See also: LeftPartialD, LeftRightPartialD, RightPartialD.

Examples

```
In[612]:= CovariantD[μ]
Out[612]= D_μ
```

```
In[613]:= CovariantD[μ, a, b]
Out[613]= D_μ^ab
```

```
In[614]:= CovariantD[μ, Explicit → True]
Out[614]= (∂→)_μ - i g_s T_c1 . A_μ^c1
```

The first argument of CovariantD is intepreted as type LorentzIndex, except for OPEDelta, which is type Momentum.

```
In[615]:= CovariantD[OPEDelta]
Out[615]= D_Δ
```

```
In[616]:= CovariantD[OPEDelta, a, b]
Out[616]= D_Δ^ab
```

```
In[617]:= CovariantD[OPEDelta, a, b, Explicit → True]
Out[617]= (∂→)_Δ δ_ab - g_s A_Δ^c2 f_abc2
```

```
In[618]:= CovariantD[OPEDelta, Explicit → True]
Out[618]= (∂→)_Δ - i g_s T_c3 . A_Δ^c3
```

```
In[619]:= CovariantD[OPEDelta, a, b, {2}]
Out[619]= ((∂→)_Δ δ_ac9 - g_s A_Δ^e1 f_ac9e1) . ((∂→)_Δ δ_bc9 - g_s A_Δ^e2 f_c9be2)
```

This gives m times $\left(\vec{\partial}\right)_\Delta$, the partial derivative $\left(\vec{\partial}\right)_\mu$ contracted with Δ^μ.

```
In[620]:= CovariantD[OPEDelta, a, b, {OPEm, 0}]
Out[620]= ((∂→)_Δ)^m δ_ab
```

The expansion up to first order in the coupling constant g_s (thesumistheFeynCalcOPESum).

```
In[621]:= CovariantD[OPEDelta, a, b, {OPEm, 1}]
```

$$\text{Out[621]= } \left(\left(\vec{\partial}\right)_{\Delta}\right)^{m} \delta_{ab} - g_s \sum_{i=0}^{m-1} \left(\left(\vec{\partial}\right)_{\Delta}\right)^{i} . A_{\Delta}^{c_1} . \left(\left(\vec{\partial}\right)_{\Delta}\right)^{-i+m-1} f_{abc_1}$$

The expansion up to second order in the g_s.

```
In[622]:= CovariantD[OPEDelta, a, b, {OPEm, 2}]
```

$$\text{Out[622]= } \delta_{ab} \left(\left(\vec{\partial}\right)_{\Delta}\right)^{m} - g_s^2 \sum_{j=0}^{m-2} \sum_{i=0}^{j} \left(\left(\vec{\partial}\right)_{\Delta}\right)^{i} . A_{\Delta}^{c_1} . \left(\left(\vec{\partial}\right)_{\Delta}\right)^{j-i} . A_{\Delta}^{c_2} . \left(\left(\vec{\partial}\right)_{\Delta}\right)^{-j+m-2} f_{ac_1e_1} f_{bc_2e_1} - g_s \sum_{i=0}^{m-1} \left(\left(\vec{\partial}\right)_{\Delta}\right)^{i} . A_{\Delta}^{c_1} . \left(\left(\vec{\partial}\right)_{\Delta}\right)^{-i+m-1} f_{abc_1}$$

```
In[623]:= CovariantD[OPEDelta, a, b]^OPEm
```

$$\text{Out[623]= } \left(D_{\Delta}^{ab}\right)^{m}$$

```
In[624]:= CovariantD[OPEDelta, {OPEm, 2}]
```

$$\text{Out[624]= } \left(\left(\vec{\partial}\right)_{\Delta}\right)^{m} - i g_s \sum_{i=0}^{m-1} T_{c_1} . \left(\left(\vec{\partial}\right)_{\Delta}\right)^{i} . A_{\Delta}^{c_1} . \left(\left(\vec{\partial}\right)_{\Delta}\right)^{-i+m-1} - g_s^2 \sum_{j=0}^{m-2} \sum_{i=0}^{j} T_{c_1} . T_{c_2} . \left(\left(\vec{\partial}\right)_{\Delta}\right)^{i} . A_{\Delta}^{c_1} . \left(\left(\vec{\partial}\right)_{\Delta}\right)^{j-i} . A_{\Delta}^{c_2} . \left(\left(\vec{\partial}\right)_{\Delta}\right)^{-j+m-2}$$

```
In[625]:= CovariantD[OPEDelta, Explicit → True]//StandardForm
Out[625]= -i Gstrong
            SUNT[SUNIndex[c4]].QuantumField[GaugeField, Momentum[OPEDelta], SUNIndex[c4]]+
          RightPartialD[Momentum[OPEDelta]]
```

```
In[626]:= CovariantD[μ, a, b, Explicit → True]//StandardForm
Out[626]= RightPartialD[LorentzIndex[μ]] SUNDelta[a, b]-
          Gstrong QuantumField[GaugeField, LorentzIndex[μ], SUNIndex[c5]] SUNF[a, b, c5]
```

CrossProduct

Description

CrossProduct[a, b] denotes the three-dimensional cross-product of the three-vectors a and b.
See also: DotProduct, ThreeVector.

Examples

```
In[627]:= CrossProduct[ThreeVector[a], CrossProduct[ThreeVector[b], ThreeVector[c]]]
```

$$\text{Out[627]= } \vec{a} \cdot \vec{c}\, \vec{b} - \vec{a} \cdot \vec{b}\, \vec{c}$$

C0

Description

C0[p10, p12, p20, m1^2, m2^2, m3^2] is the scalar Passarino-Veltman C_0function. The convention for the arguments is that if the denominator of the integrand has the form ([q^2-m1^2] [(q+p1)^2-m2^2] [(q+p2)^2-m3^2]), the first three arguments of C0 are the scalar products p10 = p1^2, p12 = (p1-p2).(p1-p2), p20 = p2^2.

See also: B0, D0, PaVe, PaVeOrder.

Examples

```
In[628]:= C0[a, b, c, m12, m22, m32]
Out[628]= C0(a, b, c, m12, m22, m32)

In[629]:= C0[b, a, c, m32, m22, m12]//PaVeOrder
Out[629]= C0(a, b, c, m12, m22, m32)

In[630]:= PaVeOrder[C0[b, a, c, m32, m22, m12], PaVeOrderList → {c, a}]
Out[630]= C0(b, c, a, m22, m32, m12)
```

DataType

Description

DataType[exp, type] = True defines the object exp to have data-type type. DataType[exp1, exp2, ..., type] defines the objects exp1, exp2, ...to have data-type type. The default setting is DataType[__, _] := False. To assign a certain data-type, do, e.g., DataType[x, PositiveInteger] = True.

Currently used DataTypes: NonCommutative, PositiveInteger, NegativeInteger, PositiveNumber, FreeIndex, GrassmannParity

PHI adds the DataTypes: UMatrix, UScalar.

See also: DeclareNonCommutative.

Examples

NonCommutative is just a data-type.

```
In[631]:= DataType[f, g, NonCommutative] = True;

In[632]:= t = f.g - g.(2a).f
Out[632]= f.g - g.(2 a).f
```

Since "f "and "g" have DataType NonCommutative the function DotSimplify extracts only "a" out of the non-commutative product.

```
In[633]:= DotSimplify[t]
```

```
Out[633]= f.g - 2 a g.f

In[634]:= DataType[m, odd] = DataType[a, even] = True;

In[635]:= ptest1[x_] := x/.(-1)^n_/; DataType[n, odd] :→ -1;

In[636]:= ptest2[x_] := x/.(-1)^n_/; DataType[n, even] :→ 1;

In[637]:= t = (-1)^m + (-1)^a + (-1)^z
Out[637]= (-1)^a + (-1)^m + (-1)^z

In[638]:= ptest1[t]
Out[638]= -1 + (-1)^a + (-1)^z

In[639]:= ptest2[%]
Out[639]= (-1)^z

In[640]:= Clear[ptest1, ptest2, t, a, m];

In[641]:= DataType[m, ganzeZahl] = True;

In[642]:= f[x_] := x/.{(-1)^p_/; DataType[p, ganzeZahl] :→ 1};

In[643]:= test = (-1)^m + (-1)^n x
Out[643]= (-1)^n x + (-1)^m

In[644]:= f[test]
Out[644]= (-1)^n x + 1

In[645]:= Clear[f, test];

In[646]:= DataType[f, g, NonCommutative] = False;

In[647]:= DataType[m, odd] = DataType[a, even] = False;
```

Certain FeynCalc objects have DataType PositiveInteger set to True.

```
In[648]:= DataType[OPEm, PositiveInteger]
Out[648]= True
```

PowerSimplify uses the DataType information.

```
In[649]:= PowerSimplify[ (-1)^(2OPEm)]
Out[649]= 1

In[650]:= PowerSimplify[ (- SO[q])^OPEm]
Out[650]= (-1)^m (Δ · q)^m
```

DB0

Description

DB0[p2, m1^2, m2^2] is the derivative of the two-point function B0[p2, m1^2, m2^2] with respect to p2.
See also: B0.

Examples

```
In[651]:= D[B0[p2, m1^2, m2^2], p2]
Out[651]= DB0(p2, m1^2, m2^2)
```

DB1

Description

DB1[p2,m1^2,m2^2] is the derivative of B1[p2,m1^2,m2^2] with respect to p2.
See also: B1.

Examples

```
In[652]:= D[B1[p2, m1^2, m2^2], p2]
```

$$\text{Out[652]}= \frac{(m_2^2 - m_1^2)\,\text{DB0}(p_2, m_1^2, m_2^2)}{2\,p_2} - \frac{1}{2}\,\text{DB0}(p_2, m_1^2, m_2^2) - \frac{(B_0(p_2, m_1^2, m_2^2) - B_0(0, m_1^2, m_2^2))\,(m_2^2 - m_1^2)}{2\,p_2^2}$$

DeclareNonCommutative

Description

DeclareNonCommutative[a, b, ...] declares a,b, ... to be non-commutative, i.e., DataType[a,b, ..., NonCommutative] is set to True.
See also: DataType, UnDeclareNonCommutative.

Examples

```
In[653]:= DeclareNonCommutative[x]
```

As a side effect of DeclareNonCommutative x is declared to be of data type NonCommutative.

```
In[654]:= DataType[x, NonCommutative]
Out[654]= True

In[655]:= DeclareNonCommutative[y, z]

In[656]:= DataType[a, x, y, z, NonCommutative]
Out[656]= {True, True, True, True}

In[657]:= UnDeclareNonCommutative[x, y, z]

In[658]:= DataType[a, x, y, z, NonCommutative]
Out[658]= {True, True, True, True}
```

DeltaFunction

Description

DeltaFunction[x] is the Dirac delta-function $\delta(x)$.
*** After version 4 of *Mathematica*, there is a built-in function, DiracDelta, with comparable properties. ***
See also: Convolute, DeltaFunctionPrime, Integrate2, SimplifyDeltaFunction.

Examples

```
In[659]:= DeltaFunction[1 - x]
Out[659]= δ(1 - x)

In[660]:= Integrate2[DeltaFunction[1 - x] f[x], {x, 0, 1}]
Out[660]= f(1)

In[661]:= Integrate2[DeltaFunction[x] f[x], {x, 0, 1}]
Out[661]= f(0)

In[662]:= Integrate2[DeltaFunction[1 - x] f[x], {x, 0, 1}]
Out[662]= f(1)

In[663]:= Convolute[DeltaFunction[1 - x], x]
Out[663]= x
```

DeltaFunctionDoublePrime

Description

DeltaFunctionDoublePrime[1-x] is the second derivative of the Dirac delta-function $\delta(x)$.
See also: DeltaFunctionPrime.

DeltaFunctionPrime

Description

DeltaFunctionPrime[1-x] is the derivative of the Dirac delta-function $\delta(x)$.
See also: Convolute, DeltaFunction, Integrate2, SimplifyDeltaFunction.

Examples

```
In[664]:= DeltaFunctionPrime[1 - x]
Out[664]= δ'(1 - x)

In[665]:= Integrate2[DeltaFunctionPrime[1 - x] f[x], {x, 0, 1}]
Out[665]= f'(1)

In[666]:= Integrate2[DeltaFunctionPrime[1 - x] x^2, {x, 0, 1}]
Out[666]= 2
```

DenominatorOrder

Description

DenominatorOrder is an option for OneLoop, if set to True the PropagatorDenominator will be ordered in a standard way.
See also: OneLoop, PropagatorDenominator.

Dimension

Description

Dimension is an option of several functions and denotes the number of space-time dimensions. Possible settings are: 4, n, d, D, ... ,the variable does not matter, but it should have Head Symbol.

```
In[667]:= Options[MetricTensor]
Out[667]= {Dimension → 4, FeynCalcInternal → False}

In[668]:= MetricTensor[m, n, Dimension → d]DiracMatrix[α, Dimension → d]//FCI
Out[668]= γ^α g^mn
```

The dimension of the indices is not shown by default but can be inspected easily.

```
In[669]:= MetricTensor[m, n, Dimension → d]DiracMatrix[α, Dimension → d]//StandardForm
Out[669]= DiracGamma[LorentzIndex[α, d], d] MetricTensor[m, n, Dimension → d]
```

Setting the global variable $LorentzIndices to True will display the dimension (if different from 4) as a subscript.

```
In[670]:= $LorentzIndices = True;

In[671]:= MetricTensor[α, β, Dimension → n]DiracMatrix[α, Dimension → n]
Out[671]= γ^α_n MetricTensor(α, β, Dimension → n)

In[672]:= %//StandardForm
Out[672]= DiracGamma[LorentzIndex[α, n], n] MetricTensor[α, β, Dimension → n]

In[673]:= $LorentzIndices = False;
```

DimensionalReduction

Description

DimensionalReduction is an option for TID and OneLoopSimplify.
See also: TID, OneLoopSimplify.

DiracBasis

Description

DiracBasis[any] is a head which is wrapped around Dirac structures (and the 1) as a result of the function DiracReduce. Eventually you want to substitute DiracBasis by Identity (or set: DiracBasis[1] = S; DiracBasis[DiracMatrix[mu]] = P; etc.).
See also: DiracReduce.

DiracCanonical

Description

DiracCanonical is an option for DiracSimplify. If set to True DiracSimplify uses the function DiracOrder internally.
See also: DiracSimplify.

DiracEquation

Description

DiracEquation[exp] applies the Dirac equation without expanding exp. If that is needed, use DiracSimplify.
See also: DiracSimplify.

Examples

```
In[674]:= Spinor[Momentum[p], m, 1].DiracGamma[Momentum[p]]
Out[674]= φ(p, m).(γ · p)

In[675]:= DiracEquation[%]
Out[675]= m φ(p, m)
```

DiracGamma

Description

DiracGamma[x, dim] is the head of all Dirac matrices and slashes (in the internal representation). Use DiracMatrix (or GA, GAD) and DiracSlash (or GS, GSD) for manual (short) input. DiracGamma[x, 4] simplifies to DiracGamma[x]. DiracGamma[5] is γ^5. DiracGamma[6] is $(1 + \gamma^5)/2$. DiracGamma[7] is $(1 - \gamma^5)/2$.

See also: DiracGammaExpand, DiracMatrix, DiracSimplify, DiracSlash, DiracTrick.

Examples

```
In[676]:= DiracGamma[5]
Out[676]= γ^5

In[677]:= DiracGamma[LorentzIndex[α]]
Out[677]= γ^α
```

A Dirac-slash, i.e., $\gamma^\mu q_\mu$, is displayed as $\gamma \cdot q$.

```
In[678]:= DiracGamma[Momentum[q]]
Out[678]= γ · q

In[679]:= DiracGamma[Momentum[q]] . DiracGamma[Momentum[p - q]]
Out[679]= (γ · q) . (γ · (p - q))

In[680]:= DiracGamma[Momentum[q, D], D]
Out[680]= γ · q

In[681]:= a1 = GS[p - q] . GS[p]
Out[681]= (γ · (p - q)) . (γ · p)

In[682]:= a2 = DiracGammaExpand[a1]
Out[682]= (γ · p - γ · q) . (γ · p)

In[683]:= a3 = GAD[μ] . GSD[p - q] . GSD[q] . GAD[μ]
Out[683]= γ^μ . (γ · (p - q)) . (γ · q) . γ^μ

In[684]:= a4 = DiracTrick[a3]
Out[684]= (D - 4) (γ · (p - q)) . (γ · q) + 4 (p - q) · q

In[685]:= a5 = DiracSimplify[a4]
Out[685]= D (γ · p) . (γ · q) - 4 (γ · p) . (γ · q) + 4 p · q - D q^2

In[686]:= Clear[a1, a2, a3, a4, a5]
```

DiracGammaCombine

Description

DiracGammaCombine[exp] is (nearly) the inverse operation to DiracGammaExpand.
See also: DiracGamma, DiracGammaExpand, DiracMatrix, DiracSimplify, DiracSlash, DiracTrick.

Examples

```
In[687]:= DiracGammaCombine[GS[p] + GS[q]]
Out[687]= γ · (p + q)

In[688]:= StandardForm[%]
Out[688]= DiracGamma[Momentum[p + q]]

In[689]:= DiracGammaCombine[2 GS[p] - 2 GS[q]]
Out[689]= γ · (2 p - 2 q)

In[690]:= StandardForm[%]
Out[690]= DiracGamma[Momentum[2 p - 2 q]]

In[691]:= DiracGammaExpand[%%]
Out[691]= 2 γ · p - 2 γ · q
```

DiracGammaExpand

Description

DiracGammaExpand[exp] expands all DiracGamma[Momentum[a+b+..]] in exp into (DiracGamma[Momentum[a]] + DiracGamma[Momentum[b]] + ...).
See also: DiracGamma, DiracGammaCombine, DiracMatrix, DiracSimplify, DiracSlash, DiracTrick.

Examples

```
In[692]:= t = DiracGamma[Momentum[q]] . DiracGamma[Momentum[p - q]]
Out[692]= (γ · q).(γ · (p - q))
```

Momentum is the head of p-q, i.e., it is treated as one four-momentum.

```
In[693]:= StandardForm[t]
Out[693]= DiracGamma[Momentum[q]].DiracGamma[Momentum[p - q]]
```

With DiracGammaExpand the Momentum[p-q] gets expanded.

```
In[694]:= DiracGammaExpand[t]
Out[694]= (γ · q).(γ · p - γ · q)
```

The inverse operation is DiracGammaCombine.

```
In[695]:= StandardForm[DiracGammaCombine[DiracGammaExpand[t]]]
Out[695]= DiracGamma[Momentum[q]].DiracGamma[Momentum[p - q]]

In[696]:= StandardForm[DiracGammaExpand[t]]
Out[696]= DiracGamma[Momentum[q]].(DiracGamma[Momentum[p]] - DiracGamma[Momentum[q]])
```

In order to do non-commutative expansion use DiracSimplify.

```
In[697]:= DiracSimplify[t]
Out[697]= (γ · q).(γ · p) - q^2

In[698]:= Clear[t]
```

DiracGammaT

Description

DiracGammaT[x] denotes the transpose of DiracGamma[x]. Transpose[DiracGammaT[x]] gives DiracGamma[x]. Note that x must have Head LorentzIndex or Momentum.

See also: DiracGamma.

Examples

```
In[699]:= DiracGammaT[LorentzIndex[μ]]
Out[699]= γ_μ^T

In[700]:= Transpose[%]
Out[700]= γ^μ

In[701]:= GS[p]//FCI//Transpose
Out[701]= (γ · p)^T
```

DiracMatrix

Description

DiracMatrix[μ] denotes a Dirac gamma matrix with Lorentz index μ. DiracMatrix[μ, ν, ...] is a product of γ matrices with Lorentz indices μ, ν, ... DiracMatrix[5] is γ^5. DiracMatrix[6] is $1/2 + \gamma^5/2$. DiracMatrix[7] is $1/2 - \gamma^5/2$.

See also: DiracGammaExpand, DiracGamma, DiracSimplify, DiracSlash, DiracTrick, GA, GAD, GS, GSD.

Examples

```
In[702]:= DiracMatrix[μ]
Out[702]= γ^μ
```

This is how to enter the non-commutative product of two $\gamma^\mu\gamma^\nu$. The *Mathematica* Dot "." is used as non-commutative multiplication operator.

```
In[703]:= DiracMatrix[μ].DiracMatrix[ν]
Out[703]= γ^μ.γ^ν

In[704]:= DiracMatrix[α]//StandardForm
Out[704]= DiracGamma[LorentzIndex[α]]

In[705]:= DiracMatrix[μ]//FCE
Out[705]= γ^μ

In[706]:= %//StandardForm
Out[706]= GA[μ]

In[707]:= GAD[μ]
Out[707]= γ^μ

In[708]:= %//FCI//StandardForm
Out[708]= DiracGamma[LorentzIndex[μ, D], D]

In[709]:= GA[μ, ν, ρ]
Out[709]= γ^μ.γ^ν.γ^ρ

In[710]:= GA[a . b]//FCI
Out[710]= γ^a.γ^b

In[711]:= %//StandardForm
Out[711]= DiracGamma[LorentzIndex[a]].DiracGamma[LorentzIndex[b]]
```

DiracOrder

Description

DiracOrder[expr] orders the Dirac matrices in expr alphabetically. DiracOrder[expr,orderlist] orders the Dirac matrices in expr according to orderlist.

See also: DiracSimplify, DiracTrick.

Examples

```
In[712]:= t1 = GA[β, α]
Out[712]= γ^β . γ^α

In[713]:= DiracOrder[t1]
Out[713]= 2 g^αβ - γ^α . γ^β
```

This is a string of Dirac matrices in D dimensions.

```
In[714]:= t2 = GAD[μ, ν, μ]
Out[714]= γ^μ . γ^ν . γ^μ

In[715]:= DiracOrder[t1]
Out[715]= 2 g^αβ - γ^α . γ^β

In[716]:= t3 = GA[5, μ, ν]
Out[716]= γ^5 . γ^μ . γ^ν
```

By default γ^5 is moved to the right.

```
In[717]:= DiracOrder[t3]
Out[717]= γ^μ . γ^ν . γ^5

In[718]:= t4 = GA[6, μ, 7]
Out[718]= γ^6 . γ^μ . γ^7

In[719]:= DiracOrder[t4]
Out[719]= γ^μ . γ^7

In[720]:= t5 = GA[α, β, δ]
Out[720]= γ^α . γ^β . γ^δ
```

This orders the $\gamma^\alpha\gamma^\beta\gamma^\delta$ in reverse order.

```
In[721]:= DiracOrder[t5, {δ, β, α}]
Out[721]= -γ^δ . γ^β . γ^α + 2 γ^δ g^αβ - 2 γ^β g^αδ + 2 γ^α g^βδ

In[722]:= DiracOrder[%]
Out[722]= γ^α . γ^β . γ^δ

In[723]:= Clear[t1, t2, t3, t4, t5];
```

DiracReduce

Description

DiracReduce[exp] reduces all four-dimensional Dirac matrices in exp to the standard basis (S,P,V,A,T) using the Chisholm identity (see Chisholm). In the result the basic Dirac structures are wrapped with a head DiracBasis. I.e., S corresponds to DiracBasis[1], P : DiracBasis[DiracMatrix[5]], V: DiracBasis[DiracMatrix[mu]], A: DiracBasis[DiracMatrix[mu, 5]], T: DiracBasis[DiracSigma[DiracMatrix[mu, nu]]]. By default DiracBasis is substituted to Identity. Notice that the result of DiracReduce is given in the FeynCalcExternal-way, i.e.,evtl. you may have to use FeynCalcInternal on the result.

```
In[724]:= Options[DiracReduce]
Out[724]= {Factoring → False, FinalSubstitutions → {DiracBasis → Identity}}
```

See also: DiracSimplify.

Examples

```
In[725]:= t1 = GA[μ, ν]
Out[725]= γ^μ.γ^ν

In[726]:= DiracReduce[t1]
Out[726]= g^μν - i σ^μν

In[727]:= t2 = DiracMatrix[μ, ν, ρ]
Out[727]= γ^μγ^νγ^ρ

In[728]:= DiracReduce[t2]
Out[728]= i γ^$MU(1).γ^5 ϵ^μνρ$MU(1) + γ^ρ g^μν - γ^ν g^μρ + γ^μ g^νρ

In[729]:= t3 = DiracMatrix[μ, ν, ρ, σ]
Out[729]= γ^μγ^νγ^ργ^σ

In[730]:= t4 = DiracReduce[t3]
Out[730]= -i γ^5 ϵ^μνρσ - i σ^ρσ g^μν + i σ^νσ g^μρ - i σ^νρ g^μσ -
          i σ^μσ g^νρ + g^μσ g^νρ + i σ^μρ g^νσ - g^μρ g^νσ - i σ^μν g^ρσ + g^μν g^ρσ

In[731]:= t5 = Calc[DiracSimplify[DiracSigmaExplicit[t4.t4]]]
Out[731]= -128

In[732]:= Calc[t4.t4]
Out[732]= -6 D^3 + 17 D^2 - 10 D + 24

In[733]:= Clear[t1, t2, t3, t4]
```

DiracSigma

Description

DiracSigma[a, b] stands for i/2*(a . b - b . a) in 4 dimensions. a and b must have Head DiracGamma, DiracMatrix or DiracSlash. Only antisymmetry is implemented.

See also: DiracSigmaExplicit.

Examples

```
In[734]:= t1 = DiracSigma[GA[α], GA[β]]
Out[734]= σ^αβ

In[735]:= DiracSigmaExplicit[t1]
Out[735]= 1/2 i (γ^α.γ^β - γ^β.γ^α)

In[736]:= t2 = DiracSigma[GA[β], GA[α]]
Out[736]= -σ^αβ

In[737]:= t3 = DiracSigma[GS[p], GS[q]]
Out[737]= σ^pq

In[738]:= DiracSigmaExplicit[t3]
Out[738]= 1/2 i ((γ · p).(γ · q) - (γ · q).(γ · p))

In[739]:= Clear[t1, t2, t3]
```

DiracSigmaExplicit

Description

DiracSigmaExplicit[exp] inserts in exp for all DiracSigma its definition. DiracSigmaExplict is also an option of DiracSimplify.

See also: DiracSigma.

Examples

```
In[740]:= DiracSigma[GA[α], GA[β]]
Out[740]= σ^αβ

In[741]:= DiracSigmaExplicit[%]
Out[741]= 1/2 i (γ^α.γ^β - γ^β.γ^α)
```

DiracSimpCombine

Description

DiracSimpCombine is an option for DiracSimplify. If set to True, sums of DiracGamma's will be merged as much as possible in DiracGamma[.. + .. +]'s.

See also: DiracSimplify.

DiracSimplify

Description

DiracSimplify[expr] simplifies products of Dirac matrices in expr and expands non-commutative products. Double Lorentz indices and four vectors are contracted. The Dirac equation is applied. All DiracMatrix[5], DiracMatrix[6] and DiracMatrix[7] are moved to the right. The order of the other Dirac matrices is not changed.

```
In[742]:= Options[DiracSimplify]
Out[742]= {DiracCanonical → False, DiracSigmaExplicit → True, DiracSimpCombine → False, DiracSubstitute6
            Expanding → True, Factoring → False, FeynCalcInternal → False, InsideDiracTrace → False}
```

See also: Calc, DiracGammaExpand, DiracTrick.

Examples

This is a string of Dirac matrices in four dimensions.

```
In[743]:= t1 = GA[μ, ν, μ]
Out[743]= γ^μ.γ^ν.γ^μ

In[744]:= DiracSimplify[t1]
Out[744]= -2 γ^ν
```

This is a string of Dirac matrices in D dimensions.

```
In[745]:= t2 = GAD[μ, ν, μ]
Out[745]= γ^μ.γ^ν.γ^μ

In[746]:= DiracSimplify[t1]
Out[746]= -2 γ^ν

In[747]:= t3 = GA[5, μ, ν]
Out[747]= γ^5.γ^μ.γ^ν
```

By default $\gamma^5$is moved to the right.

```
In[748]:= DiracSimplify[t3]
Out[748]= γ^μ.γ^ν.γ^5
```

```
In[749]:= t4 = GA[6, μ, 7]
Out[749]= γ⁶.γ^μ.γ⁷

In[750]:= DiracSimplify[t4]
Out[750]= γ^μ.γ⁷

In[751]:= t5 = GS[a + b] . GS[p].GS[p].GS[c + d]
Out[751]= (γ · (a + b)).(γ · p).(γ · p).(γ · (c + d))
```

Contrary to DiracTrick DiracSimplify does non-commutative expansion.

```
In[752]:= DiracSimplify[t5]
Out[752]= (γ · a).(γ · c) p² + (γ · a).(γ · d) p² + (γ · b).(γ · c) p² + (γ · b).(γ · d) p²

In[753]:= DiracTrick[t5]
Out[753]= (γ · (a + b)).(γ · (c + d)) p²

In[754]:= t6 = SpinorVBar[p] . GS[p] . SpinorUBar[q]
Out[754]= ▽(p).(γ · p).u(q)

In[755]:= DiracSimplify[t6]
Out[755]= 0

In[756]:= GAD@@Join[{μ}, Table[ν_i, {i, 6}], {μ}]
Out[756]= γ^μ.γ^ν1.γ^ν2.γ^ν3.γ^ν4.γ^ν5.γ^ν6.γ^μ

In[757]:= DiracSimplify[%]
Out[757]= D γ^ν1.γ^ν2.γ^ν3.γ^ν4.γ^ν5.γ^ν6 − 12 γ^ν1.γ^ν2.γ^ν3.γ^ν4.γ^ν5.γ^ν6 + 4 γ^ν3.γ^ν4.γ^ν5.γ^ν6 g^ν1ν2 −
            4 γ^ν2.γ^ν4.γ^ν5.γ^ν6 g^ν1ν3 + 4 γ^ν2.γ^ν3.γ^ν5.γ^ν6 g^ν1ν4 − 4 γ^ν2.γ^ν3.γ^ν4.γ^ν6 g^ν1ν5 +
            4 γ^ν2.γ^ν3.γ^ν4.γ^ν5 g^ν1ν6 + 4 γ^ν1.γ^ν4.γ^ν5.γ^ν6 g^ν2ν3 − 4 γ^ν1.γ^ν3.γ^ν5.γ^ν6 g^ν2ν4 +
            4 γ^ν1.γ^ν3.γ^ν4.γ^ν6 g^ν2ν5 − 4 γ^ν1.γ^ν3.γ^ν4.γ^ν5 g^ν2ν6 + 4 γ^ν1.γ^ν2.γ^ν5.γ^ν6 g^ν3ν4 − 4 γ^ν1.γ^ν2.γ^ν4.γ^ν6 g^ν3ν5 +
            4 γ^ν1.γ^ν2.γ^ν4.γ^ν5 g^ν3ν6 + 4 γ^ν1.γ^ν2.γ^ν3.γ^ν6 g^ν4ν5 − 4 γ^ν1.γ^ν2.γ^ν3.γ^ν5 g^ν4ν6 + 4 γ^ν1.γ^ν2.γ^ν3.γ^ν4 g^ν5ν6
```

With the option DiracCanonical an alphabetic ordering is done.

```
In[758]:= DiracSimplify[GA[ν, μ], DiracCanonical → True]
Out[758]= 2 g^μν − γ^μ.γ^ν
```

Setting InsideDiracTrace→True assumes that a trace is still to be taken later on.

```
In[759]:= DiracSimplify[GA[μ, ν, ρ, σ], InsideDiracTrace → True]
Out[759]= g^μσ g^νρ − g^μρ g^νσ + g^μν g^ρσ

In[760]:= DiracSimplify[GA[μ, ν, ρ], InsideDiracTrace → True]
Out[760]= 0

In[761]:= Clear[t1, t2, t3, t4, t5, t6]
```

DiracSimplify2

Description

DiracSimplify2[exp] simplifies the Dirac structure but leaves any γ^5 untouched.
See also: DiracSimplify.

Examples

```
In[762]:= GAD[μ, ν, μ, 5, α, β, α]
Out[762]= γ^μ.γ^ν.γ^μ.γ^5.γ^α.γ^β.γ^α

In[763]:= DiracSimplify2[%]
Out[763]= γ^μ.γ^ν.γ^μ.γ^5.γ^α.γ^β.γ^α
```

DiracSlash

Description

DiracSlash[p] is the contraction $p^\mu \gamma_\mu$ (FourVector[p, μ] DiracMatrix[μ]). Products of those can be entered in the form DiracSlash[p1, p2, ...].

```
In[764]:= Options[DiracSlash]
Out[764]= {Dimension → 4, FeynCalcInternal → False}
```

See also: DiracGammaExpand, DiracGamma, DiracMatrix, DiracSimplify, DiracTrick, GS, GSD.

Examples

This is q-slash, i.e., $\gamma^\mu q_\mu$.

```
In[765]:= DiracSlash[q]
Out[765]= γ · q

In[766]:= DiracSlash[p].DiracSlash[q]
Out[766]= (γ · p).(γ · q)

In[767]:= DiracSlash[p, q]
Out[767]= (γ · p).(γ · q)

In[768]:= GS[p]
Out[768]= γ · p

In[769]:= DiracSlash[q]//StandardForm
Out[769]= DiracSlash[q]

In[770]:= DiracSlash[q, Dimension → n]//StandardForm
Out[770]= DiracSlash[q, Dimension → n]
```

DiracSpinor

Description

DiracSpinor is simply a quantity defined as noncommutative (with DeclareNonCommutative[DiracSpinor]). The convention intended is that DiracSpinor[p, m, ind] is a Dirac spinor for a fermion with momentum p and mass m and indices ind.

See also: Spinor.

DiracSubstitute67

Description

DiracSubstitute67 is an option for DiracSimplify. If set to True the chirality-projectors DiracGamma[6] and DiracGamma[7] are substituted by their definitions.

See also: DiracGamma, DiracSimplify.

DiracTrace

Description

DiracTrace[expr] is the head of Dirac traces. Whether the trace is evaluated depends on the option DiracTraceEvaluate. Direct trace evaluation should be performed with Tr. The argument expr may be a product of Dirac matrices or slashes separated by the Mathematica Dot (.).

```
In[771]:= Options[DiracTrace]
Out[771]= {EpsContract → False, Factoring → False, FeynCalcExternal → False, Mandelstam → {},

            PairCollect → True, DiracTraceEvaluate → False, Schouten → 0, LeviCivitaSign → -1, TraceOfOne →
```

For comments regarding γ^5schemes see the notes for Tr.

See also: Tr.

Examples

```
In[772]:= DiracTrace[GA[μ, ν]]
Out[772]= tr(γ^μ.γ^ν)

In[773]:= DiracTrace[GA[μ, ν, ρ, σ]]
Out[773]= tr(γ^μ.γ^ν.γ^ρ.γ^σ)

In[774]:= % /. DiracTrace → Tr
Out[774]= 4 (g^μσ g^νρ - g^μρ g^νσ + g^μν g^ρσ)
```

```
In[775]:= DiracTrace[GA[μ, ν, ρ, σ, 5], DiracTraceEvaluate → True]
Out[775]= -4 i ϵ^(μνρσ)

In[776]:= DiracTrace[GA[μ, ν, ρ, σ, δ, τ, 5], DiracTraceEvaluate → True]
Out[776]= 4 (i ϵ^(νρστ) g^(δμ) - i ϵ^(μρστ) g^(δν) + i ϵ^(μνστ) g^(δρ) - i ϵ^(μνρτ) g^(δσ) -
                i ϵ^(μνρσ) g^(δτ) - i ϵ^(δρστ) g^(μν) + i ϵ^(δνστ) g^(μρ) - i ϵ^(δνρτ) g^(μσ) + i ϵ^(δνρσ) g^(μτ) -
                i ϵ^(δμστ) g^(νρ) + i ϵ^(δμρτ) g^(νσ) - i ϵ^(δμρσ) g^(ντ) - i ϵ^(δμντ) g^(ρσ) + i ϵ^(δμνσ) g^(ρτ) - i ϵ^(δμνρ) g^(στ))

In[777]:= DiracTrace[GS[p, q, r, s]]
Out[777]= tr((γ · p).(γ · q).(γ · r).(γ · s))

In[778]:= DiracTrace[GA[μ, ν], DiracTraceEvaluate → True, FCE → True]
Out[778]= 4 g^(μν)

In[779]:= %//StandardForm
Out[779]= 4 MT[μ, ν]
```

DiracTraceEvaluate

Description

DiracTraceEvaluate is an option for DiracTrace and Tr. If set to False, DiracTrace remains unevaluated.
See also: DiracTrace, Tr.

DiracTrick

Description

DiracTrick[exp] contracts gamma matrices with each other and performs several simplifications, but no expansion, use Calc or DiracSimplify for non-commutative expansion.

```
In[780]:= Options[DiracTrick]
Out[780]= {Expanding → False}
```

See also: Calc, DiracGammaExpand, DiracSimplify.

Examples

This is a string of Dirac matrices in four dimensions.

```
In[781]:= t1 = GA[μ, ν, μ]
Out[781]= γ^μ.γ^ν.γ^μ

In[782]:= DiracTrick[t1]
```

```
Out[782]= -2 γ^ν
```

This is a string of Dirac matrices in D dimensions.

```
In[783]:= t2 = GAD[μ, ν, μ]
Out[783]= γ^μ.γ^ν.γ^μ

In[784]:= DiracTrick[t2]
Out[784]= (2 - D) γ^ν

In[785]:= t3 = GA[5, μ, ν]
Out[785]= γ^5.γ^μ.γ^ν
```

By default $\gamma^5$is moved to the right.

```
In[786]:= DiracTrick[t3]
Out[786]= γ^μ.γ^ν.γ^5

In[787]:= t4 = GA[6, μ, 7]
Out[787]= γ^6.γ^μ.γ^7

In[788]:= DiracTrick[t4]
Out[788]= γ^μ.γ^7

In[789]:= t5 = GS[a + b] . GS[p].GS[p].GS[c + d]
Out[789]= (γ · (a + b)).(γ · p).(γ · p).(γ · (c + d))

In[790]:= DiracTrick[t5]
Out[790]= (γ · (a + b)).(γ · (c + d)) p^2

In[791]:= Calc[t5]
Out[791]= (γ · a).(γ · c) p^2 + (γ · a).(γ · d) p^2 + (γ · b).(γ · c) p^2 + (γ · b).(γ · d) p^2

In[792]:= GAD@@Join[{μ}, Table[ν_i, {i, 6}], {μ}]
Out[792]= γ^μ.γ^ν1.γ^ν2.γ^ν3.γ^ν4.γ^ν5.γ^ν6.γ^μ

In[793]:= DiracTrick[%]
Out[793]= (D - 12) γ^ν1.γ^ν2.γ^ν3.γ^ν4.γ^ν5.γ^ν6 -
              4 (-γ^ν3.γ^ν4.γ^ν5.γ^ν6 g^ν1ν2 + γ^ν2.γ^ν4.γ^ν5.γ^ν6 g^ν1ν3 - γ^ν2.γ^ν3.γ^ν5.γ^ν6 g^ν1ν4 + γ^ν2.γ^ν3.γ^ν4.γ^ν6 g^ν1ν5 -
                 γ^ν2.γ^ν3.γ^ν4.γ^ν5 g^ν1ν6 - γ^ν1.γ^ν4.γ^ν5.γ^ν6 g^ν2ν3 + γ^ν1.γ^ν3.γ^ν5.γ^ν6 g^ν2ν4 -
                 γ^ν1.γ^ν3.γ^ν4.γ^ν6 g^ν2ν5 + γ^ν1.γ^ν3.γ^ν4.γ^ν5 g^ν2ν6 - γ^ν1.γ^ν2.γ^ν5.γ^ν6 g^ν3ν4 + γ^ν1.γ^ν2.γ^ν4.γ^ν6 g^ν3ν5 -
                 γ^ν1.γ^ν2.γ^ν4.γ^ν5 g^ν3ν6 - γ^ν1.γ^ν2.γ^ν3.γ^ν6 g^ν4ν5 + γ^ν1.γ^ν2.γ^ν3.γ^ν5 g^ν4ν6 - γ^ν1.γ^ν2.γ^ν3.γ^ν4 g^ν5ν6)
```

Divideout

Description

Divideout is an option for OPEInt and OPEInsert. The setting is divided out at the end.
See also: OPEInt, OPEInsert.

DOT

Description

DOT[a, b, ...] is the FeynCalc function for non-commutative multiplication. By default it is set to the Mathematica Dot functions. By setting
DOT=.
this can be disabled. Note that then non-commutative products should to be entered like DOT[DiracMatrix[mu], m + DiracSlash[p], DiracMatrix[mu]], etc.

See also: DotSimplify.

DotExpand

Description

DotExpand[expr] expands DOT products in expr.

See also: DOT, DotSimplify, DeclareNonCommutative, UnDeclareNonCommutative.

Examples

```
In[794]:= DotExpand[DOT[a x + b y + c z, d + e + f]]
Out[794]= a d x + a e x + a f x + b d y + b e y + b f y + c d z + c e z + c f z

In[795]:= DeclareNonCommutative/@{a, b, c, d, e, f};

In[796]:= DotExpand[DOT[a x + b y + c z, d + e + f]]
Out[796]= x a.d + x a.e + x a.f + y b.d + y b.e + y b.f + z c.d + z c.e + z c.f

In[797]:= UnDeclareNonCommutative/@{a, b, c, d, e, f};

In[798]:= DotExpand[DOT[a x + b y + c z, d + e + f]]
Out[798]= a d x + a e x + a f x + b d y + b e y + b f y + c d z + c e z + c f z
```

DotPower

Description

DotPower is an option for DotSimplify. It determines whether non-commutative powers are represented by succesive multiplication or by Power.

See also: DotSimplify.

DotProduct

Description

DotProduct[x, y] denotes the three-dimensional dot-product. If x and y have Head List, DotProduct[x, a] (where a is a vector) performs Sum[x[[k]] a[[k]], {k, 0, 3}].

See also: CrossProduct, ThreeVector.

Examples

```
In[799]:= DotProduct[ThreeVector[a], 3ThreeVector[b]]
Out[799]= 3 a⃗ · b⃗
```

DotSimplifyRelations

Description

DotSimplifyRelations is an option for DotSimplify. Its setting may be a list of substitution rules of the form DotSimplifyRelations → {a . b → c, b^2 → 0, ...}.

See also: DotSimplify.

DotSimplify

Description

DotSimplify[expr] expands and reorders noncommutative terms in expr. Simplifying relations may be specified by the option DotSimplifyRelations or by Commutator and AntiCommutator definitions. Whether expr is expanded noncommutatively depends on the option Expanding.

```
In[800]:= Options[DotSimplify]
Out[800]= {Expanding → True, DotSimplifyRelations → {}, DotPower → False}
```

See also: AntiCommutator, Commutator, Calc.

Examples

```
In[801]:= t1 = GA[μ].(2 GS[p] - GS[q]).GA[ν]
Out[801]= γ^μ.(2 γ · p - γ · q).γ^ν

In[802]:= DotSimplify[t1]
Out[802]= 2 γ^μ.(γ · p).γ^ν - γ^μ.(γ · q).γ^ν

In[803]:= DeclareNonCommutative[a, b, c]
```

```
In[804]:= t2 = a.(b - z c).a
Out[804]= a.(b - c z).a

In[805]:= DotSimplify[t2]
Out[805]= a.b.a - z a.c.a

In[806]:= Commutator[a, c] = 1
Out[806]= 1

In[807]:= DotSimplify[t2]
Out[807]= a.b.a - z (a + c.a.a)

In[808]:= Commutator[a, c] = .

In[809]:= DotSimplify[t2]
Out[809]= a.b.a - z a.c.a

In[810]:= AntiCommutator[b, a] = c
Out[810]= c

In[811]:= DotSimplify[t2]
Out[811]= a.c - a.a.b - z a.c.a

In[812]:= AntiCommutator[b, a] = .

In[813]:= DotSimplify[t2, DotSimplifyRelations → {a.c → 1/z}]
Out[813]= a.b.a - a

In[814]:= DeclareNonCommutative[x]

In[815]:= DotSimplify[x.x.x]
Out[815]= x.x.x

In[816]:= DotSimplify[x.x.x, DotPower → False]
Out[816]= x.x.x

In[817]:= UnDeclareNonCommutative[a, b, c, x]
```

DummyIndex

Description

DummyIndex is an option of CovariantD specifying an index to use as dummy summation index. If set to Automatic, unique indices are generated

See also: CovariantD.

D0

Description

D0[p10, p12, p23, p30, p20, p13, m1^2, m2^2, m3^2, m4^2] is the Passarino-Veltman D_0 function. The convention for the arguments is that if the denominator of the integrand has the form ([q^2-m1^2] [(q+p1)^2-m2^2] [(q+p2)^2-m3^2] [(q+p3)^2-m4^2]), the first six arguments of D0 are the scalar products p10 = p1^2, p12 = (p1-p2)^2, p23 = (p2-p3)^2, p30 = p3^2, p20 = p2^2, p13 = (p1-p3)^2.

See also: B0, C0, PaVe, PaVeOrder.

Examples

```
In[818]:= D0[p10, p12, p23, p30, p20, p13, m1^2, m2^2, m3^2, m4^2]
Out[818]= D0(p10, p12, p23, p30, p20, p13, m1^2, m2^2, m3^2, m4^2)

In[819]:= PaVeOrder[D0[p10, p12, p23, p30, p20, p13, m1^2, m2^2, m3^2, m4^2],
            PaVeOrderList -> {p13, p20}]
Out[819]= D0(p10, p30, p23, p12, p13, p20, m2^2, m1^2, m4^2, m3^2)

In[820]:= PaVeOrder[%]
Out[820]= D0(p10, p12, p23, p30, p20, p13, m1^2, m2^2, m3^2, m4^2)
```

D0Convention

Description

D0Convention is an option for Write2. If set to 1, the convention for the arguments of D0 is changed when writing a Fortran file with Write2: The fifth and sixth argument of D0 are interchanged and the square root is taken of the last four arguments.

See also: D0, Write2.

Eps

Description

Eps[a, b, c, d] is the head of the totally antisymmetric ϵ (Levi-Civita) tensor. The a,b, ... may have head LorentzIndex, Momentum or Integer. In case of integers the Levi-Civita tensor is evaluated immediately.

```
In[821]:= Options[Eps]
Out[821]= {Dimension -> 4}
```

See also: EpsEvaluate, LC, LCD, LeviCivita.

Examples

```
In[822]:= Eps[LorentzIndex[μ], LorentzIndex[ν], LorentzIndex[ρ], LorentzIndex[σ]]
Out[822]= ϵ^μνρσ

In[823]:= Eps[Momentum[p], LorentzIndex[ν], LorentzIndex[ρ], LorentzIndex[σ]]
Out[823]= ϵ^pνρσ

In[824]:= Eps[b, a, c, d]//StandardForm
Out[824]= -Eps[a, b, c, d]

In[825]:= Eps[0, 1, 2, 3]
Out[825]= 1

In[826]:= Eps[1, 0, 2, 3]
Out[826]= -1

In[827]:= SetOptions[Eps, Dimension → 4];

In[828]:= a1 =
            Eps[LorentzIndex[μ, D], LorentzIndex[ν, D], LorentzIndex[ρ, D], LorentzIndex[σ, D]]
Out[828]= ϵ^μνρσ

In[829]:= Contract[a1a1]
Out[829]= -24

In[830]:= SetOptions[Eps, Dimension → D];

In[831]:= a2 =
            Eps[LorentzIndex[μ, D], LorentzIndex[ν, D], LorentzIndex[ρ, D], LorentzIndex[σ, D]]
Out[831]= ϵ^μνρσ

In[832]:= Contract[a2a2]//Factor2
Out[832]= (1 - D) (2 - D) (3 - D) D

In[833]:= g5 = -(i/24) LCD[μ, ν, ρ, α].GAD[μ, ν, ρ, α]//FCI
Out[833]= -(1/24) i ϵ^μνρα . γ^μ . γ^ν . γ^ρ . γ^α

In[834]:= g5p = -(i/24) LCD[μ', ν', ρ', α'].GAD[μ', ν', ρ', α']//FCI
Out[834]= -(1/24) i ϵ^μ'ν'ρ'α' . γ^μ' . γ^ν' . γ^ρ' . γ^α'

In[835]:= g52 = Factor2[Calc[g5.g5p]]
Out[835]= -(1/24) (1 - D) (2 - D) (3 - D) D

In[836]:= g52/.D → 4
Out[836]= 1

In[837]:= Clear[a1, a2, g5, g5p, g52]
```

EpsChisholm

Description

EpsChisholm[expr] substitutes for a gamma matrix contracted with a Levi-Civita tensor (Eps) the Chisholm identity.

See also: Chisholm.

Examples

```
In[838]:= Chisholm[GA[μ, ν, ρ, σ]]
Out[838]= -i γ^$MU$372 . γ^σ . γ^5 ϵ^μνρ$MU$372 + γ^ρ . γ^σ g^μν - γ^ν . γ^σ g^μρ + γ^μ . γ^σ g^νρ

In[839]:= EpsChisholm[%]
Out[839]= γ^μ . γ^ν . γ^ρ . γ^σ
```

EpsContract

Description

EpsContract is an option of Contract specifying whether Levi-Civita tensors Eps[...] will be contracted, i.e., products of two Eps are replaced via the determinant formula.

See also: Eps, Contract.

Examples

```
In[840]:= a1 =

            Eps[LorentzIndex[μ, D], LorentzIndex[ν, D], LorentzIndex[ρ, D], LorentzIndex[σ, D]]
Out[840]= ϵ^μνρσ

In[841]:= Contract[a1 a1, EpsContract → False]
Out[841]= (ϵ^μνρσ)^2

In[842]:= Contract[a1 a1, EpsContract → True]
Out[842]= -D^4 + 6 D^3 - 11 D^2 + 6 D

In[843]:= Clear[a1]
```

EpsDiscard

Description

EpsDiscard is an option for FeynCalc2FORM. If set to True all Levi-Civita tensors are replaced by 0 after contraction.
See also: FeynCalc2FORM.

EpsEvaluate

Description

EpsEvaluate[expr] applies total antisymmetry and linearity (w.r.t. Momentum's) to all Levi-Civita tensors (Eps') in expr.
See also: Contract, Eps, LeviCivita, Trick.

Examples

```
In[844]:= Trick[LeviCivita[μ, ν, ρ, σ] FourVector[p + q, σ]]
Out[844]= ϵ^(μνρp+q)

In[845]:= EpsEvaluate[%]
Out[845]= ϵ^(μνρp) + ϵ^(μνρq)

In[846]:= StandardForm[%]
Out[846]= Eps[LorentzIndex[μ, D], LorentzIndex[ν, D], LorentzIndex[ρ, D], Momentum[p]]+
            Eps[LorentzIndex[μ, D], LorentzIndex[ν, D], LorentzIndex[ρ, D], Momentum[q]]
```

EpsUncontract

Description

EpsUncontract does Uncontract on scalar products involving Eps.
See also: Eps, Uncontract.

Epsilon

Description

Epsilon is (n-4), where n is the space-time dimension. Epsilon stands for a small positive number.
See also: Series2.

Examples

```
In[847]:= Epsilon
Out[847]= ε
```

Epsilon has no functional properties, but some upvalues are changed:

```
In[848]:= {Re[Epsilon] > -4, Re[Epsilon] > -3,
            Re[Epsilon] > -2, Re[Epsilon] > -1, Re[Epsilon] > 0}
Out[848]= {True, True, True, True, True}
```

EpsilonOrder

Description

EpsilonOrder is an option of OPEIntDelta and RHI. The setting determines the order n (Epsilon^n) which should be kept.

See also: OPEIntDelta, RHI.

Expanding

Description

Expanding is an option for Calc, Contract, DiracSimplify,DotSimplify, SUNSimplify, etc. As option for Contract it specifies whether expansion w.r.t. LorentzIndex is done BEFORE contraction. If set to False in DiracSimplify or SUNSimplify, only a limited set of simplifications (multiplicative linearity etc.) is performed.

See also: Calc, Contract, DiracSimplify, DotSimplify, SUNSimplify.

ExpandPartialD

Description

ExpandPartialD[exp] expands all products of QuantumField's and partial differentiation operators in exp and applies the Leibniz rule.

```
In[849]:= Options[ExpandPartialD]
```

```
Out[849]= {PartialDRelations → {a$___.∂/∂x$_^mu$_.b$___ :↦ a$.∂_mu$(Dot[b$]),
        UMatrix(Idl, ___)*(PartialD|RightPartialD|LeftPartialD)[__] :↦ 0, a___*D_mu_*b___ :↦
          a*𝒟_μ(b)/; (go = False; {b}/.__|_[___, __, ___][y_?AtomQ] :↦ (go = True; z = y); go),
        a___*∂/Dx_^mu_*b___ :↦ a*𝒟_μ(b), a___*(←∂)_mu_*b___ :↦
          ∂_μ(a)*b/; (go = False; {a}/.__|_[___, __, ___][y_?AtomQ] :↦ (go = True; z = y); go),
        a___*LeftPartialD(x_, mu_)*b___ :↦ ∂_μ(a)*b, a___*(PartialD|RightPartialD)[mu_]*b___ :↦
          a*∂_μ(b)/; (go = False; {b}/.__|_[___, __, ___][y_?AtomQ] :↦ (go = True; z = y); go),
        a___*(PartialD|RightPartialD)[x_, mu_]*b___ :↦ a*∂_μ(b)}}
```

See also: ExplicitPartialD, LeftPartialD, LeftRightPartialD, PartialDRelations, RightPartialD.

Examples

```
In[850]:= RightPartialD[μ].QuantumField[A, LorentzIndex[μ]].QuantumField[A, LorentzIndex[ν]]
Out[850]= (→∂)_μ.A_μ.A_ν

In[851]:= ExpandPartialD[%]
Out[851]= A_μ.∂_μA_ν + ∂_μA_μ.A_ν

In[852]:= %//StandardForm
Out[852]= QuantumField[A, LorentzIndex[μ]].
            QuantumField[PartialD[LorentzIndex[μ]], A, LorentzIndex[ν]]+
          QuantumField[PartialD[LorentzIndex[μ]], A, LorentzIndex[μ]].
            QuantumField[A, LorentzIndex[ν]]

In[853]:= LeftRightPartialD[μ].QuantumField[A, LorentzIndex[ν]]
Out[853]= (⇆∂)_μ.A_ν

In[854]:= ExpandPartialD[%]
Out[854]= ∂_μA_ν/2 - 1/2 (←∂)_μ.A_ν

In[855]:= QuantumField[A, LorentzIndex[μ]].
            (LeftRightPartialD[OPEDelta]^2).QuantumField[A, LorentzIndex[ρ]]
Out[855]= A_μ.⇆∂_Δ^2.A_ρ

In[856]:= ExpandPartialD[%]
Out[856]= 1/4 A_μ.(∂_Δ∂_ΔA_ρ) - 1/2 ∂_ΔA_μ.∂_ΔA_ρ + 1/4 (∂_Δ∂_ΔA_μ).A_ρ

In[857]:= 8 LeftRightPartialD[OPEDelta]^3
Out[857]= 8 ⇆∂_Δ^3
```

```
In[858]:= ExplicitPartialD[%]
Out[858]= (Dt[] - Dt[])^3

In[859]:= ExpandPartialD[%]
Out[859]= -Dt[].Dt[].Dt[] + 3 Dt[].Dt[].Dt[] - 3 Dt[].Dt[].Dt[] + Dt[].Dt[].Dt[]

In[860]:= LeviCivita[μ, ν, ρ, τ] RightPartialD[α, μ, β, ν]
Out[860]= Dt[].Dt[].Dt[].Dt[] Eps^(μνρτ)

In[861]:= ExpandPartialD[%]
Out[861]= 0
```

ExpandScalarProduct, ScalarProductExpand

Description

ExpandScalarProduct[expr] expands scalar products of sums of momenta in expr. ExpandScalarProduct does not use Expand on expr.
ScalarProductExpand=ExpandScalarProduct.

```
In[862]:= ExpandScalarProduct//Options
Out[862]= {FeynCalcInternal → True}
```

See also: Calc, MomentumExpand, MomentumCombine.

Examples

```
In[863]:= SP[p1 + p2 + p3, p4 + p5 + p6]
Out[863]= (p1 + p2 + p3) · (p4 + p5 + p6)

In[864]:= %//ScalarProductExpand
Out[864]= p1 · p4 + p1 · p5 + p1 · p6 + p2 · p4 + p2 · p5 + p2 · p6 + p3 · p4 + p3 · p5 + p3 · p6

In[865]:= SP[p, p - q]
Out[865]= p · (p - q)

In[866]:= ExpandScalarProduct[%]
Out[866]= p^2 - p · q

In[867]:= FV[p - q, μ]
Out[867]= p - q^μ

In[868]:= ExpandScalarProduct[%]
Out[868]= p^μ - q^μ

In[869]:= SP[p - q, q - r]//FCI
Out[869]= (p - q) · (q - r)

In[870]:= %/.Pair → ExpandScalarProduct
Out[870]= p · q - p · r - q^2 + q · r
```

Expand2

Description

Expand2[exp, x] expands all sums containing x. Expand2[exp, {x1, x2, ...}] expands all sums containing x1, x2, ...

Examples

```
In[871]:= Expand2[(x1 + x2 + x3)(2x1 + 3x2) + (y1 + y2 + y3)(2y1 + 3y2), {x1, x2}]
Out[871]= 2 x1^2 + 5 x2 x1 + 2 x3 x1 + 3 x2^2 + 3 x2 x3 + (2 y1 + 3 y2) (y1 + y2 + y3)
```

Explicit

Description

Explicit is an option for FieldStrength, GluonVertex, SUNF, and Twist2GluonOperator. If set to True the full form of the operator is inserted. Explicit[exp] inserts explicit expressions of GluonVertex, Twist2GluonOperator, etc., in exp. SUNF's are replaced by SUNTrace objects.

See also: GluonVertex, Twist2GluonOperator.

```
In[872]:= Explicit//Options
Out[872]= {CouplingConstant → g_s, Dimension → D, Gauge → 1, Ω → False}
```

Examples

```
In[873]:= GluonPropagator[p, μ, ν]
Out[873]= Π_g^{μν}(p)

In[874]:= Explicit[%]
Out[874]= -(i g^{μν} / p^2)

In[875]:= Explicit[GluonPropagator[p, μ, ν], Gauge → ξ]
Out[875]= -i ξ p^μ p^ν (1/p^2)^2 + i p^μ p^ν (1/p^2)^2 - i g^{μν} / p^2

In[876]:= GluonVertex[p, μ, a, q, ν, b, r, ρ, c]
Out[876]= V^{μνρ}(p, q, r) f_abc

In[877]:= Explicit[%]
Out[877]= (g_s q^μ g^{νρ} - g_s r^μ g^{νρ} - g_s g^{μρ} p^ν + g_s g^{μρ} r^ν + g_s g^{μν} p^ρ - g_s g^{μν} q^ρ) f_abc

In[878]:= Twist2GluonOperator[p, μ, a, ν, b]
Out[878]= 1/2 ((-1)^m + 1) δ_ab (O^{G2}_{μν}(p))

In[879]:= Explicit[%]
```

```
Out[879]= 1/2 (g^μν Δ · p^2 - (p^μ Δ^ν + Δ^μ p^ν) Δ · p + Δ^μ Δ^ν p^2) ((-1)^m + 1) (Δ · p)^(m-2) δ_ab

In[880]:= FieldStrength[μ, ν, a]
Out[880]= F^a_μν

In[881]:= Explicit[%]
Out[881]= ∂_μ A^a_ν - ∂_ν A^a_μ + g_s A^b1_μ . A^c11_ν f_ab1c11
```

ExplicitLorentzIndex

Description

ExplicitLorentzIndex[ind] is an explicit Lorentz index, i.e., ind is an integer.
See also: LorentzIndex, Pair.

Examples

```
In[882]:= Pair[LorentzIndex[1], LorentzIndex[μ]]
Out[882]= g^1μ

In[883]:= %//StandardForm
Out[883]= Pair[ExplicitLorentzIndex[1], LorentzIndex[μ]]
```

ExplicitPartialD

Description

ExplicitPartialD[exp] inserts in exp the definition for LeftRightPartialD[z] (and LeftRightPartialD2[z]).
See also: ExpandPartialD, LeftRightPartialD, LeftRightPartialD2.

Examples

```
In[884]:= ExplicitPartialD[ LeftRightPartialD[μ] ]
Out[884]= 1/2 ( (∂⃗)_μ - (∂⃖)_μ )

In[885]:= ExplicitPartialD[ LeftRightPartialD2[μ] ]
Out[885]= (∂⃖)_μ + (∂⃗)_μ

In[886]:= ExplicitPartialD[ LeftRightPartialD[OPEDelta] ]
Out[886]= 1/2 ( (∂⃗)_Δ - (∂⃖)_Δ )

In[887]:= 16 LeftRightPartialD[OPEDelta]^4
Out[887]= 16 ∂⃡^4_Δ
```

```
In[888]:= ExplicitPartialD[%]
Out[888]= ( (→∂)_Δ - (←∂)_Δ )^4
```

ExplicitSUNIndex

Description

ExplicitSUNIndex[ind] is a specific SU(*N*) index, i.e., ind is an integer.
See also: FCI, SUNDelta, SUNF, SUNIndex.

Examples

```
In[889]:= ExplicitSUNIndex[1]
Out[889]= 1

In[890]:= SUNDelta[1, a]//FCI//StandardForm
Out[890]= SUNDelta[ExplicitSUNIndex[1], SUNIndex[a]]
```

ExtraFactor

Description

ExtraFactor is an option for FermionSpinSum. The setting ExtraFactor → fa multiplies the whole amplitude with the factor fa before squaring.
See also: FermionSpinSum.

ExtraVariables

Description

ExtraVariables is an option for OneLoopSum; it may be set to a list of variables which are also bracketed out in the result, just like B0, C0, D0 and PaVe.
See also: OneLoop, OneLoopSum.

FactorFull

Description

FactorFull is an option of Factor2 (default False). If set to False, products like (*a*-*b*) (*a*+*b*) will be replaced by (a^2-b^2).

See also: Factor2.

Factoring

Description

Factoring is an option for Collect2, Contract, Tr and more functions. If set to True, the result will be factored, using Factor2. If set to any function f, this function will be used.
See also: Collect2, Contract, Tr.

Factorout

Description

Factorout is an option for OPEInt.
See also: OPEInt.

FactorTime

Description

FactorTime is an option for Factor2. It denotes the maximum time (in seconds) during which Factor2 tries to factor.
See also: Factor2.

Factor1

Description

Factor1[poly] factorizes common terms in the summands of poly. It uses basically PolynomialGCD.
See also: Factor2.

Examples

```
In[891]:= t1 = (a - x)(b - x)
Out[891]= (a - x) (b - x)

In[892]:= t2 = {Factor1[t1], Factor[t1]}
Out[892]= {(a - x) (b - x), -(a - x) (x - b)}

In[893]:= t3 = Expand[(a - b)(a + b)]
```

```
Out[893]= a^2 - b^2

In[894]:= Factor[t3]
Out[894]= (a - b) (a + b)

In[895]:= Factor1[t3]
Out[895]= a^2 - b^2

In[896]:= Clear[t1, t2, t3]
```

Factor2

Description

Factor2[poly] factors a polynomial in a standard way. Factor2 works sometimes better than Factor on polynomials involving rationals with sums in the denominator. Factor2 uses Factor internally and is in general slower than Factor. There are four possible settings of the option Method (0,1,2,3). In general Factor will work faster than Factor2.

```
In[897]:= Options[Factor2]
Out[897]= {FactorFull → False, Method → 3}
```

See also: Collect2.

Examples

```
In[898]:= t1 = (a - x) (b - x)
Out[898]= (a - x) (b - x)

In[899]:= t2 = {Factor2[t1], Factor[t1]}
Out[899]= {(a - x) (b - x), -(a - x) (x - b)}

In[900]:= t3 = Expand[(a - b) (a + b)]
Out[900]= a^2 - b^2

In[901]:= Factor[t3]
Out[901]= (a - b) (a + b)

In[902]:= Factor2[t3]
Out[902]= a^2 - b^2

In[903]:= Factor2[t3, FactorFull → True]
Out[903]= (a - b) (a + b)

In[904]:= Clear[t1, t2, t3]
```

FAD

Description

FAD is the FeynCalc external form of FeynAmpDenominator and denotes an inverse propagator. FAD[q, q-p, ...] is 1/(q^2 (q-p)^2 ...). FAD[{q1,m}, {q1-p,m}, q2, ...] is 1/((q1^2 - m^2) ((q1-p)^2 - m^2) q2^2 ...). Translation into FeynCalc internal form is performed by FeynCalcInternal.

See also: FAD, FCE, FCI, FeynAmpDenominator, FeynAmpDenominatorSimplify, PropagatorDenominator.

Examples

```
In[905]:= FAD[q, p - q]
Out[905]= 1/(([(p - q)^2]) ([q^2]))

In[906]:= FAD[p, {p - q, m}]
Out[906]= 1/(([p^2]) ([(p - q)^2 - m^2]))

In[907]:= FAD[q, p - q]//FCI//FCE//StandardForm
Out[907]= FAD[q, p - q]

In[908]:= FAD[q, p - q]//FCI//StandardForm
Out[908]= FeynAmpDenominator[PropagatorDenominator[Momentum[q, D], 0],
            PropagatorDenominator[Momentum[p, D] - Momentum[q, D], 0]]

In[909]:= FAD[p] FAD[p - q] //FeynAmpDenominatorCombine//StandardForm
Out[909]= FeynAmpDenominator[PropagatorDenominator[Momentum[p, D], 0],
            PropagatorDenominator[Momentum[p, D] - Momentum[q, D], 0]]
```

FC

Description

FC changes the output format to FeynCalcForm. To change to InputForm use FI.

See also: FeynCalcForm, FI, FeynCalcExternal, FeynCalcInternal.

Examples

```
In[910]:= FI

In[911]:= {DiracGamma[5], DiracGamma[Momentum[p]]}
Out[911]= {DiracGamma[5], DiracGamma[Momentum[p]]}

In[912]:= FC
```

```
In[913]:= {DiracGamma[5], DiracGamma[Momentum[p]]}
Out[913]= {γ^5, γ · p}
```

FCE

Description

FCE[exp] translates exp from the internal FeynCalc representation to a short form.
FCE is equivalent to FeynCalcExternal.
See also: FeynCalcExternal, FCI, FeynCalcInternal.

Examples

```
In[914]:= FCE[{DiracGamma[5], DiracGamma[Momentum[p]]}]
Out[914]= {γ^5, γ · p}

In[915]:= %//StandardForm
Out[915]= {GA[5], GS[p]}

In[916]:= {GA[μ], GAD[ρ], GS[p], SP[p, q], MT[α, β], FV[p, μ]}
Out[916]= {γ^μ, γ^ρ, γ · p, p · q, g^(αβ), p^μ}

In[917]:= %//StandardForm
Out[917]= {GA[μ], GAD[ρ], GS[p], SP[p, q], MT[α, β], FV[p, μ]}

In[918]:= %//FCI
Out[918]= {γ^μ, γ^ρ, γ · p, p · q, g^(αβ), p^μ}

In[919]:= %//StandardForm
Out[919]= {DiracGamma[LorentzIndex[μ]], DiracGamma[LorentzIndex[ρ, D], D],
            DiracGamma[Momentum[p]], Pair[Momentum[p], Momentum[q]],
            Pair[LorentzIndex[α], LorentzIndex[β]], Pair[LorentzIndex[μ], Momentum[p]]}

In[920]:= FCE[%]//StandardForm
Out[920]= {GA[μ], GAD[ρ], GS[p], SP[p, q], MT[α, β], FV[p, μ]}
```

FCI

Description

FCI[exp] translates exp into the internal FeynCalc (datatype-)representation.FCI is equivalent to FeynCalcInternal.
See also: FeynCalcExternal, FeynCalcInternal, FCE.

Examples

```
In[921]:= {GA[μ], GAD[ρ], GS[p], SP[p, q], MT[α, β], FV[p, μ]}
Out[921]= {γ^μ, γ^ρ, γ · p, p · q, g^αβ, p^μ}

In[922]:= %//StandardForm
Out[922]= {GA[μ], GAD[ρ], GS[p], SP[p, q], MT[α, β], FV[p, μ]}

In[923]:= %//FCI
Out[923]= {γ^μ, γ^ρ, γ · p, p · q, g^αβ, p^μ}

In[924]:= %//StandardForm
Out[924]= {DiracGamma[LorentzIndex[μ]], DiracGamma[LorentzIndex[ρ, D], D],

            DiracGamma[Momentum[p]], Pair[Momentum[p], Momentum[q]],

            Pair[LorentzIndex[α], LorentzIndex[β]], Pair[LorentzIndex[μ], Momentum[p]]}

In[925]:= FCE[%]//StandardForm
Out[925]= {GA[μ], GAD[ρ], GS[p], SP[p, q], MT[α, β], FV[p, μ]}
```

FCIntegral

Description

FCIntegral is the head of integrals in a setting of the option IntegralTable of FeynAmpDenominatorSimplify. Currently implemented only for 2-loop integrals.

See also: IntegralTable, FeynAmpDenominatorSimplify.

FCIntegrate

Description

FCIntegrate is an option of certain Feynman integral related functions. It determines which integration function is used to evaluate analytic integrals. Possible settings include Integrate, NIntegrate, (Dot[Integratedx@@#2, #1] &).

See also: FCNIntegrate.

FCNIntegrate

Description

FCNIntegrate is an option of certain Feynman integral related functions which may return output containing both integrals that can be evaluated and integrals that can only be evaluated numerically. It then determines

which integration function is used to evaluate numeric integrals. Possible settings include NIntegrate, (0*#1)&, (Dot[Integratedx@@#2, #1] &).

See also: FCIntegrate.

FC2RHI

Description

FC2RHI[exp, k1, k2] transforms all 2-loop OPE-integrals in FeynAmpDenominator form to the RHI-integrals. FC2RHI[exp] is equivalent to FC2RHI[exp,q1,q2]. The option IncludePair governs the inclusion of scalar products p.k1, p.k2 and k1.k2 (setting True).

```
In[926]:= Options[FC2RHI]
Out[926]= {Dimension → D, IncludePair → True, Do → True}

In[927]:= t = FAD[q1, q1 - q2, q2 - p] SP[q1, OPEDelta]^OPEm//FCI
                                  (Δ · q1)^m
Out[927]= ------------------------------------------------
          q1^2 . (q1 - q2)^2 . (q2 - p)^2

In[928]:= FC2RHI[t, q1, q2]
                m0000
Out[928]= T      10011

In[929]:= %//InputForm
Out[929]= RHI[{0, 0, OPEm, 0, 0}, {0, 1, 1, 0, 1}]
```

See also: RHI.

Examples

```
In[930]:= {}
```

FC2TLI

Description

FC2TLI[exp, k1, k2] transforms all 2-loop OPE-integrals in FeynAmpDenominator form to the TLI-integrals. The option IncludePair governs the inclusion of scalar products p.k1, p.k2 and k1.k2 (setting True).

```
Out[930]= Options[FC2TLI]

In[931]:= {Dimension → D, IncludePair → True, Do → True}

?FAD

In[932]:= FAD[q, q - p, ...] denotes 1/(q^2 (q - p)^2 ...). FAD[{q1, m}, {q1 - p, m},
              q2, ...] is 1/( (q1^2 - m^2) ( (q1 - p)^2 - m^2 ) q2^2 ... ). (Translation
              into FeynCalc internal form is performed by FeynCalcInternal.)
```

```
Out[932]= t = FAD[{q1, m1}, q2 - q1, q2 - p] SP[q1, OPEDelta]^OPEm

In[933]:= 1 / ([(q2 - p)^2]) ([(q2 - q1)^2]) ([q1^2 - m1^2]) (Δ · q1)^m
Out[933]= FC2TLI[t, q1, q2]

In[934]:= T^m0000_10011
Out[934]= %//InputForm

In[935]:= TLI[{OPEm, 0, 0, 0, 0}, {{1, m1}, 0, 0, 1, 1}]
Out[935]= TLI2FC[%]
```

See also: TLI.

Examples

```
In[936]:= (Δ · q1)^m / ((q2 - p)^2 . (q1^2 - m1^2) . (q1 - q2)^2)
```

FDS

Description

FDS is shorthand for FeynAmpDenominatorSimplify.
See also: FeynAmpDenominatorSimplify.

FermionSpinSum

Description

FermionSpinSum[x] constructs the Traces out of squared ampliudes.

```
In[937]:= {}
```

See also: Spinor, ComplexConjugate, DiracTrace, Tr.

Examples

Spinors of fermions of mass *m* are normalized to have square 2 *m* or -2 *m*.

```
In[938]:= Options[FermionSpinSum]
Out[938]= {SpinPolarizationSum → Identity, SpinorCollect → False, ExtraFactor → 1}

In[939]:= SpinorUBar[Momentum[p], m].SpinorU[Momentum[p], m]
Out[939]= ū(p, m).u(p, m)

In[940]:= %//FCI//FermionSpinSum
```

```
Out[940]= tr(m + γ · p)

In[941]:= %/.DiracTrace → Tr
Out[941]= 4 m

In[942]:= SpinorVBar[Momentum[p], m].SpinorV[Momentum[p], m]
Out[942]= v̄(p, m).v(p, m)

In[943]:= %//FCI//FermionSpinSum
Out[943]= tr(γ · p - m)

In[944]:= %/.DiracTrace → Tr
Out[944]= -4 m
```

Notice that SpinorUBar and SpinorU are only input functions. Internally they are converted to Spinor objects.

```
In[945]:= t = Spinor[k1, m].DiracSlash[p].GA[5].Spinor[p1, m]
Out[945]= φ(k1, m).(γ · p).γ^5.φ(p_1, m)

In[946]:= ct = ComplexConjugate[t]
Out[946]= -φ(p_1, m).γ^5.(γ · p).φ(k1, m)

In[947]:= FermionSpinSum[t ct]
Out[947]= -tr((m + γ · k1).(γ · p).γ^5.(m + γ · p_1).γ^5.(γ · p))

In[948]:= % /.DiracTrace → Tr
```

FeynAmp

Description

FeynAmp[q, amp] is the head of a Feynman amplitude. amp denotes the analytical expression for the amplitude and q is the integration variable. FeynAmp[q1, q2, amp] denotes a two-loop amplitude.

FeynAmp has no functional properties and serves just as a head. There are however special typesetting rules attached.

See also: Amplitude.

Examples

This is a 1-loop gluon self-energy amplitude (ignoring factors of (2π)).

```
In[949]:= -4 (p^2 m^2 + k1 · p_1 p^2 - 2 k1 · p p · p_1)
Out[949]= Clear[t, ct]
```

This is a generic 2-loop amplitude.

```
In[950]:= FeynAmp[q, GV[p, μ, a, q - p, α, c, -q, β, e]
              GP[p - q, α, c, ρ, d]GV[-p, ν, b, p - q, ρ, d, q, σ, f] GP[q, β, e, σ, f]]
```

```
Out[950]= ∫ d^D q (Π^{αρ}_{cd}(p - q) Π^{βσ}_{ef}(q) V^{νρσ}(-p, p - q, q) V^{μαβ}(p, q - p, -q) f_{ace} f_{bdf})
```

FeynAmpDenominator

Description

FeynAmpDenominator[PropagatorDenominator[...], PropagatorDenominator[...], ...] is the head of the denominators of the propagators, i.e., FeynAmpDenominator[x] is the representation of 1/x .

See also: FAD, FeynAmpDenominatorSimplify.

Examples

```
In[951]:= FeynAmp[q1, q2, anyexpression]
Out[951]= ∫ d^D q1 ∫ d^D q2 (anyexpression)

In[952]:= FeynAmpDenominator[PropagatorDenominator[p, m]]
Out[952]=  1
          ------
          p^2 - m^2

In[953]:= FeynAmpDenominator[PropagatorDenominator[p, m], PropagatorDenominator[p - q, m]]

In[954]:=  1
          ------------------------
          (p^2 - m^2) . ((p - q)^2 - m^2)
Out[954]= t = FeynAmpDenominator[PropagatorDenominator[p, m]];

In[955]:= StandardForm[t//FCI]
Out[955]= FeynAmpDenominator[PropagatorDenominator[Momentum[p, D], m]]

In[956]:= StandardForm[t//FCE]
```

FeynAmpDenominatorCombine

Description

FeynAmpDenominatorCombine[expr] expands expr with respect to FeynAmpDenominator and combines products of FeynAmpDenominator in expr into one FeynAmpDenominator.

See also: FeynAmpDenominatorSplit.

Examples

```
In[957]:= FAD[{p, m}]
Out[957]= Clear[t];

In[958]:= t = FAD[q] FAD[q - p]
```

```
Out[958]= 1/[q^2] 1/[(q - p)^2]

In[959]:= FeynAmpDenominatorCombine[%]//FCE//StandardForm
Out[959]= FAD[q, -p + q]
```

FeynAmpDenominatorSimplify

Description

FeynAmpDenominatorSimplify[exp] tries to simplify each PropagatorDenominator in a canonical way. FeynAmpDenominatorSimplify[exp, q1] simplifies all FeynAmpDenominator's in exp in a canonical way, including some translation of momenta. FeynAmpDenominatorSimplify[exp, q1, q2] additionally removes integrals with no mass scale.

FDS can be used as an alias.

```
In[960]:= FeynAmpDenominatorSplit[%]//FCE//StandardForm
Out[960]= FAD[q] FAD[-p + q]
```

See also: OneLoopSimplify.

Examples

The cornerstone of dimensional regularization is that FDS

```
In[961]:= FeynAmpDenominatorSimplify
Out[961]= ∫ d^n k f(k)/k^2m = 0 .
```

This brings FeynAmpDenominatorSimplify[f[k] FAD[k, k], k] into a standard form.

```
In[962]:= 0
Out[962]= 1/((k - p_1)^2 (k - p_2)^2)

In[963]:= FeynAmpDenominatorSimplify[FAD[k - p_1, k - p_2], k]
Out[963]= 1/(k^2 . (k - p_1 + p_2)^2)

In[964]:= t = FeynAmpDenominatorSimplify[FAD[k - p_1, k - p_2] SPD[k, k], k]
Out[964]= k^2/(k^2 . (k - p_1 + p_2)^2) + 2 k . p_2/(k^2 . (k - p_1 + p_2)^2) + p_2^2/(k^2 . (k - p_1 + p_2)^2)

In[965]:= r = SPC[t, k, FDS → True]
Out[965]= p_2^2/(k^2 . (k - p_1 + p_2)^2) - 2 k . p_2/(k^2 . (k + p_1 - p_2)^2)

In[966]:= OneLoopSimplify[r, k]
Out[966]= p_1 . p_2/(k^2 . (k - p_1 + p_2)^2)

In[967]:= FDS[FAD[k - p1, k - p2]SPD[k, OPEDelta]^2, k]
```

FeynAmpDenominatorSplit

Description

FeynAmpDenominatorSplit[expr] splits all FeynAmpDenominator[a,b, ...] in expr into FeynAmpDenominator[a]*FeynAmpDenominator[b] ... . FeynAmpDenominatorSplit[expr, q1] splits all FeynAmpDenominator in expr into a product of two, one containing q1 and other momenta, the second without q1.

See also: FeynAmpDenominatorCombine.

Examples

```
In[968]:= k·Δ^2/(k^2.(k - p_1 + p_2)^2) + 2 k·Δ Δ·p_2/(k^2.(k - p_1 + p_2)^2) + Δ·p_2^2/(k^2.(k - p_1 + p_2)^2)
Out[968]= Clear[t, r]

In[969]:= t = FAD[q1, q1 - p, q1 - q2, q2, q2 - p]//FCI
                                    1
Out[969]= ------------------------------------------------
          q1^2.(q1 - p)^2.(q1 - q2)^2.q2^2.(q2 - p)^2

In[970]:= t//Head
Out[970]= FeynAmpDenominator

In[971]:= FeynAmpDenominatorSplit[t]
                                    1
Out[971]= ------------------------------------------------
          q1^2 (q1 - p)^2 (q1 - q2)^2 q2^2 (q2 - p)^2

In[972]:= %//FCE//StandardForm
Out[972]= FAD[q1] FAD[-p + q1] FAD[q1 - q2] FAD[q2] FAD[-p + q2]

In[973]:= FeynAmpDenominatorSplit[t, q1]//FCE//StandardForm
Out[973]= FAD[q2, -p + q2] FAD[q1, -p + q1, q1 - q2]

In[974]:= FeynAmpDenominatorCombine[%]//FCE//StandardForm
```

FeynAmpList

Description

FeynAmpList[info][FeynAmp[...], FeynAmp[...], ...] is a head of a list of Feynman amplitudes.

FeynAmpList has no functional properties and serves just as a head.

See also: FeynAmp.

FeynCalc

Description

FeynCalc is simply a symbol with a usage definition.

```
In[975]:= FAD[q1, q2, q1 - q2, -p + q1, -p + q2]
Clear[t]
```

FeynCalcExternal

Description

FeynCalcExternal[exp] translates exp from the internal FeynCalc representation to a shorthand form. See also: FeynCalcInternal.

Examples

```
In[976]:= ?FeynCalc
Out[976]= For installation notes visit www.feyncalc.org

          For a list of availabe objects type $FeynCalcStuff,

             which contains a list of all functions and options in StringForm.

             You can get on - line information by ?function, e.g., ?Contract.

          There are several useful functions for short input, type

             $FCS for a list of short commands. Then type, e.g., ?GA.


          To get rid of the start - up messages put the line

          $FeynCalcStartupMessages = False;

           into your init.m or the HighEnergyPhysics/FeynCalcConfig.m file.

In[977]:= FeynCalcExternal[DiracGamma[5]]
Out[977]= γ^5

In[978]:= %//StandardForm
Out[978]= GA[5]

In[979]:= {GA[μ], GAD[ρ], GS[p], SP[p, q], MT[α, β], FV[p, μ]}
Out[979]= {γ^μ, γ^ρ, γ · p, p · q, g^αβ, p^μ}

In[980]:= %//StandardForm
Out[980]= {GA[μ], GAD[ρ], GS[p], SP[p, q], MT[α, β], FV[p, μ]}
```

```
In[981]:= %//FeynCalcInternal
Out[981]= {γ^μ, γ^ρ, γ · p, p · q, g^αβ, p^μ}

In[982]:= %//StandardForm
Out[982]= {DiracGamma[LorentzIndex[μ]], DiracGamma[LorentzIndex[ρ, D], D],

            DiracGamma[Momentum[p]], Pair[Momentum[p], Momentum[q]],

            Pair[LorentzIndex[α], LorentzIndex[β]], Pair[LorentzIndex[μ], Momentum[p]]}
```

FeynCalcForm

Description

FeynCalcForm[expr] changes the printed output to a an easy-to-read form. It allows a readable output also when running a terminal based *Mathematica* session. Whether the result of FeynCalcForm[expr] is displayed or not, depends on the setting of $PrePrint. $PrePrint = FeynCalcForm forces displaying everything after applying FeynCalcForm. In order to change to the normal (internal) Mathematica OutputForm, do: ($PrePrint=.).

See also: FC, FeynCalcExternal, FeynCalcInternal.

Examples

This is the normal notebook display:

```
In[983]:= FeynCalcExternal[%]//StandardForm
Out[983]= {GA[μ], GAD[ρ], GS[p], SP[p, q], MT[α, β], FV[p, μ]}
```

This is the shorthand (terminal) display (easy-to-read form):

```
In[984]:= SUNTrace[SUNT[a].SUNT[b].SUNT[c]]

In[985]:= tr(T_a.T_b.T_c)

In[986]:= $PrePrint = FeynCalcForm;
Out[986]= tr[T[a] T[b] T[c]]
```

Reset to normal notebook display:

```
In[987]:= SetOptions[$FrontEnd, Evaluate[(Options[$FrontEnd, "CommonDefaultFormatTypes"]/.

                  ("Output" → _) → ("Output" → OutputForm))[[1]]]];

In[988]:= SUNTrace[SUNT[a].SUNT[b].SUNT[c]]
```

FeynCalcInternal

Description

FeynCalcInternal[exp] translates exp into the internal FeynCalc (abstract data-type) representation.
See also: FeynCalcExternal, FCI, FCE.

Examples

```
In[989]:= $PrePrint = .;
Out[989]= SetOptions[$FrontEnd, Evaluate[(Options[$FrontEnd, "CommonDefaultFormatTypes"] /.
                ("Output" → _) → ("Output" → TraditionalForm))[[1]]]];

In[990]:= {GA[μ], GAD[ρ], GS[p], SP[p, q], MT[α, β], FV[p, μ]}
Out[990]= {γ^μ, γ^ρ, γ · p, p · q, g^αβ, p^μ}

In[991]:= %//StandardForm
Out[991]= {GA[μ], GAD[ρ], GS[p], SP[p, q], MT[α, β], FV[p, μ]}

In[992]:= %//FeynCalcInternal
Out[992]= {γ^μ, γ^ρ, γ · p, p · q, g^αβ, p^μ}

In[993]:= %//StandardForm
Out[993]= {DiracGamma[LorentzIndex[μ]], DiracGamma[LorentzIndex[ρ, D], D],
           DiracGamma[Momentum[p]], Pair[Momentum[p], Momentum[q]],
           Pair[LorentzIndex[α], LorentzIndex[β]], Pair[LorentzIndex[μ], Momentum[p]]}

In[994]:= FeynCalcExternal[%]//StandardForm
Out[994]= {GA[μ], GAD[ρ], GS[p], SP[p, q], MT[α, β], FV[p, μ]}

In[995]:= FCI[{SD[a, b], SUND[a, b, c], SUNF[a, b, c], FAD[q], LC[μ, ν, ρ, σ]}]
Out[995]= {δ_ab, d_abc, f_abc, 1/q^2, ε^μνρσ}
```

FeynCalc2FORM

Description

FeynCalc2FORM[expr] displays expr in FORM syntax. FeynCalc2FORM[file, x] writes x in FORM syntax to a file. FeynCalc2FORM[file, x==y] writes x=y to a file in FORM syntax.
See also: FORM2FeynCalc.

```
In[996]:= %//StandardForm
```

```
Out[996]= {SUNDelta[SUNIndex[a], SUNIndex[b]], SUND[SUNIndex[a], SUNIndex[b], SUNIndex[c]],
            SUNF[SUNIndex[a], SUNIndex[b], SUNIndex[c]],
            FeynAmpDenominator[PropagatorDenominator[Momentum[q, D], 0]],
            Eps[LorentzIndex[μ], LorentzIndex[ν], LorentzIndex[ρ], LorentzIndex[σ]]}
```

Examples

```
In[997]:= Options[FeynCalc2FORM]
Out[997]= {EpsDiscard → False, FORMEpilog →, FORMProlog → write statistics;,
            Replace → {\[Alpha] → al, \[Beta] → be, \[Gamma] → ga, \[Delta] → de,
                \[Mu] → μ, \[Nu] → ν, \[Rho] → ro, \[Sigma] → si}, TraceDimension → 4}

In[998]:= MT[μ, ν]FV[p, ρ] y^2/d
```

$$\frac{y^2\, p^{\rho}\, g^{\mu\nu}}{d}$$

```
In[999]:= FeynCalc2FORM[%];
Out[999]= (y^2 * d_(mu, nu) * p(ro))/d

In[1000]:= LC[α, β, δ, ρ]
```

$$\epsilon^{\alpha\beta\delta\rho}$$

```
In[1001]:= FeynCalc2FORM[%];
Out[1001]= (-i_) * e_(al, be, de, ro)

In[1002]:= DiracTrace[GA[μ, ν, ρ, σ]]
```

$$\mathbf{tr}(\gamma^{\mu}.\gamma^{\nu}.\gamma^{\rho}.\gamma^{\sigma})$$

```
In[1003]:= FeynCalc2FORM[%];
Out[1003]= g_(0, mu) * g_(0, nu) * g_(0, ro) * g_(0, si)

In[1004]:= DiracTrace[GA[μ, ν]]DiracTrace[GA[μ, ρ]]
```

$$\mathbf{tr}(\gamma^{\mu}.\gamma^{\nu})\,\mathbf{tr}(\gamma^{\mu}.\gamma^{\rho})$$

```
In[1005]:= FeynCalc2FORM[%];

In[1006]:= g_(0, mu) * g_(0, nu) * g_(1, mu) * g_(1, ro)
Out[1006]= FeynCalc2FORM["fc2ftest.f", MT[μ, ν]FV[p, μ]];

In[1007]:= ReadList[If[$OperatingSystem === "MacOS", " : ", ""] <> "fc2ftest.f", String]
Out[1007]= {d_(mu, nu) * p(mu)}

In[1008]:= t = Tr[GA[μ, ν, ρ, σ].GS[p, q]]

In[1009]:= 4 (q^μ p^ν g^ρσ - p^μ q^ν g^ρσ + g^μν p · q g^ρσ - q^μ g^νσ p^ρ + g^μσ q^ν p^ρ + p^μ g^νσ q^ρ - g^μσ p^ν q^ρ + q^μ g^νρ p^σ -
              g^μρ q^ν p^σ + g^μν q^ρ p^σ - p^μ g^νρ q^σ + g^μρ p^ν q^σ - g^μν p^ρ q^σ + g^μσ g^νρ p · q - g^μρ g^νσ p · q)
Out[1009]= FeynCalc2FORM["fc2ftest.f", L == t];
```

```
In[1010]:= TableForm[
             ReadList[If[$OperatingSystem === "MacOS", " : ", ""] <> "fc2ftest.f", String]]

           Indices \[Mu], \[Nu], \[Rho], \[Sigma];
           Vectors p, q;


           write statistics;


           Local L = (
In[1011]:= 4 * (d_(mu, si) * d_(nu, ro) * q.p - d_(mu, ro) * d_(nu, si) * q.p + d_(mu, nu) * d_(ro, si) * q.p+
           d_(ro, si) * p(nu) * q(mu) - d_(nu, si) * p(ro) * q(mu) + d_(nu, ro) * p(si) * q(mu)-
           d_(ro, si) * p(mu) * q(nu) + d_(mu, si) * p(ro) * q(nu) - d_(mu, ro) * p(si) * q(nu)+
           d_(nu, si) * p(mu) * q(ro) - d_(mu, si) * p(nu) * q(ro) + d_(mu, nu) * p(si) * q(ro)-
           d_(nu, ro) * p(mu) * q(si) + d_(mu, ro) * p(nu) * q(si) - d_(mu, nu) * p(ro) * q(si)) );


           Print;
           .end
In[1012]:= If[FileNames["fc2ftest.f"] =!= {}, DeleteFile["fc2ftest.f"]];
Clear[t];
```

FeynCalcToLaTeX

Description

FeynCalcToLaTeX[expr] generates LaTeX with line-breaking for expr.
FeynCalcToLaTeX[expr, 500] generates LaTeX for expr with 500 being the Window width setting for the Mathematica frontend. Increasing its value will generate less line breaks.

```
In[1013]:= ?FeynCalcToLaTeX
Out[1013]= FeynCalcToLaTeX[expr] generates LaTeX with line-breaking
               for expr. FeynCalcToLaTeX[expr, 500] generates LaTeX for expr
               with 500 being the Window width setting for the Mathematica
               frontend. Increasing its value will generate less line breaks.
In[1014]:= GluonPropagator[p, 1, 2]//Explicit
Out[1014]= -(I g^(li1li2) δ_(ci1ci2)) / p^2
```

FeynmanParameterNames

Description

FeynmanParameterNames is an option for FeynmanParametrize and FeynmanParametrize.
See also: FeynmanParametrize, FeynmanParametrize.

FeynmanParametrize ***unfinished***

Description

FeynmanParametrize[exp,k] introduces feynman parameters for all one-loop integrals in exp (k = integration momentum).

```
In[1015]:= FeynCalcToLaTeX[%]
Out[1015]= -\frac{\ImaginaryI \multsp {g^{\Mvariable{li1}\
              Mvariable{li2}}}\multsp {{\delta }_{\Mvariable{ci1}\Mvariable{ci2}}}}{{p^2}}
```

Examples

```
In[1016]:= Options[FeynmanParametrize]
```

FeynRule

Description

FeynRule[lag, {fields}] derives the Feynman rule corresponding to the field configuration fields of the lagrangian lag.

```
In[1017]:= {FeynmanParameterNames → {x, y, z}}
Out[1017]= {}
```

FeynRule does not calculate propagator Feynman rules.
The option ZeroMomentumInsertion can be used for twist-2 and higher twist operators.
See also: Lagrangian.

Examples

```
Out[1017]= Options[FeynRule]

In[1018]:= {Anti5 → -∞, Contract → False, Factor1 → False, FinalSubstitutions → {}, PartialD → RightPartial
             Schouten → False, ZeroMomentumInsertion → True, InitialFunction → PhiToFC}
Out[1018]= gou = Lagrangian["ogu"]
```

In[1019]:= $\frac{1}{2}\, i^{m-1}\, F^{a}_{\alpha\Delta} \cdot \left(D^{ab}_{\Delta}\right)^{m-2} \cdot F^{b}_{\alpha\Delta}$

Out[1019]= gop = Lagrangian["ogp"]

In[1020]:= $\frac{1}{2}\, i^{m}\, \epsilon^{\alpha\beta\gamma\Delta} \cdot F^{a}_{\beta\gamma} \cdot \left(D^{ab}_{\Delta}\right)^{m-2} \cdot F^{b}_{\alpha\Delta}$

Out[1020]= Explicit[gop]

2-gluon Feynman rules (unpolarized)

In[1021]:= $\frac{1}{2}\, i^{m}\, \epsilon^{\alpha\beta\gamma\Delta} \cdot \left(\partial_{\beta} A^{a}_{\gamma} - \partial_{\gamma} A^{a}_{\beta} + g_s\, A^{b2}_{\beta} \cdot A^{c12}_{\gamma}\, f_{ab2c12}\right) \cdot \left(D^{ab}_{\Delta}\right)^{m-2} \cdot \left(\text{QuantumField}(\partial_{\alpha}, \mathbf{A}, \Delta, b) - \partial_{\Delta} A^{b}_{\alpha} + g_s\, A^{b3}_{\alpha} \cdot A^{c13}_{\Delta}\, f_{bb3c13}\right)$

Out[1021]= Cases2[%, QuantumField]

In[1022]:= $\left\{A^{b3}_{\alpha}, A^{b2}_{\beta}, A^{c12}_{\gamma}, A^{c13}_{\Delta}, \text{QuantumField}(\partial_{\alpha}, \mathbf{A}, \Delta, b), \partial_{\beta} A^{a}_{\gamma}, \partial_{\gamma} A^{a}_{\beta}, \partial_{\Delta} A^{b}_{\alpha}\right\}$

Out[1022]= fi = {QuantumField[GaugeField, {μ}, {a}][p], QuantumField[GaugeField, {ν}, {b}][q]}

2-gluon Feynman rules (polarized)

In[1023]:= $\left\{A^{a}_{\mu}, A^{b}_{\nu}\right\}$

Out[1023]= f2u = FullSimplify/@Factor2[FeynRule[gou, fi, ZeroMomentumInsertion → False]]

In[1024]:= $\frac{(\Delta \cdot q^2\, (\Delta \cdot p)^m + (\Delta \cdot q)^m\, \Delta \cdot p^2)\, (q^{\mu}\, \Delta^{\nu}\, \Delta \cdot p - g^{\mu\nu}\, \Delta \cdot q\, \Delta \cdot p + \Delta^{\mu}\, (p^{\nu}\, \Delta \cdot q - \Delta^{\nu}\, p \cdot q))\, \delta_{ab}}{2\, \Delta \cdot p^2\, \Delta \cdot q^2}$

Out[1024]= fi = {QuantumField[GaugeField, {μ}, {a}][p], QuantumField[GaugeField, {ν}, {b}][q]}

In[1025]:= $\left\{A^{a}_{\mu}, A^{b}_{\nu}\right\}$

Out[1025]= f2p = FullSimplify/@Factor2[FeynRule[gop, fi, ZeroMomentumInsertion → False]]

Compare with the Feynman rule tabulated in Twist2GluonOperator.

In[1026]:= $-\frac{i\, ((\Delta \cdot p)^m\, (\epsilon^{\nu pq\Delta}\, \Delta^{\mu} + \epsilon^{\mu\nu q\Delta}\, \Delta \cdot p)\, \Delta \cdot q^2 - \Delta \cdot p^2\, (\Delta \cdot q)^m\, (\epsilon^{\mu pq\Delta}\, \Delta^{\nu} + \epsilon^{\mu\nu p\Delta}\, \Delta \cdot q))\, \delta_{ab}}{\Delta \cdot p^2\, \Delta \cdot q^2}$

Out[1026]= Factor2[Calc[f2p/.p → -q]]

quark-quark Feynman rule (unpolarized)

In[1027]:= $i\, (1 - (-1)^m)\, \epsilon^{\mu\nu\Delta q}\, (\Delta \cdot q)^{m-1}\, \delta_{ab}$

Out[1027]= Twist2GluonOperator[q, {μ, a}, {ν, b}, Polarization → 1, Explicit → True]

quark-quark -gluon-gluon Feynman rule (unpolarized)

In[1028]:= $i\, \epsilon^{\mu\nu\Delta q}\, (1 - (-1)^m)\, (\Delta \cdot q)^{m-1}\, \delta_{ab}$

Out[1028]= qo = Lagrangian["oqu"]

In[1029]:= $i^{m}\, \overline{\psi} \cdot (\gamma \cdot \Delta) \cdot D_{\Delta}^{m-1} \cdot \psi$

Out[1029]= qo = Lagrangian["oqu"]

In[1030]:= $i^{m}\, \overline{\psi} \cdot (\gamma \cdot \Delta) \cdot D_{\Delta}^{m-1} \cdot \psi$

Out[1030]= qggf = {QuantumField[QuarkField][p], QuantumField[AntiQuarkField][q], QuantumField[GaugeField, {μ}, {a}][r], QuantumField[GaugeField, {ν}, {b}][s]}

In[1031]:= $\left\{\psi, \overline{\psi}, A^{a}_{\mu}, A^{b}_{\nu}\right\}$

Out[1031]= n4 = FeynRule[qo, qggf, ZeroMomentumInsertion → True]

```
In[1032]:= g_s^2 T_b.T_a.(γ·Δ) (Σ_{j=0}^{m-3} (j+1) (-1)^j (Δ·p)^{-j+m-3} (Δ·q)^i (Δ·q+Δ·s)^{j-i}) Δ^μ Δ^ν -
           (-1)^m g_s^2 T_a.T_b.(γ·Δ) (Σ_{j=0}^{m-3} (j+1) (-1)^j (Δ·p)^i (Δ·q)^{-j+m-3} (Δ·p+Δ·s)^{j-i}) Δ^μ Δ^ν
Out[1032]= t4 = Twist2QuarkOperator[{p}, {q}, {r, μ, a}, {s, ν, b}, Polarization → 0]
```

In general equality can be shown by Timing[Factor2[FCE[Calc[ChangeDimension[FCE[OPESumExplicit[n4-t4]],4]/.s→-p-q-r]]]] but it is a little bit slow ...

```
In[1033]:= -(-1)^m g_s^2 (γ·Δ).(T_a.T_b (Σ_{i=0}^{m-3} (i+1) (-(Δ·p))^{-i+m-3} (Δ·q)^j (Δ·q+Δ·r)^{i-j}) +
           T_b.T_a (Σ_{i=0}^{m-3} (i+1) (-(Δ·p))^{-i+m-3} (Δ·q)^j (Δ·q+Δ·s)^{i-j})) Δ^μ Δ^ν
```

QCD vertices

```
In[1034]:= Calc[n4 - t4/.OPEm → 5/.s → -p - q - r/.D → 4]
Out[1034]= 0

In[1035]:= Clear[qggf, n2, n4]
Out[1035]= fii = {QuantumField[GaugeField, {μ}, {a}][p],
             QuantumField[GaugeField, {ν}, {b}][q], QuantumField[GaugeField, {ρ}, {c}][r]}

In[1036]:= {A_μ^a, A_ν^b, A_ρ^c}
Out[1036]= g3 = FeynRule[Lagrangian["QCD"], fii]

In[1037]:= g_s ((q^μ - r^μ) g^{νρ} - g^{μρ} (p^ν - r^ν) + g^{μν} (p^ρ - q^ρ)) f_abc
Out[1037]= GluonVertex[{p, μ, a}, {q, ν, b}, {r, ρ, c}, Explicit → True]

In[1038]:= g_s ((q - r)^μ g^{νρ} + g^{μρ} (r - p)^ν + g^{μν} (p - q)^ρ) f_abc
Out[1038]= Calc[g3 - ChangeDimension[%, 4]]

In[1039]:= 0
Out[1039]= fi4 = {QuantumField[GaugeField, {μ}, {a}][p], QuantumField[GaugeField, {ν}, {b}][q],
             QuantumField[GaugeField, {ρ}, {c}][r], QuantumField[GaugeField, {σ}, {d}][s]}

In[1040]:= {A_μ^a, A_ν^b, A_ρ^c, A_σ^d}
Out[1040]= g4 = FeynRule[Lagrangian["QCD"], fi4]

In[1041]:= i (g^{μρ} g^{νσ} - g^{μν} g^{ρσ}) f_adsi1 f_bcsi1 g_s^2 +
           i (g^{μσ} g^{νρ} - g^{μν} g^{ρσ}) f_acsi1 f_bdsi1 g_s^2 + i (g^{μσ} g^{νρ} - g^{μρ} g^{νσ}) f_absi1 f_cdsi1 g_s^2
Out[1041]= GluonVertex[{p, μ, a}, {q, ν, b}, {r, ρ, c}, {s, σ, d}, Explicit → True]

In[1042]:= -i g_s^2
           ((g^{μν} g^{ρσ} - g^{μρ} g^{νσ}) f_adu8 f_bcu8 + (g^{μν} g^{ρσ} - g^{μσ} g^{νρ}) f_acu8 f_bdu8 + (g^{μρ} g^{νσ} - g^{μσ} g^{νρ}) f_abu8 f_cdu8)
```

FI

Description

FI changes the output format to InputForm. This is useful to see the internal representation of FeynCalc objects. To change back to FeynCalcForm use FC.

See also: FeynCalcForm, FC, FeynCalcExternal, FeynCalcInternal.

FieldDerivative

Description

FieldDerivative[f[x],x,li1,li2,...] is the derivative of f[x] with respect to space-time variables x and with Lorentz indices li1, li2, ..., where li1, li2, ... have head LorentzIndex. FieldDerivative[f[x],x,li1,li2,...] can be given as FieldDerivative[f[x],x,{l1,l2,...}], where l1 is li1 without the head, ... FieldDerivative is defined only for objects with head QuantumField[...]. If the space-time derivative of other objects is wanted, the corresponding rule must be specified.

See also: PartialD, ExpandPartialD.

Examples

```
In[1043]:= Calc[g4 - ChangeDimension[%, 4]]
Out[1043]= 0

In[1044]:= Clear[f2p, f2u, f3, f32, fi, fi4, fii, g3, g4, gop, gou,
             n3, nf3, n4, np2, npf3, p33, pf3, pn3, pqo, qf, qp, qgf, qo, t4]
Out[1044]= QuantumField[A, {μ}][x].QuantumField[B, {ν}][y].
             QuantumField[C, {ρ}][x].QuantumField[D, {σ}][y]

In[1045]:= A_μ.B_ν.C_ρ.D_σ
Out[1045]= FieldDerivative[%, x, {μ}]//DotExpand
```

FieldStrength

Description

FieldStrength[μ, ν, a] is the field strength tensor $A_\mu.B_\nu.\partial_\mu C_\rho.D_\sigma+\partial_\mu A_\mu.B_\nu.C_\rho.D_\sigma$FieldDerivative[%, y, {$\nu$}]//DotExpand. FieldStrength[μ, ν] is the field strength tensor $A_\mu.B_\nu.\partial_\mu C_\rho.\partial_\nu D_\sigma + A_\mu.\partial_\nu B_\nu.\partial_\mu C_\rho.D_\sigma + \partial_\mu A_\mu.B_\nu.C_\rho.\partial_\nu D_\sigma + \partial_\mu A_\mu.\partial_\nu B_\nu.C_\rho.D_\sigma$ The name of the field (A) and the coupling constant (g) can be set through the options or by additional arguments. The first two indices are interpreted as type LorentzIndex, except OPEDelta, which is converted to Momentum[OPEDelta].

```
In[1046]:= ∂_μ A^a_ν - ∂_ν A^a_μ + g_s A^b1_μ A^c1_ν
Out[1046]= f^a b1 c1
```

See also: QuantumField.

Examples

```
In[1047]:= ∂_μ A_ν - ∂_ν A_μ .
Out[1047]= Options[FieldStrength]

In[1048]:= {CouplingConstant → g_s, Explicit → False,
             HighEnergyPhysics`fctools`IndexPosition`IndexPosition → {0, 0}, Symbol → F, QuantumField → A
Out[1048]= FieldStrength[μ, ν]

In[1049]:= F_μν
Out[1049]= FieldStrength[μ, ν, a]

In[1050]:= F^a_μν
Out[1050]= FieldStrength[μ, ν, Explicit → True]

In[1051]:= ∂_μ A_ν - ∂_ν A_μ
Out[1051]= FieldStrength[μ, ν, a, Explicit → True]

In[1052]:= ∂_μ A^a_ν - ∂_ν A^a_μ + g_s A^b12_μ .A^c50_ν f_ab12c50
Out[1052]= StandardForm[FieldStrength[μ, ν, Explicit → True]]

In[1053]:= QuantumField[PartialD[LorentzIndex[μ]], GaugeField, LorentzIndex[ν]]-
             QuantumField[PartialD[LorentzIndex[ν]], GaugeField, LorentzIndex[μ]]
Out[1053]= StandardForm[FieldStrength[μ, OPEDelta, Explicit → True]]
```

FinalFunction

Description

FinalFunction is an option for OneLoopSum.
See also: OneLoopSum.

FinalSubstitutions

Description

FinalSubstitutions is an option for OneLoop and OneLoopSum and Write2. All substitutions indicated hereby are done at the end of the calculation.
See also: OneLoop, OneLoopSum, Write2.

FORM

Description

FORM is an option for RHI. If set to True a FORM file is generated and run from Mathematica (provided R. Hamberg's FORM-program is installed correctly ...).
See also: RHI, FeynCalc2FORM.

FORMEpilog

Description

FORMEpilog is an option for FeynCalc2FORM. It may be set to a string which is put at the end of the FORM-file.
See also: FeynCalc2FORM, FORMProlog.

FORMProlog

Description

FORMProlog is an option for FeynCalc2FORM. It may be set to a string which is put after the type declarations of the FORM-file.
See also: FeynCalc2FORM, FORMEpilog.

FORM2FeynCalc

Description

FORM2FeynCalc[expr] translates the FORM expr into FeynCalc notation. FORM2FeynCalc[file] translates the FORM expresssions in file into FeynCalcnotation. FORM2FeynCalc[file, x1, x2, ...] reads in a file in FORM-format and translates the assignments for the variables a, b, ... into FeynCalc syntax. If the option Set is True, the variables x1, x2 are assigned to the right hand sides defined in the FORM-file.
See also: FeynCalc2FORM.

```
In[1054]:= QuantumField[PartialD[LorentzIndex[μ]], GaugeField, Momentum[OPEDelta]]-
             QuantumField[PartialD[Momentum[OPEDelta]], GaugeField, LorentzIndex[μ]]
Out[1054]= FieldStrength[μ, ν, a, CouplingConstant → -Gstrong, Explicit → True]
```

Examples

```
In[1055]:= ∂_μ A^a_ν - ∂_ν A^a_μ - g_s A^b13_μ . A^c51_ν f_ab13c51
Out[1055]= Options[FORM2FeynCalc]
```

```
In[1056]:= {Dimension → 4, FinalSubstitutions → {}, Dot → Dot, HoldForm → True,
             LorentzIndex → {μ, ν, al, be}, Set → False, Replace → {}, Vectors → Automatic}
Out[1056]= FORM2FeynCalc["p.q + 2 * x m^2"]
```

Functions are automatically converted right, but bracketed expressions need to be substituted explicitly.

```
In[1057]:= 2 x.m^2 + p · q
Out[1057]= %//StandardForm

In[1058]:= 2 x.m^2 + SP[p, q]
Out[1058]= FORM2FeynCalc["x + f(z) + log(x)^2 + [li2(1 - x)]",
             Replace → {"[li2(1 - x)]" → "PolyLog[2, 1 - x]"}]

In[1059]:= Log^2(x) + x + f(z) + Li_2(1 - x)
Out[1059]= %//StandardForm

In[1060]:= x + f[z] + Log[x]^2 + PolyLog[2, 1 - x]
Out[1060]= FORM2FeynCalc["x + [(1)] * y - [(-1)^m]"]

In[1061]:= x + Hold[1].y - Hold[(-1)^m]
Out[1061]= ReleaseHold[%]

In[1062]:= x - (-1)^m + 1.y
Out[1062]= FORM2FeynCalc["p(mu) * q(nu) + d_(mu, nu)"]

In[1063]:= p^μ.q^ν + g^μν
Out[1063]= %//StandardForm

In[1064]:= FV[p, μ].FV[q, ν] + MT[μ, ν]
Out[1064]= FORM2FeynCalc["p(mu) * q(nu) + d_(mu, nu)", Replace → {μ → μ, ν → ν}]
```

FourDivergence

Description

FourDivergence[exp, FourVector[p, mu]] calculates the partial derivative of exp w.r.t. p(mu). FourDivergence[exp, FourVector[p, mu], FourVector[p,nu], ...]gives the multiple derivative.

See also: RussianTrick.

Examples

```
In[1065]:= p^μ.q^ν + g^μν
Out[1065]= FORM2FeynCalc["i_*az*bz*aM^2*D1*[(1)]*b_G1*(4*eperp(mu, nu)*avec.bvec*blam)"]

In[1066]:= (4 i).az.bz.aM^2.D1.Hold[1].b$G1.eperp(μ, ν).(avec · bvec).blam
Out[1066]= t = ScalarProduct[p, q]
```

```
In[1067]:= p · q
Out[1067]= FourDivergence[t, FourVector[q, μ]]

In[1068]:= p^μ
Out[1068]= t = ScalarProduct[p - k, q]

In[1069]:= (p - k) · q
```

FourLaplacian

Description

FourLaplacian[exp, p, q] is FourDivergence[t, FourVector[$k - p, \mu$]]0exp.

```
In[1070]:= Clear[t]
Out[1070]= ∂/∂p_μ
```

See also: FourDivergence, RussianTrick.

Examples

```
In[1071]:= ∂/∂q_μ
Out[1071]= Options[FourLaplacian]

In[1072]:= {Dimension → D}
Out[1072]= SP[q, q]

In[1073]:= q^2
Out[1073]= FourLaplacian[%, q, q]

In[1074]:= 2 D
Out[1074]= SOD[q]^OPEmFAD[q, q - p]//FCI
```

FourVector

Description

FourVector[p, $\frac{(\Delta\cdot q)^m}{q^2.(q-p)^2}$] is the four-dimensional vector p with Lorentz index FourLaplacian[$\%, q, q$]. A vector with space-time Dimension D is obtained by supplying the option Dimension → D.

See also: FV, FVD, Pair.

Examples

```
In[1075]:= 4 m Δ · p (Δ · q)^(m-1) / (q^2 . (q - p)^2 . (q - p)^2) - 2 D (Δ · q)^m / (q^2 . q^2 . (q - p)^2) - 4 m (Δ · q)^m / (q^2 . q^2 . (q - p)^2) + 12 (Δ · q)^m / (q^2 . q^2 . (q - p)^2) -
           2 D (Δ · q)^m / (q^2 . (q - p)^2 . (q - p)^2) - 4 m (Δ · q)^m / (q^2 . (q - p)^2 . (q - p)^2) + 12 (Δ · q)^m / (q^2 . (q - p)^2 . (q - p)^2) - 4 (Δ · q)^m p^2 / (q^2 . q^2 . (q - p)^2 . (q - p)^2)
Out[1075]= μ

In[1076]:= μ
Out[1076]= FourVector[p, μ]

In[1077]:= p_μ
Out[1077]= FourVector[p - q, μ]

In[1078]:= (p - q)_μ
Out[1078]= StandardForm[FourVector[p, μ]]

In[1079]:= FourVector[p, μ]
Out[1079]= StandardForm[FourVector[p, μ, Dimension → D]]
```

There is no special function to expand momenta in FourVector. Since FourVector is turned into Pair internally ExpandScalarProduct may be used.

```
In[1080]:= FourVector[p, μ, Dimension → D]
Out[1080]= StandardForm[FCE[FourVector[p, μ]]]
```

FreeIndex

Description

FreeIndex is a datatype which is recognized by Contract. Possible use: DataType[mu, FreeIndex] = True.

See also: Contract, DataType.

FreeQ2

Description

FreeQ2[expr, {form1, form2, ...}] yields True if expr does not contain any occurence of form1, form2, ... and False otherwise. FreeQ2[expr, form] is the same as FreeQ[expr, form].

See also: SelectFree, SelectNotFree.

Examples

```
In[1081]:= FourVector[p, μ]
Out[1081]= ExpandScalarProduct[FourVector[p - q, μ]]

In[1082]:= p^μ - q^μ
Out[1082]= FreeQ2[x + f[x] + y, {a, x}]

In[1083]:= False
Out[1083]= FreeQ2[x + f[x] + y, {a, b}]

In[1084]:= True
Out[1084]= FreeQ2[x, y]
```

FRH

Description

FRH[exp_] := FixedPoint[ReleaseHold, exp], i.e., FRH removes all HoldForm and Hold in exp.
See also: Isolate.

Examples

```
In[1085]:= True
Out[1085]= FreeQ2[f[x], f]

In[1086]:= False
Out[1086]= Hold[1 - 1 - Hold[2 - 2]]

In[1087]:= Hold[-Hold[2 - 2] + 1 - 1]
Out[1087]= FRH[%]

In[1088]:= 0
Out[1088]= Isolate[Solve[x^3 - x - 1 == 0], x, IsolateNames → HH]
```

FromTFi

Description

FromTFi[expr, q1, q2, p] translates the TFi notatation from the TARCER package to the usual FeynCalc notation. See TFi for details on the conventions.
See also: TFi, ToTFi.

Examples

```
In[1089]:= {{x → HH(3)}, {x → HH(6)}, {x → HH(7)}}
Out[1089]= FRH[HH[3]]
```

$$In[1090]:= \frac{1}{3}\sqrt[3]{\frac{27}{2}-\frac{3\sqrt{69}}{2}}+\frac{\sqrt[3]{\frac{1}{2}\left(9+\sqrt{69}\right)}}{3^{2/3}}$$

```
Out[1090]= FAD[q1, q1 - p, {q2, M}, {q2 - p, m}, q1 - q2]//ToTFi
```

$$In[1091]:= \mathbf{F}^{(D)}_{\{1,0\}\{1,M\}\{1,0\}\{1,m\}\{1,0\}}$$

```
Out[1091]= FromTFi[TFi[D, SPD[p, p], SOD[p], {{1, 0}, {1, M}, {1, 0}, {1, m}, {1, 0}}], q1, q2, p]
```

FUNCTION

Description

FUNCTION[exp, string] is a head of an expression to be declared a function (of type string), if used in Write2.
See also: Write2.

FunctionalD

Description

FunctionalD[expr, {QuantumField[name, {mu}, {a}][p], ...}] calculates the functional derivative of expr with respect to the field list (with incoming momenta p, etc.) and does the fourier transform. FunctionalD[expr, {QuantumField[name, {mu},{a}], ...}] calculates the functional derivative and does partial integration but omits the x-space delta functions.
FunctionalD is a low level function used in FeynRule.
See also: FeynRule, QuantumField.

Examples

Instead of the usual $\frac{1}{q_2^2.((q_1-p)^2-M^2).(q_2-p)^2.(q_1-q_2)^2.(q_1^2-m^2)}$ the arguments and the δ function are omitted, i.e., for the program for simplicity:

```
FromTFi[
  TFi[D, SPD[p, p], SOD[p], {0, 1}, {{1, m}, {1, M}, {1, 0}, {1, m}, {1, 0}}], q1, q2, p]
```

$$In[1092]:= \frac{\Delta \cdot q_2}{(q_1-q_2)^2.(q_2^2-M^2).((q_2-p)^2-m^2).(q_1^2-m^2).(q_1-p)^2}$$

```
Out[1092]= δϕ(x)/δϕ(y) = δ^(D)(x - y)
```

$$In[1093]:= \frac{\delta\phi}{\delta\phi} = 1$$

```
Out[1093]= FunctionalD[QuantumField[ϕ], QuantumField[ϕ]]
```

Instead of the usual 1 the arguments are omitted, and the FunctionalD[QuantumField[ϕ]^2, QuantumField[ϕ]]operator is specified by default to be an integration by parts operator, i.e., the right hand side will be just 2 ϕ or, more precisely (by default) $(\delta\, \partial_\mu \phi(x))/\delta\phi(y) = \partial_\mu \delta^{(D)}(x-y)$.

```
In[1094]:= ∂_μ
Out[1094]= -∂_μ,
```

$$-\left(\vec{\partial}\right)_\mu$$

```
In[1095]:= FunctionalD[QuantumField[PartialD[μ], ϕ], QuantumField[ϕ]]
Out[1095]= -(∂⃗)_μ
```

```
In[1096]:= S[ϕ] = 1/2 ∫ d^D x [ ∂_μ ϕ(x) ∂^μ ϕ(x) - m^2 ϕ(x) ϕ(y)]
Out[1096]= s[ϕ] = (QuantumField[PartialD[μ], ϕ].QuantumField[PartialD[μ], ϕ]-
                   m^2 QuantumField[ϕ].QuantumField[ϕ])/2
```

$\frac{1}{2}\,(\partial_\mu \phi.\partial_\mu \phi - m^2\, \phi.\phi)$

First approach

```
In[1097]:= FunctionalD[s[ϕ], QuantumField[ϕ]]
Out[1097]= -ϕ m^2 - ∂_μ ∂_μ ϕ
```

```
In[1098]:= S[A] = -∫ d^D x 1/4 F_a^μν(x) F_μνa(x)
Out[1098]= F1 = FieldStrength[μ, ν, a, {A, b, c}, 1, Explicit → True]
```

```
In[1099]:= ∂_μ A_ν^a - ∂_ν A_μ^a + A_μ^b . A_ν^c f_abc
Out[1099]= F2 = FieldStrength[μ, ν, a, {A, d, e}, 1, Explicit → True]
```

In order to derive the equation of motion the functional derivative of $\partial_\mu A^a_\nu - \partial_\nu A^a_\mu + A^d_\mu.A^e_\nu\, f_{ade}$ with respect to $S[A] = -1/4$F1.F2has to be set to zero. Bearing in mind that for FeynCalc we have to be precise as to where which operators (coming from the substitution of the derivative of the δ function) act:, act with the functional derivative operator on the first field strength:

$-\frac{1}{4}\left(\partial_\mu A^a_\nu - \partial_\nu A^a_\mu + A^b_\mu.A^c_\nu\, f_{abc}\right).\left(\partial_\mu A^a_\nu - \partial_\nu A^a_\mu + A^d_\mu.A^e_\nu\, f_{ade}\right)S$

A^g_σ

```
In[1100]:= 0 = (δS)/(δA_σ^g(y)) = -2/4 ∫ d^D x (δ/(δA_σ^g(y))
                F_μνa(x))
Out[1100]= F'_a^μν(x)
```

```
In[1101]:= See what happens with just (δS[A])/(δA_σ^g).
Out[1101]= Ag = QuantumField[A, {σ}, {g}]
```

In order to minimize the number of dummy indices, replace b → c.

```
In[1102]:= A_σ^g
```

```
Out[1102]= t1 = FunctionalD[F1, Ag]
```

Instead of inserting the definition for the second $-g^{\nu\sigma}\left(\vec{\partial}\right)_{\mu}\delta_{ag}+g^{\mu\sigma}\left(\vec{\partial}\right)_{\nu}\delta_{ag}+g^{\nu\sigma}A^{b}_{\mu}f_{abg}-g^{\mu\sigma}A^{c}_{\nu}f_{acg}$, introduce a QuantumField object with antisymmetry built into the Lorentz indices:

```
In[1103]:= t1 = t1 /. b → c

In[1104]:= -g^νσ (∂⃗)_μ δ_ag + g^μσ (∂⃗)_ν δ_ag + g^νσ A^c_μ f_acg - g^μσ A^c_ν f_acg
Out[1104]= F^μν_a

In[1105]:= F/: QuantumField[pard___, F, β_, α_, s_] :=
             -QuantumField[pard, F, α, β, s]/; !OrderedQ[{β, α}]
Out[1105]= QuantumField[F, {μ, ν}, {a}]

In[1106]:= F^a_μν
Out[1106]= %/.{μ :→ ν, ν :→ μ}

In[1107]:= -F^a_μν
Out[1107]= t2 = Contract[ExpandPartialD[-1/2 t1.
                    QuantumField[F, LorentzIndex[μ], LorentzIndex[ν], SUNIndex[a]]]]/.Dot → Times

In[1108]:= 1/2 ∂_μ F^g_μσ + 1/2 ∂_ν F^g_νσ - 1/2 A^c_μ F^a_μσ f_acg - 1/2 A^c_ν F^a_νσ f_acg
Out[1108]= t3 = t2 /. ν → μ
```

Since the variational derivative vanishes t4 implies that $0=\partial_{\mu}F^{g}_{\mu\sigma}-A^{c}_{\mu}F^{a}_{\mu\sigma}f_{acg}$.

Second approach

It is of course also possible to do the functional deriviate on the S[A] with both field strength tensors inserted.

```
In[1109]:= t4 = FCE[t3]/. SUNF[a, c, g] → -SUNF[g, c, a]
Out[1109]= ∂_μ F^g_μσ + A^c_μ F^a_μσ f_gca

In[1110]:= D_μ F^μσ_g
Out[1110]= S[A]
```

This is just funcional derivatves and partial integration and simple contraction of indices. No attempt is made to rename dummy indices (since this is difficult in general ...).

With a general replacement rule only valid for commuting fields the color indices can be canonicalized a bit more. The idea is to use the commutative properties of the vector fields, and canonicalize the color indices by a trick. This function will work on any commuting product of fields.

```
In[1111]:= -1/4 (∂_μ A^a_ν - ∂_ν A^a_μ + A^b_μ . A^c_ν f_abc) . (∂_μ A^a_ν - ∂_ν A^a_μ + A^d_μ . A^e_ν f_ade)

In[1112]:= r1 = FunctionalD[S[A], Ag]
```

```
In[1113]:= 1/2 (∂_μ ∂_μ A^g_σ) - 1/2 (∂_μ ∂_σ A^g_μ) + 1/2 (∂_ν ∂_ν A^g_σ) - 1/2 (∂_ν ∂_σ A^g_ν) - 1/4 A^b_μ.∂_μ A^a_σ f_abg + 1/4 A^b_μ.∂_σ A^a_μ f_abg -
           1/4 A^c_ν.∂_ν A^a_σ f_acg + 1/4 A^c_ν.∂_σ A^a_ν f_acg - 1/4 A^b_μ.A^d_μ.A^e_σ f_abg f_ade + 1/4 A^c_ν.A^d_σ.A^e_ν f_acg f_ade -
           1/4 ∂_μ A^a_σ.A^d_μ f_adg + 1/4 ∂_σ A^a_μ.A^d_μ f_adg - 1/4 A^b_μ.A^c_σ.A^d_μ f_abc f_adg - 1/4 ∂_ν A^a_σ.A^e_ν f_aeg + 1/4 ∂_σ A^a_ν.A^e_ν f_aeg +
           1/4 A^b_σ.A^c_ν.A^e_ν f_abc f_aeg + 1/4 A^b_μ.∂_μ A^c_σ f_bcg - 1/4 A^b_σ.∂_ν A^c_ν f_bcg + 1/4 ∂_μ A^b_μ.A^c_σ f_bcg -
           1/4 ∂_ν A^b_σ.A^c_ν f_bcg + 1/4 A^d_μ.∂_μ A^e_σ f_deg - 1/4 A^d_σ.∂_ν A^e_ν f_deg + 1/4 ∂_μ A^d_μ.A^e_σ f_deg - 1/4 ∂_ν A^d_σ.A^e_ν f_deg
Out[1113]= Clear[symfun];

In[1114]:= symfun[z_, fieldname_Symbol] :=
             Expand[SUNSimplify[FixedPoint[Collect2[DotSimplify[#1 /. Times → Dot] /.
                         (((qi___).QuantumField[par1___, fieldname, li1_, sui1_].
                               QuantumField[par2___, fieldname, li2_, sui2_].qf___) any_) :→
                          ((qi.QuantumField[par1, A, li1, sui2].QuantumField[par2, fieldname,
                                li2, sui1].qf) (any /. {sui1 :→ sui2, sui2 :→ sui1})) /;
                           (!(FreeQ2[any, {sui1, su2}])) && !(OrderedQ[{sui1, sui2}]) /.
                     Dot → Times, QuantumField] &, z, 42]]]
Out[1114]= r2 = symfun[r1, A]
```

$$\frac{1}{2}(\partial_\mu\partial_\mu A^g_\sigma) - \frac{1}{2}(\partial_\mu\partial_\sigma A^g_\mu) + \frac{1}{2}(\partial_\nu\partial_\nu A^g_\sigma) - \frac{1}{2}(\partial_\nu\partial$$

Inspection reveals that still terms are the same. Gather the terms with two

$$A^a_\mu\,\partial_\mu A^b_\sigma\,f_{abg} - \frac{1}{2}A^a_\sigma\,\partial_\nu A^b_\nu\,f_{abg} + A^a_\nu\,\partial_\nu A^b_\sigma\,f_{abg}$$

$$\frac{1}{4}A^b_\mu\,A^c_\mu\,A^e_\sigma\,f_{abg}\,f_{ace} - \frac{1}{2}A^b_\nu\,A^c_\nu\,A^e_\sigma\,f_{abg}\,f_{ace} - \frac{1}{2}$$

```
In[1115]:= r3 = r2 /. ν → μ
Out[1115]= ∂_μ ∂_μ A^g_σ - ∂_μ ∂_σ A^g_μ - A^a_σ ∂_μ A^b_μ f_abg + 2 A^a_μ ∂_μ A^b_σ f_abg -
             A^a_μ ∂_σ A^b_μ f_abg - 3/4 A^b_μ A^c_μ A^e_σ f_abg f_ace - 1/4 A^b_μ A^c_μ A^e_σ f_abe f_acg

In[1116]:= f
Out[1116]= twof = Select[r3, Count[#, SUNF[__]] === 2 &]

In[1117]:= -3/4 A^b_μ A^c_μ A^e_σ f_abg f_ace - 1/4 A^b_μ A^c_μ A^e_σ f_abe f_acg
Out[1117]= twofnew = (twof[[1]] + (twof[[2]] /. {b :↦ c, c :↦ b})) /. {a :↦ c, c :↦ a}
```

Check that this is now indeed the same as the t4 result from the first attempt.

```
In[1118]:= -A^a_μ A^b_μ A^e_σ f_ace f_bcg
Out[1118]= r4 = r3 - twof + twofnew

In[1119]:= ∂_μ ∂_μ A^g_σ - ∂_μ ∂_σ A^g_μ - A^a_σ ∂_μ A^b_μ f_abg + 2 A^a_μ ∂_μ A^b_σ f_abg - A^a_μ ∂_σ A^b_μ f_abg - A^a_μ A^b_μ A^e_σ f_ace f_bcg
```

```
Out[1119]= t4
```

```
In[1120]:= ∂_μ F^g_μσ + A^c_μ F^a_μσ f_gca
Out[1120]= w0 = RightPartialD[μ].FieldStrength[μ, σ, g, {A, a, b}, 1]+
               QuantumField[A, LorentzIndex[μ], SUNIndex[c]]
                 FieldStrength[μ, σ, a, {A, b, d}, 1]SUNF[g, c, a]
```

```
In[1121]:= (∂⃗)_μ .F^{g{A,a,b}1}_μσ + F^{a{A,b,d}1}_μσ A^c_μ f_gca
Out[1121]= w1 = Explicit[w0]
```

```
In[1122]:= (∂⃗)_μ . (∂_μ A^g_σ - ∂_σ A^g_μ + A^a_μ . A^b_σ f_abg) - A^c_μ (∂_μ A^a_σ - ∂_σ A^a_μ + A^b_μ . A^d_σ f_abd) f_acg
Out[1122]= w2 = ExpandPartialD[w1] /. Dot → Times
```

```
In[1123]:= ∂_μ ∂_μ A^g_σ - ∂_μ ∂_σ A^g_μ + A^{bc}_σ ∂_μ A^{ac}_μ f_abg + A^{ac}_μ ∂_μ A^{bc}_σ f_abg +
           A^{cc}_μ ∂_σ A^{ac}_μ f_acg - A^c_μ ∂_μ A^a_σ f_acg - A^b_μ A^c_μ A^d_σ f_abd f_acg
Out[1123]= dif1 = w2 - r4
```

quod erat demonstrandum.

```
In[1124]:= -(∂_μ ∂_μ A^g_σ) + ∂_μ ∂_σ A^g_μ + ∂_μ ∂_μ A^g_σ - ∂_μ ∂_σ A^g_μ + A^{bc}_σ ∂_μ A^{ac}_μ f_abg +
           A^{ac}_μ ∂_μ A^{bc}_σ f_abg + A^a_σ ∂_μ A^b_μ f_abg - 2 A^a_μ ∂_μ A^b_σ f_abg + A^a_μ ∂_σ A^b_μ f_abg +
           A^{cc}_μ ∂_σ A^{ac}_μ f_acg - A^c_μ ∂_μ A^a_σ f_acg - A^b_μ A^c_μ A^d_σ f_abd f_acg + A^a_μ A^b_μ A^e_σ f_ace f_bcg
```

Examples of funcional differentiation as used in FeynRule

This is a part of the QCD Lagrangian.

```
In[1125]:= dif2 = symfun[dif1, A]
Out[1125]= -(∂_μ ∂_μ A^g_σ) + ∂_μ ∂_σ A^g_μ + ∂_μ ∂_μ A^g_σ - ∂_μ ∂_σ A^g_μ + A^{bc}_σ ∂_μ A^{ac}_μ f_abg +
           A^{ac}_μ ∂_μ A^{bc}_σ f_abg + A^{bc}_μ ∂_σ A^{ac}_μ f_abg + A^a_σ ∂_μ A^b_μ f_abg - 2 A^a_μ ∂_μ A^b_σ f_abg +
           A^a_μ ∂_σ A^b_μ f_abg + A^a_μ ∂_μ A^b_σ f_abg + A^b_μ A^c_μ A^d_σ f_abd f_acg - A^b_μ A^c_μ A^d_σ f_abd f_acg
```

```
In[1126]:= Unset[s[ϕ]]; Unset[S[A]]; Clear[Ag, F1, F2, t1, t2, t3, t4, F,
             r1, r2, r3, r4, symfun, twof, twofnew, w0, w1, w2, dif1, dif2, dif3]
Out[1126]= (Gstrong * QuantumField[GaugeField, LorentzIndex[li1], SUNIndex[si2]].
              QuantumField[GaugeField, LorentzIndex[li2], SUNIndex[si4]] . QuantumField[
               PartialD[LorentzIndex[li1]], GaugeField, LorentzIndex[li2], SUNIndex[si1]]*
             SUNF[SUNIndex[si1], SUNIndex[si2], SUNIndex[si4]])/4
```

```
In[1127]:= 1/4 g_s A^{si2}_{li1} . A^{si4}_{li2} . ∂_{li1} A^{si1}_{li2} f_{si1si2si4}
```

```
Out[1127]= FunctionalD[%, {QuantumField[GaugeField, {μ}, {a}][p],
             QuantumField[GaugeField, {ν}, {b}][q], QuantumField[GaugeField, {ρ}, {c}][r]}]
```

FV

Description

$$\frac{1}{4} g_s \left(\left(g^{\text{li1}\mu}\, \delta_{a\text{si2}}\right).\left(g^{\text{li2}\nu}\, \delta_{b\text{si4}}\right).\left(-i\, r^{\text{li1}}\, g^{\text{li2}\rho}\, \delta_{c\text{si1}}\right) + \left(g^{\text{li1}\mu}\, \delta_{a\text{si2}}\right).\left(g^{\text{li2}\rho}\, \delta_{c\text{si4}}\right).\left(-i\, q^{\text{li1}}\, g^{\text{li2}\nu}\, \delta_{b\text{si1}}\right) +\right.$$

FV[p, $\left(g^{\text{li1}\nu}\, \delta_{b\text{si2}}\right).\left(g^{\text{li2}\mu}\, \delta_{a\text{si4}}\right).\left(-i\, r^{\text{li1}}\, g^{\text{li2}\rho}\, \delta_{c\text{si1}}\right) + \left(g^{\text{li1}\nu}\, \delta_{b\text{si2}}\right).\left(g^{\text{li2}\rho}\, \delta_{c\text{si4}}\right).\left(-i\, p^{\text{li1}}\, g^{\text{li2}\mu}\, \delta_{a\text{si1}}\right) +$]

$\left.\left(g^{\text{li1}\rho}\, \delta_{c\text{si2}}\right).\left(g^{\text{li2}\mu}\, \delta_{a\text{si4}}\right).\left(-i\, q^{\text{li1}}\, g^{\text{li2}\nu}\, \delta_{b\text{si1}}\right) + \left(g^{\text{li1}\rho}\, \delta_{c\text{si2}}\right).\left(g^{\text{li2}\nu}\, \delta_{b\text{si4}}\right).\left(-i\, p^{\text{li1}}\, g^{\text{li2}\mu}\, \delta_{a\text{si1}}\right)\right) f_{\text{si1si2si4}}$

is the four-dimensional vector Calc[%//Calc]//Factor.
See also: FCE, FCI, FVD, FourVector, Pair.

Examples

```
In[1128]:= 1/4 I g_s (q^μ g^νρ - r^μ g^νρ - g^μρ p^ν + g^μρ r^ν + g^μν p^ρ - g^μν q^ρ) f_abc
Out[1128]= μ

In[1129]:= p^μ
Out[1129]= FV[p, μ]

In[1130]:= p^μ
Out[1130]= FV[p - q, μ]

In[1131]:= p - q^μ
Out[1131]= FV[p, μ]//StandardForm
```

There is no special function to expand momenta in FV; ExpandScalarProduct does the job.

```
In[1132]:= FV[p, μ]
Out[1132]= FCI[FV[p, μ]]//StandardForm

In[1133]:= Pair[LorentzIndex[μ], Momentum[p]]
Out[1133]= ExpandScalarProduct[FV[p - q, μ]]
```

FVD

Description

FVD[p, $p^\mu - q^\mu$] is the D-dimensional vector p with Lorentz index StandardForm[%].
See also: FCE, FCI, FV, FourVector, Pair.

Examples

```
In[1134]:= Pair[LorentzIndex[μ], Momentum[p]] - Pair[LorentzIndex[μ], Momentum[q]]
Out[1134]= μ

In[1135]:= μ
Out[1135]= FVD[p, μ]

In[1136]:= p^μ
Out[1136]= FVD[p - q, μ]

In[1137]:= p - q^μ
Out[1137]= FVD[p, μ]//StandardForm
```

There is no special function to expand momenta in FVD.

```
In[1138]:= FVD[p, μ]
Out[1138]= FCI[FVD[p, μ]]//StandardForm

In[1139]:= Pair[LorentzIndex[μ, D], Momentum[p, D]]
Out[1139]= ExpandScalarProduct[FVD[p - q, μ]]
```

GA

Description

GA[p^μ – q^μ] can be used as input for a 4-dimensional StandardForm[%]and is transformed into DiracGamma[LorentzIndex[Pair[LorentzIndex[μ, D], Momentum[p, D]]−Pair[LorentzIndex[μ, D], Momentum[q, D]]]] by FeynCalcInternal (=FCI).GA[μ ...] is a short form for GA[γ_μ].GA[μ]. ... .

See also: DiracMatrix, GAD, GS.

Examples

```
In[1140]:= μ, ν,
Out[1140]= μ

In[1141]:= ν
Out[1141]= GA[μ]

In[1142]:= γ^μ
Out[1142]= GA[μ, ν] - GA[ν, μ]

In[1143]:= γ^μ.γ^ν - γ^ν.γ^μ
Out[1143]= StandardForm[FCI[GA[μ]]]

In[1144]:= DiracGamma[LorentzIndex[μ]]
Out[1144]= GA[μ, ν, ρ, σ]

In[1145]:= γ^μ.γ^ν.γ^ρ.γ^σ
Out[1145]= StandardForm[GA[μ, ν, ρ, σ]]
```

GAD

Description

GAD[GA[μ].GA[ν].GA[ρ].GA[σ]] can be used as input for a D-dimensional GA[α].(GS[p] + m).GA[β]and is transformed into DiracGamma[LorentzIndex[γ^{α}.(m + $\gamma \cdot p$).γ^{β},D],D] by FeynCalcInternal (=FCI).GAD[μ] is a short form for GAD[γ_{μ}].GAD[μ]. ... .

See also: DiracMatrix, GA, GS.

Examples

```
In[1146]:= μ, ν, ...
Out[1146]= μ

In[1147]:= ν
Out[1147]= GAD[μ]

In[1148]:= γ^μ
Out[1148]= GAD[μ, ν] - GAD[ν, μ]

In[1149]:= γ^μ.γ^ν - γ^ν.γ^μ
Out[1149]= StandardForm[FCI[GAD[μ]]]

In[1150]:= DiracGamma[LorentzIndex[μ, D], D]
Out[1150]= GAD[μ, ν, ρ, σ]

In[1151]:= γ^μ.γ^ν.γ^ρ.γ^σ
Out[1151]= StandardForm[GAD[μ, ν, ρ, σ]]
```

GammaEpsilon

Description

GammaEpsilon[exp] gives a series expansion of Gamma[exp] in Epsilon up to order 6 (where EulerGamma is neglected).

See also: GammaExpand, Series2.

Examples

If the argument is of the form (1+a Epsilon) the result is not calculated but tabulated.

```
In[1152]:= GAD[μ].GAD[ν].GAD[ρ].GAD[σ]
Out[1152]= GAD[α].(GSD[p] + m).GAD[β]

In[1153]:= γ^α.(m + γ · p).γ^β
```

```
Out[1153]= GammaEpsilon[1 + a Epsilon]
```

For other arguments the expansion is calculated.

$$In[1154]:= C\$1075\,\varepsilon^6 + \left(-\frac{1}{5}\zeta(5)\,a^5 - \frac{1}{36}\pi^2\,\zeta(3)\,a^5\right)\varepsilon^5 + \frac{1}{160}a^4\pi^4\varepsilon^4 - \frac{1}{3}a^3\zeta(3)\,\varepsilon^3 + \frac{1}{12}a^2\pi^2\varepsilon^2 + 1$$

```
Out[1154]= GammaEpsilon[1 - Epsilon/2]
```

$$In[1155]:= C\$1076\,\varepsilon^6 + \left(\frac{\pi^2\,\zeta(3)}{1152} + \frac{\zeta(5)}{160}\right)\varepsilon^5 + \frac{\pi^4\,\varepsilon^4}{2560} + \frac{1}{24}\zeta(3)\,\varepsilon^3 + \frac{\pi^2\,\varepsilon^2}{48} + 1$$

```
Out[1155]= GammaEpsilon[Epsilon]
```

GammaExpand

Description

GammaExpand[exp] rewrites Gamma[n + m] in exp (where n has Head Integer).
See also: GammaEpsilon.

Examples

$$In[1156]:= C\$1080\,\varepsilon^6 + \frac{\left(\frac{21}{4}\pi^4\,\psi^{(2)}(1) + \psi^{(6)}(1) - 84\,\pi^2\,\zeta(5)\right)\varepsilon^6}{5040} + \frac{1}{720}\left(\frac{61\,\pi^6}{168} + 10\,\psi^{(2)}(1)^2\right)\varepsilon^5 + \frac{1}{120}\left(\frac{5}{3}\pi^2\,\psi^{(2)}(1) - 24\,\zeta(5)\right)\varepsilon^4 + \frac{\pi^4\,\varepsilon^3}{160} + \frac{1}{6}\psi^{(2)}(1)\,\varepsilon^2 + \frac{\pi^2\,\varepsilon}{12} + \frac{1}{\varepsilon}$$

```
Out[1156]= GammaEpsilon[x]
```

$$In[1157]:= \Gamma(x)$$

```
Out[1157]= GammaExpand[Gamma[2 + Epsilon]]
```

$$In[1158]:= (\varepsilon + 1)\,\Gamma(\varepsilon + 1)$$

```
Out[1158]= GammaExpand[Gamma[-3 + Epsilon]]
```

Gamma1

Description

Gamma1[al,ga, be,de] is a special product of Gamma functions expanded up to order Epsilonˆ2.
See also: Gamma2, Gamma3.

Gamma2

Description

Gamma2[x,y] is a special product of Gamma functions expanded up to order Epsilonˆ3 when positive integer arguments are given.
See also: Gamma1, Gamma3.

Gamma3

Description

Gamma3[al,be,ga,ep] is a special product of Gamma functions expanded up to order Epsilonˆn when positive integer arguments are given (the order n is determined by the option EpsilonOrder).
See also: Gamma1, Gamma2.

Gauge

Description

Gauge is an option for GluonProgagator. If set to 1 the 't Hooft Feynman gauge is used.
See also: $Gauge, GluonProgagator.

GaugeField

Description

GaugeField is just a name. No functional properties are associated with it. GaugeField is used as default setting for the option QuantumField of FieldStrength.
See also: FieldStrength, QuantumField.

Examples

```
In[1159]:= Γ(ε + 1)
           ------------------
           (ε - 3) (ε - 2) (ε - 1) ε
Out[1159]= GammaExpand[Gamma[1 + Epsilon]]

In[1160]:= Γ(ε + 1)
Out[1160]= GaugeField
```

GaugeXi

Description

GaugeXi is a head for gauge parameters.
See also: Gauge, GaugeField.

GA5

Description

GA5 is equivalent to DiracGamma[5] and denotes gamma5.
See also: DiracGamma, GA, GS.

Examples

```
In[1161]:= A
Out[1161]= QuantumField[GaugeField, LorentzIndex[μ], SUNIndex[a]]

In[1162]:= A_μ^a
Out[1162]= GA5
```

GGV

Description

GGV is equivalent to GluonGhostVertex.
See also: GluonGhostVertex.

GhostPropagator

Description

GhostPropagator[p, a, b] gives the ghost propagator where "a" and "b" are the color indices. GhostPropagator[p] omits the γ^5
See also: GHP, GluonPropagator, GluonGhostVertex.

```
In[1163]:= %//Tr
Out[1163]= 0
```

Examples

```
In[1164]:= δab .
Out[1164]= Options[GhostPropagator]

In[1165]:= {Explicit → False}
Out[1165]= GhostPropagator[p, a, b]

In[1166]:= Πab (p)
Out[1166]= GhostPropagator[p, a, b, Explicit → True]

In[1167]:= i δab / p^2
Out[1167]= GhostPropagator[p]

In[1168]:= Πu (p)
Out[1168]= GhostPropagator[p, 1, 2]//Explicit
```

GHP

Description

GHP[p, a, b] gives the ghost propagator where "a" and "b" are the color indices. GHP[p] omits the $\frac{i\,\delta_{ci1ci2}}{p^2}$

See also: GhostPropagator, GluonPropagator, GluonGhostVertex.

Examples

```
In[1169]:= GhostPropagator[-k, 3, 4]//Explicit//FCE//StandardForm
Out[1169]= i FAD[-k] SD[ci3, ci4]

In[1170]:= δab .
Out[1170]= GHP[p, a, b]

In[1171]:= Πab (p)
Out[1171]= GHP[p]//Explicit

In[1172]:= i / p^2
Out[1172]= GHP[p, 1, 2]
```

GluonField

Description

GluonField is a name of a gauge field.
See also: GaugeField.

Examples

```
In[1173]:= Π_{ci1ci2}(p)
Out[1173]= StandardForm[FCE[GHP[-k, 3, 4]//Explicit]]

In[1174]:= i FAD[-k] SD[ci3, ci4]
Out[1174]= GluonField
```

GluonGhostVertex

Description

GluonGhostVertex[{p,mu,a}, {q,nu,b}, {k,rho,c}] or GluonGhostVertex[p,mu,a , q,nu,b , k,rho,c] yields the Gluon-Ghost vertex. The first argument represents the gluon and the third argument the outgoing ghost field (but incoming four-momentum). The dimension and the name of the coupling constant are determined by the options Dimension and CouplingConstant.

See also: GluonPropagator, GluonSelfEnergy, GhostPropagator, GluonVertex.

```
In[1175]:= A
Out[1175]= QuantumField[GluonField, LorentzIndex[μ], SUNIndex[a]]
```

Examples

```
In[1176]:= A^a_μ
Out[1176]= Options[GluonGhostVertex]

In[1177]:= {CouplingConstant → g_s, Dimension → D, Explicit → False}
Out[1177]= GluonGhostVertex[{p, μ, a}, {q, ν, b}, {k, ρ, c}]

In[1178]:= (Λ̃)^μ (k) f_abc
Out[1178]= Explicit[%]

In[1179]:= -g_s k^μ f_abc
Out[1179]= GluonGhostVertex[k, μ]

In[1180]:= (Λ̃)^μ (k)
Out[1180]= Explicit[%]

In[1181]:= -g_s k^μ
Out[1181]= %//Explicit//StandardForm
```

GluonPropagator

Description

GluonPropagator[p, {mu, a}, {nu, b}] or GluonPropagator[p, mu, a , nu, b] yields the gluon propagator. GluonPropagator[p, {mu}, {nu}] or GluonPropagator[p, mu, nu] omits the SUNDelta. The gauge and the dimension is determined by the option Gauge and Dimension. The following settings of Gauge are possible: 1 for the Feynman gauge; −GstrongPair[LorentzIndex[*μ, D*], Momentum[*k, D*]] for the general covariant gauge; {Momentum[n] ,1} for the axial gauge.

GP can be used as an abbreviation of GluonPropagator.

```
In[1182]:= GluonGhostVertex[p, 1, q, 2, k, 3]
Out[1182]= GluonGhostVertex(p, li1, ci1, q, li2, ci2, k, li3, ci3)

In[1183]:= α
Out[1183]= GP
```

See also: GluonSelfEnergy, GluonVertex, GhostPropagator, GluonGhostVertex.

GluonPropagator

```
In[1184]:= Options[GluonPropagator]
Out[1184]= {CounterTerm → False, CouplingConstant → g_s, Dimension → D, Explicit → False, Gauge →
1, Ω → False}

In[1185]:= Examples
Out[1185]= GluonPropagator[p, μ, a, ν, b]

In[1186]:= Π^{μν}_{ab}(p)
Out[1186]= GluonPropagator[p, μ, a, ν, b]//Explicit

In[1187]:= -i g^{μν} δ_{ab} / p^2
Out[1187]= GluonPropagator[p, μ, a, ν, b, Gauge → α]//Explicit

In[1188]:= -i α p^μ p^ν δ_{ab} (1/p^2)^2 + i p^μ p^ν δ_{ab} (1/p^2)^2 - i g^{μν} δ_{ab} / p^2
Out[1188]= GluonPropagator[p, μ, a, ν, b, Gauge → {Momentum[n], 1}]//Explicit
```

This is a convenient way to enter amplitudes by hand (GP is an abbreviation GluonPropagator).

```
In[1189]:= -i g^{μν} δ_{ab} / p^2 + i n^μ n^ν p^2 δ_{ab} / (p^2 n·p^2) + i p^μ n^ν δ_{ab} / (p^2 n·p) + i n^μ p^ν δ_{ab} / (p^2 n·p) - i p^μ p^ν n^2 δ_{ab} / (p^2 n·p^2)
Out[1189]= GluonPropagator[p, μ, ν]//Explicit

In[1190]:= -i g^{μν} / p^2
Out[1190]= Explicit[GP[p, 1, 2], Gauge → ξ]

In[1191]:= -i ξ p^{li1} p^{li2} δ_{ci1ci2} (1/p^2)^2 + i p^{li1} p^{li2} δ_{ci1ci2} (1/p^2)^2 - i g^{li1li2} δ_{ci1ci2} / p^2
Out[1191]= GP[-k, 3, 4]//Explicit//FCE//StandardForm
```

```
In[1192]:= -i FAD[-k] MTD[li3, li4] SD[ci3, ci4]
Out[1192]= GluonPropagator[p, μ, a, ν, b, CounterTerm → 1]//Explicit
```

$$\text{In[1193]:=}\quad \frac{19\, i\, C_A\, g_s^2\, S_n\, g^{\mu\nu}\, p^2\, \delta_{ab}}{6\,\varepsilon} - \frac{11\, i\, C_A\, g_s^2\, S_n\, p^\mu\, p^\nu\, \delta_{ab}}{3\,\varepsilon}$$

```
Out[1193]= GluonPropagator[p, μ, a, ν, b, CounterTerm → 2]//Explicit
```

$$\text{In[1194]:=}\quad \frac{i\, C_A\, S_n\, p^\mu\, p^\nu\, \delta_{ab}\, g_s^2}{3\,\varepsilon} + \frac{i\, C_A\, S_n\, g^{\mu\nu}\, p^2\, \delta_{ab}\, g_s^2}{6\,\varepsilon}$$

```
Out[1194]= GluonPropagator[p, μ, a, ν, b, CounterTerm → 3]//Explicit
```

$$\text{In[1195]:=}\quad \frac{8\, i\, g_s^2\, S_n\, T_f\, p^\mu\, p^\nu\, \delta_{ab}}{3\,\varepsilon} - \frac{8\, i\, g_s^2\, S_n\, T_f\, g^{\mu\nu}\, p^2\, \delta_{ab}}{3\,\varepsilon}$$

```
Out[1195]= GluonPropagator[p, μ, a, ν, b, CounterTerm → 4]//Explicit
```

GluonSelfEnergy

Description

GluonSelfEnergy[{mu, a}, {nu,b}] yields the 1-loop Gluon selfenergy.

$$\text{In[1196]:=}\quad \frac{10\, i\, C_A\, g_s^2\, S_n\, g^{\mu\nu}\, p^2\, \delta_{ab}}{3\,\varepsilon} - \frac{10\, i\, C_A\, g_s^2\, S_n\, p^\mu\, p^\nu\, \delta_{ab}}{3\,\varepsilon}$$

```
Out[1196]= GluonPropagator[p, μ, a, ν, b, CounterTerm → 5]//Explicit
```

See also: GluonPropagator, GluonVertex, GhostPropagator, GluonGhostVertex.

$$\frac{10\, i\, C_A\, S_n\, p^\mu\, p^\nu\, \delta_{ab}\, g_s^2}{3\,\varepsilon} - \frac{4\, i\, S_n\, T_f\, p^\mu\, p^\nu\, \delta_{ab}\, g_s^2}{3\,\varepsilon} - \frac{10\, i\, C_A\, S_n\, g^{\mu\nu}\, p^2\, \delta_{ab}\, g_s^2}{3\,\varepsilon} + \frac{4\, i\, S_n\, T_f\, g^{\mu\nu}\, p^2\, \delta_{ab}\, g_s^2}{3\,\varepsilon}$$

```
In[1197]:= Options[GluonSelfEnergy]
Out[1197]= {Dimension → D, CouplingConstant → g_s,
              FinalSubstitutions → {Log(μ^2_) :→ 0, Γ_E :→ Log(4 π)}, Gauge → 1, Momentum → p}
```

GluonVertex

Description

Note: All momenta are flowing into the vertex.

GluonVertex[{p,mu,a}, {q,nu,b}, {k,la,c}] or GluonVertex[p,mu,a , q,nu,b , k,la,c] yields the 3-gluon vertex. GluonVertex[{p,mu}, {q,nu}, {k,la}] yields the 3-gluon vertex without color structure and the coupling constant. GluonVertex[{p,mu,a}, {q,nu,b}, {k,la,c}, {s,si,d}] or GluonVertex[{mu,a}, {nu,b}, {la,c}, {si,d}] or GluonVertex[p,mu,a , q,nu,b , k,la,c , s,si,d] or GluonVertex[mu,a , nu,b , la,c , si,d] yields the 4-gluon vertex. The dimension and the name of the coupling constant are determined by the options Dimension and CouplingConstant.

GV can be used as an abbreviation of GluonVertex.

```
In[1198]:= Examples
Out[1198]= GluonSelfEnergy[{μ, a}, {ν, b}]

In[1199]:= 1/2 i C_A (20/(3 ε) - 62/9) (p^μ p^ν - g^μν p^2) δ_ab g_s^2 + i (20/9 - 8/(3 ε)) T_f (p^μ p^ν - g^μν p^2) δ_ab g_s^2
Out[1199]= GV
```

See also: GluonGhostVertex, GluonPropagator.

Examples

```
In[1200]:= GluonVertex
Out[1200]= Options[GluonVertex]

In[1201]:= {CouplingConstant → g_s, Dimension → D, Explicit → False, Ω → False}
Out[1201]= GluonVertex[{p, μ, a}, {q, ν, b}, {r, ρ, c}]

In[1202]:= V^μνρ(p, q, r) f_abc
Out[1202]= GluonVertex[{p, μ, a}, {q, ν, b}, {r, ρ, c}]//Explicit

In[1203]:= (g_s q^μ g^νρ - g_s r^μ g^νρ - g_s g^μρ p^ν + g_s g^μρ r^ν + g_s g^μν p^ρ - g_s g^μν q^ρ) f_abc
Out[1203]= GluonVertex[{p, μ}, {q, ν}, {r, ρ}]

In[1204]:= V^μνρ(p, q, r)
Out[1204]= GluonVertex[{p, μ}, {q, ν}, {r, ρ}]//Explicit

In[1205]:= g_s q^μ g^νρ - g_s r^μ g^νρ - g_s g^μρ p^ν + g_s g^μρ r^ν + g_s g^μν p^ρ - g_s g^μν q^ρ
Out[1205]= GluonVertex[{p, μ, a}, {q, ν, b}, {r, ρ, c}, {s, σ, d}]

In[1206]:= V^μνρσ_abcd(p, q, r, s)
Out[1206]= GluonVertex[{p, μ, a}, {q, ν, b}, {r, ρ, c}, {s, σ, d}]//Explicit

In[1207]:= i g^μρ g^νσ f_adu9 f_bcu9 g_s^2 - i g^μν g^ρσ f_adu9 f_bcu9 g_s^2 + i g^μσ g^νρ f_acu9 f_bdu9 g_s^2 -
           i g^μν g^ρσ f_acu9 f_bdu9 g_s^2 + i g^μσ g^νρ f_abu9 f_cdu9 g_s^2 - i g^μρ g^νσ f_abu9 f_cdu9 g_s^2
Out[1207]= GluonVertex[{μ, a}, {ν, b}, {ρ, c}, {σ, d}]
```

A very convenient way to enter diagrams by hand is to label each line hitting a vertex by a number and put this number after the inflowing momentum.

```
In[1208]:= W^abcd
Out[1208]= GluonVertex[μ, a, ν, b, ρ, c, σ, d]//Explicit

In[1209]:= i g^μρ g^νσ f_adu10 f_bcu10 g_s^2 - i g^μν g^ρσ f_adu10 f_bcu10 g_s^2 + i g^μσ g^νρ f_acu10 f_bdu10 g_s^2 -
           i g^μν g^ρσ f_acu10 f_bdu10 g_s^2 + i g^μσ g^νρ f_abu10 f_cdu10 g_s^2 - i g^μρ g^νσ f_abu10 f_cdu10 g_s^2
Out[1209]= GV[p, 1, q, 2, r, 3]
```

GO

Description

GO is equivalent to Twist2GluonOperator.
See also: Twist2GluonOperator.

GP

Description

GP is equivalent to GluonPropagator.
See also: GluonPropagator.

GrassmannParity

Description

GrassmannParity is a data type. E.g. DataType[F, GrassmannParity] = 1 declares F to be of bosonic type and DataType[F, GrassmannParity] = -1 of fermionic one.
See also: DataType.

GS

Description

GS[p] can be used as input for a 4-dimensional $V^{\text{li1li2li3}}(p,\ q,\ r)\ f_{\text{ci1ci2ci3}}$ and is transformed into DiracGamma[Momentum[p]] by FeynCalcInternal (=FCI). GS[p,q, ...] is a short form for GS[p].GS[q]. ... .
See also: DiracGamma, DiracSlash, GA, GAD.

Examples

```
In[1210]:= GV[p, 1, q, 2, r, 3, s, 4]//Explicit
```

$$\begin{aligned}\text{Out[1210]}= &\ i\, g^{\text{li1li3}}\, g^{\text{li2li4}}\, f_{\text{ci1ci4u11}}\, f_{\text{ci2ci3u11}}\, g_s^2 - i\, g^{\text{li1li2}}\, g^{\text{li3li4}}\, f_{\text{ci1ci4u11}}\, f_{\text{ci2ci3u11}}\, g_s^2 + \\ &\ i\, g^{\text{li1li4}}\, g^{\text{li2li3}}\, f_{\text{ci1ci3u11}}\, f_{\text{ci2ci4u11}}\, g_s^2 - i\, g^{\text{li1li2}}\, g^{\text{li3li4}}\, f_{\text{ci1ci3u11}}\, f_{\text{ci2ci4u11}}\, g_s^2 + \\ &\ i\, g^{\text{li1li4}}\, g^{\text{li2li3}}\, f_{\text{ci1ci2u11}}\, f_{\text{ci3ci4u11}}\, g_s^2 - i\, g^{\text{li1li3}}\, g^{\text{li2li4}}\, f_{\text{ci1ci2u11}}\, f_{\text{ci3ci4u11}}\, g_s^2\end{aligned}$$

```
In[1211]:= p / (= γ.p = γ_μ p^μ)
Out[1211]= GS[p]
```

```
In[1212]:= γ · p
Out[1212]= GS[p]//FCI//StandardForm

In[1213]:= DiracGamma[Momentum[p]]
Out[1213]= GS[p, q, r, s]

In[1214]:= (γ · p).(γ · q).(γ · r).(γ · s)
Out[1214]= GS[p, q, r, s]//StandardForm
```

GSD

Description

GSD[p] can be used as input for a D-dimensional GS[*p*].GS[*q*].GS[*r*].GS[*s*] and is transformed into DiracGamma[Momentum[p,D],D] by FeynCalcInternal (=FCI). GSD[p,q, ...] is a short form for GSD[p].GSD[q]. ... .

See also: DiracGamma, DiracSlash, GA, GAD.

Examples

```
In[1215]:= GS[q].(GS[p] + m).GS[q]
Out[1215]= (γ · q).(m + γ · p).(γ · q)

In[1216]:= p / (= γ.p = γ_μ p^μ)
Out[1216]= GSD[p]

In[1217]:= γ · p
Out[1217]= GSD[p]//FCI//StandardForm

In[1218]:= DiracGamma[Momentum[p, D], D]
Out[1218]= GSD[p, q, r, s]

In[1219]:= (γ · p).(γ · q).(γ · r).(γ · s)
Out[1219]= GSD[p, q, r, s]//StandardForm
```

Gstrong

Description

Gstrong denotes the strong coupling constant.

See also: CovariantD, FieldStrength, GluonVertex.

Examples

Gstrong has no functional properties. Only a typesetting rule is defined.

```
In[1220]:= GSD[p].GSD[q].GSD[r].GSD[s]
Out[1220]= GSD[q].(GSD[p] + m).GSD[q]
```

GTI

Description

GTI is like RHI, but with no functional properties.
See also: RHI.

GV

Description

GV is equivalent to GluonVertex.
See also: GluonVertex.

Hill

Description

Hill[x, y] gives the Hill identity with arguments x and y. The returned object is 0.
See also: SimplifyPolyLog.

Examples

```
In[1221]:= (γ · q).(m + γ · p).(γ · q)
Out[1221]= Gstrong

In[1222]:= g_s
Out[1222]= Hill[a, b]
```

$$\text{In[1223]:=}\quad \text{Log}(a)\,(\text{Log}(1-a)-\text{Log}(1-b))+\text{Log}\left(\frac{1-a}{1-b}\right)\left(-\text{Log}(a)+\text{Log}(1-b)-\text{Log}\left(\frac{a-b}{a}\right)+\text{Log}\left(\frac{a-b}{1-b}\right)\right)-$$
$$\left(\text{Log}(1-b)-\text{Log}\left(\frac{a-b}{a}\right)+\text{Log}\left(\frac{a-b}{a\,(1-b)}\right)\right)\text{Log}\left(\frac{(1-a)\,b}{a\,(1-b)}\right)+$$
$$\text{Li}_2(a)+\text{Li}_2\left(\frac{1-a}{1-b}\right)-\text{Li}_2(b)+\text{Li}_2\left(\frac{b}{a}\right)-\text{Li}_2\left(\frac{(1-a)\,b}{a\,(1-b)}\right)-\frac{\pi^2}{6}$$

```
Out[1223]= % /. a :→ 0.123 /. b :→ 0.656 // Chop
```

```
In[1224]:= 0
Out[1224]= Hill[x, x y]//PowerExpand//SimplifyPolyLog//Expand
```

HypergeometricAC

Description

HypergeometricAC[n][exp] analytically continues Hypergeometric2F1 functions in exp. The second argument n refers to the equation number (n) in chapter 2.10 of "Higher Transcendental Functions" by Ergelyi, Magnus, Oberhettinger, Tricomi. In case of eq. (6) (p.109) the last line is returned for HypergeometricAC[6][exp], while the first equality is given by HypergeometricAC[61][exp]. ((2.10.1) is identical to eq. (9.5.7) of "Special Functions & their Applications" by N.N.Lebedev).

$$\text{In[1225]:=}\quad \zeta(2) - \text{Log}(1-y)\,\text{Log}(y) - \text{Log}(x)\,\text{Log}(1-x\,y) - \text{Li}_2(1-x) - \text{Li}_2(1-y) - \text{Li}_2(x\,y) + \text{Li}_2\left(\frac{1-x}{1-x\,y}\right) - \text{Li}_2\left(\frac{(1-x)\,y}{1-x\,y}\right)$$

```
Out[1225]= % /. x :> 0.34/. y → 0.6//N//Chop
```

See also: HypExplicit, HypergeometricIR, HypergeometricSE, ToHypergeometric.

Examples

These are all transformation rules currently built in.

```
In[1226]:= 0
Out[1226]= Options[HypergeometricAC]

In[1227]:= {Collect2 → True}
Out[1227]= HypergeometricAC[1][Hypergeometric2F1[α, β, γ, z]]
```

$$\text{In[1228]:=}\quad \frac{\Gamma(\alpha+\beta-\gamma)\,\Gamma(\gamma)\,{}_2F_1(\gamma-\alpha,\gamma-\beta;-\alpha-\beta+\gamma+1;1-z)\,(1-z)^{-\alpha-\beta+\gamma}}{\Gamma(\alpha)\,\Gamma(\beta)} + \frac{\Gamma(\gamma)\,\Gamma(-\alpha-\beta+\gamma)\,{}_2F_1(\alpha,\beta;\alpha+\beta-\gamma+1;1-z)}{\Gamma(\gamma-\alpha)\,\Gamma(\gamma-\beta)}$$

```
Out[1228]= HypergeometricAC[2][Hypergeometric2F1[α, β, γ, z]]
```

$$\text{In[1229]:=}\quad \frac{\Gamma(\beta-\alpha)\,\Gamma(\gamma)\,{}_2F_1\left(\alpha,\alpha-\gamma+1;\alpha-\beta+1;\frac{1}{z}\right)\,(-z)^{-\alpha}}{\Gamma(\beta)\,\Gamma(\gamma-\alpha)} + \frac{\Gamma(\alpha-\beta)\,\Gamma(\gamma)\,{}_2F_1\left(\beta,\beta-\gamma+1;-\alpha+\beta+1;\frac{1}{z}\right)\,(-z)^{-\beta}}{\Gamma(\alpha)\,\Gamma(\gamma-\beta)}$$

```
Out[1229]= HypergeometricAC[3][Hypergeometric2F1[α, β, γ, z]]
```

$$\text{In[1230]:=}\quad \frac{\Gamma(\beta-\alpha)\,\Gamma(\gamma)\,{}_2F_1\left(\alpha,\gamma-\beta;\alpha-\beta+1;\frac{1}{1-z}\right)\,(1-z)^{-\alpha}}{\Gamma(\beta)\,\Gamma(\gamma-\alpha)} + \frac{\Gamma(\alpha-\beta)\,\Gamma(\gamma)\,{}_2F_1\left(\beta,\gamma-\alpha;-\alpha+\beta+1;\frac{1}{1-z}\right)\,(1-z)^{-\beta}}{\Gamma(\alpha)\,\Gamma(\gamma-\beta)}$$

```
Out[1230]= HypergeometricAC[4][Hypergeometric2F1[α, β, γ, z]]
```

$$\text{In[1231]:=}\quad \frac{\Gamma(\gamma)\,\Gamma(-\alpha-\beta+\gamma)\,{}_2F_1\left(\alpha,\alpha-\gamma+1;\alpha+\beta-\gamma+1;-\frac{1-z}{z}\right)\,z^{-\alpha}}{\Gamma(\gamma-\alpha)\,\Gamma(\gamma-\beta)} + \frac{(1-z)^{-\alpha-\beta+\gamma}\,\Gamma(\alpha+\beta-\gamma)\,\Gamma(\gamma)\,{}_2F_1\left(\gamma-\alpha,1-\alpha;-\alpha-\beta+\gamma+1;-\frac{1-z}{z}\right)\,z^{\alpha-\gamma}}{\Gamma(\alpha)\,\Gamma(\beta)}$$

```
Out[1231]= HypergeometricAC[6][Hypergeometric2F1[α, β, γ, z]]
```

HypergeometricIR

Description

HypergeometricIR[exp, t] substitutes for all Hypergeometric2F1[a,b,c,z] in exp by its Euler integral reprentation. The factor Integratedx[t, 0, 1] can be omitted by setting the option Integratedx → False.

See also: HypergeometricAC, HypergeometricSE, ToHypergeometric.

```
In[1232]:= (1 - z)^-β 2F1(β, γ - α; γ; -z/(1 - z))
Out[1232]= HypergeometricAC[61][Hypergeometric2F1[α, β, γ, z]]
```

Examples

```
In[1233]:= (1 - z)^-α 2F1(α, γ - β; γ; -z/(1 - z))
Out[1233]= Options[HypergeometricIR]

In[1234]:= {Integratedx → False}
Out[1234]= HypergeometricIR[Hypergeometric2F1[a, b, c, z], t]

                                  (1 - t)^(-b+c-1) t^(b-1) (1 - t z)^-a Γ(c)
In[1235]:= ------------------------------------------------------------------
                                          Γ(b) Γ(c - b)
Out[1235]= ToHypergeometric[t^b (1 - t)^c (1 + t z)^a, t]
```

HypergeometricSE

Description

HypergeometricSE[exp, $\frac{\Gamma(b+1)\,\Gamma(c+1)\,{}_2F_1(-a,b+1;b+c+2;-z)}{\Gamma(b+c+2)}$] expresses Hypergeometric functions by their series expansion in terms of a sum (the Sum is omitted and HypergeometricIR[%, t], running from 0 to $(1-t)^c t^b (t\,z+1)^a$, is the summation index).

```
In[1236]:= ν
Out[1236]= ν
```

See also: HypergeometricIR.

Examples

```
In[1237]:= ∞
Out[1237]= Options[HypergeometricSE]

In[1238]:= {Simplify → FullSimplify}
Out[1238]= HypergeometricSE[Hypergeometric2F1[a, b, c, z], ν]
```

HypExplicit

Description

HypExplicit[exp, $\frac{z^{\nu}\,\Gamma(c)\,\Gamma(a+\nu)\,\Gamma(b+\nu)}{\Gamma(a)\,\Gamma(b)\,\Gamma(\nu+1)\,\Gamma(c+\nu)}$] expresses Hypergeometric functions in exp by their definition in terms of a sum (the Sum is omitted and HypergeometricSE[HypergeometricPFQ[{*a, b, c*}, {*d, e*}, *z*], *ν*] is the summation index).

See also: HypergeometricIR.

Examples

```
In[1239]:= z^ν Γ(d) Γ(e) Γ(a + ν) Γ(b + ν) Γ(c + ν)
           ---------------------------------------
           Γ(a) Γ(b) Γ(c) Γ(ν + 1) Γ(d + ν) Γ(e + ν)
Out[1239]= ν

In[1240]:= ν
Out[1240]= Hypergeometric2F1[a, b, c, z]

In[1241]:= 2F1(a, b; c; z)
Out[1241]= HypExplicit[%, ν]

In[1242]:= z^ν Γ(c) Γ(a + ν) Γ(b + ν)
           -------------------------
           Γ(a) Γ(b) Γ(ν + 1) Γ(c + ν)
Out[1242]= HypergeometricPFQ[{a, b, c}, {d, e}, z]
```

HypInt

Description

HypInt[exp, t] substitutes for all Hypergeometric2F1[a,b,c,z] in exp Gamma[c]/(Gamma[b] Gamma[c-b]) Integratedx[t,0,1] tˆ(b-1) (1-t)ˆ(c-b-1) (1-t z)ˆ(-a).

Examples

```
In[1243]:= 3F2(a, b, c; d, e; z)
Out[1243]= HypExplicit[%, ν]

In[1244]:= z^ν Γ(d) Γ(e) Γ(a + ν) Γ(b + ν) Γ(c + ν)
           ---------------------------------------
           Γ(a) Γ(b) Γ(c) Γ(ν + 1) Γ(d + ν) Γ(e + ν)
Out[1244]= Hypergeometric2F1[a, b, c, z]
```

IFPD

Description

IFPD[p, m] denotes (p^2 - m^2).
See also: PropagatorDenominator.

IFPDOff

Description

IFPDOff[exp_,q1_, q2_, ...] changes from IFPD representation to FeynAmpDenominator[...]. The q1, q2, ... are the integration momenta.
See also: IFPD, IFPDOn.

Examples

```
In[1245]:= ₂F₁(a, b; c; z)
Out[1245]= HypInt[%, t]

In[1246]:= (1 - t)^(-b+c-1) t^(b-1) (1 - t z)^(-a) Γ(c) ∫₀¹ dt / (Γ(b) Γ(c - b))
Out[1246]= IFPD[Momentum[p], m]

In[1247]:= (p^2 - m^2)
Out[1247]= %//StandardForm

In[1248]:= IFPD[Momentum[p], m]
Out[1248]= IFPDOff[%, p]
```

IFPDOn

Description

IFPDOn[exp_,q1_, q2_, ...] changes from FeynAmpDenominator[...] representation to the IFPD one (Inverse Feynman Propagator Denominator). I.e., FeynAmpDenominator[PropagatorDenominator[a,b]] is replaced by 1/IFPD[a,b] and The q1, q2, ... are the integration momenta.
See also: IFPD, IFPDOff.

IncludePair

Description

IncludePair is an option for FC2RHI. Possible settings are True and False.
See also: FC2RHI.

IndexPosition

Description

IndexPosition is an option for FieldStrength.
See also: FieldStrength.

Indices

Description

Indices is an option for FORM2FeynCalc. Its default setting is Automatic. It may be set to a list, if the FORM-file does not contain a I(ndices) statement.
See also: FORM2FeynCalc.

InitialFunction

Description

InitialFunction is an option of FeynRule the setting of which is applied to the first argument of FeynRule before anything else
See also: FeynRule.

InitialSubstitutions

Description

InitialSubstitutions is an option for OneLoop and OneLoopSum and Write2. All substitutions indicated hereby are done at the end of the calculation.
See also: OneLoop, OneLoopSum, Write2.

InsideDiracTrace

Description

InsideDiracTrace is and option of DiracSimplify. If set to True, DiracSimplify assumes to operate inside a DiracTrace, i.e., products of an odd number of Dirac matrices are discarded. Furthermore simple traces are calculated (but divided by a factor 4, i.e. : DiracSimplify[DiracMatrix[a,b], InsideDiracTrace→True] yields ScalarProduct[a,b]) Traces involving more than four DiracGamma's and DiracGamma[5] are not performed.

See also: DiracSimplify, DiracTrace.

IntegralTable

Description

IntegralTable is an option of OneLoopSimplify, TwoLoopSimplify and FeynAmpDenominatorSimplify. It may be set to a list of the form : {FCIntegral[...] :→ bla, ...}.

See also: OneLoopSimplify, TwoLoopSimplify, FeynAmpDenominatorSimplify.

IntegrateByParts

Description

IntegrateByParts[(1-t)^(a Epsilon -1) g[t], deriv, t] does an integration by parts of the definite integral over t from 0 to 1.

```
In[1249]:= p^2 - m^2
Out[1249]= %//StandardForm
```

Examples

```
In[1250]:= -m^2 + Pair[Momentum[p], Momentum[p]]
Out[1250]= Options[IntegrateByParts]
```

Integratedx

Description

Integratedx[x, low, up] is a variable representing the integration operator Integrate[#, {x,low,up}]&.

See also: FeynmanParametrize, CalculateCounterTerm, TimedIntegrate, HypergeometricIR, HypInt, TimedIntegrate.

Integrate2

Description

Integrate2 is like Integrate, but : Integrate2[a_Plus, b__] := Map[Integrate2[#, b]&, a] (more linear algebra and partial fraction decomposition is done) Integrate2[f[x] DeltaFunction[x], x] → f[0] Integrate2[f[x] DeltaFunction[x0-x], x] → f[x0]. Integrate2[f[x] DeltaFunction[a + b x], x] → Integrate[f[x] (1/Abs[b]) DeltaFunction[a/b + x], x], where abs[b] → b, if b is a Symbol, and if b = -c, then abs[-c] → c, i.e., the variable contained in b is supposed to be positive. {Hold → False} is replaced by 6 Zeta2. Integrate2[1/(1-y),{y,x,1}] is intepreted as distribution, i.e. as Integrate2[-1/(1-y)],{y, 0, x}] → Log[1-y]. Integrate2[1/(1-x),{x,0,1}] → 0.

See also: DeltaFunction, Integrate3, SumS, SumT.

NOTE: Since Integrate2 does do a reordering and partial fraction decomposition before calling the integral table of Integrate3 it will in general be slower compared to Integrate3 for sums of integrals. I.e., if the integrand has already an expanded form and if partial fraction decomposition is not necessary it is more effective to use Integrate3.

Examples

```
In[1251]:= IntegrateByParts[(1 - t)^(a Epsilon - 1) g[t], (1 - t)^(a Epsilon - 1), t]
Out[1251]= g'(t) (1 - t)^(a ε) / (a ε) + g(0) / (a ε)
```

Since Integrate2 uses table-look-up methods it is much faster than *Mathematica*'s Integrate.

```
In[1252]:= π^2
Out[1252]= Integrate2[Log[1 + x] Log[x]/(1 - x), {x, 0, 1}]//Timing

In[1253]:= {0.05 Second, ζ(3) - 3/2 ζ(2) Log(2)}
Out[1253]= Integrate2[PolyLog[2, x^2], {x, 0, 1}]

In[1254]:= ζ(2) + 4 Log(2) - 4
Out[1254]= Integrate2[PolyLog[3, -x], {x, 0, 1}]

In[1255]:= ζ(2)/2 - 3 ζ(3)/4 - 2 Log(2) + 1
Out[1255]= Integrate2[PolyLog[3, 1/(1 + x)], {x, 0, 1}]
```

Integrate2 does integration in a Hadamard sense, i.e., $-\log(2)\zeta(2)+\frac{3\,\zeta(3)}{4}+\frac{\log^3(2)}{3}-\log^2(2)+2\log(2)$ means acutally expanding the result of Integrate2[DeltaFunction[$1-x$] $f[x]$, {x, 0, 1}]up do $f(1)$ and neglecting all $\int_0^1 f(x)dx$ - dependent terms. E.g. $\int_\delta^{1-\delta} f(x)dx$

```
In[1256]:= O(δ)
Out[1256]= δ
```

In the physics literature sometimes the "+" notation is used. In FeynCalc the $\int_\delta^{1-\delta} 1/(1-x)dx = -\log(1-x)|_\delta^{1-\delta} = -\log(\delta)+\log(1) \Rightarrow 0.$ is represented by Integrate2[1/($1-x$), {x, 0, 1}] or just 0

In[1257]:= $\left(\frac{1}{1-x}\right)_+$

Out[1257]= PlusDistribution[1/(1 - x)]

In[1258]:= **1/(1 - x) .**

Out[1258]= Integrate2[PlusDistribution[1/(1 - x)], {x, 0, 1}]

In[1259]:= **0**

Out[1259]= Integrate2[PolyLog[2, 1 - x]/(1 - x)^2, {x, 0, 1}]

In[1260]:= **2 - ζ(2)**

Out[1260]= Integrate2[(Log[x] Log[1 + x])/(1 + x), {x, 0, 1}]

In[1261]:= $-\frac{\zeta(3)}{8}$

Out[1261]= Integrate2[Log[x]^2/(1 - x), {x, 0, 1}]

In[1262]:= **2 ζ(3)**

Out[1262]= Integrate2[PolyLog[2, -x]/(1 + x), {x, 0, 1}]

In[1263]:= $\frac{\zeta(3)}{4} - \frac{1}{2}\zeta(2)\,\mathrm{Log}(2)$

Out[1263]= Integrate2[Log[x] PolyLog[2, x], {x, 0, 1}]

In[1264]:= **3 - 2 ζ(2)**

Out[1264]= Integrate2[x PolyLog[3, x], {x, 0, 1}]

In[1265]:= $-\frac{\zeta(2)}{4} + \frac{\zeta(3)}{2} + \frac{3}{16}$

Out[1265]= Integrate2[(Log[x]^2 Log[1 - x])/(1 + x), {x, 0, 1}]

In[1266]:= $\mathrm{Log}^2(2)\,\zeta(2) - 4\,\mathrm{Li}_4\left(\frac{1}{2}\right) - \frac{\mathrm{Log}^4(2)}{6} + \frac{\pi^4}{90}$

Out[1266]= Integrate2[PolyLog[2, ((x (1 - z) + z) (1 - x + x z))/z]/(1 - x + x z), {x, 0, 1}]

In[1267]:=
$$\frac{\mathrm{Log}^3(z)}{6(1-z)} - \frac{\mathrm{Log}(1-z)\,\mathrm{Log}^2(z)}{1-z} - \frac{\mathrm{Log}(z+1)\,\mathrm{Log}^2(z)}{1-z} - \frac{i\pi\,\mathrm{Log}^2(z)}{2(1-z)} - \frac{\zeta(2)\,\mathrm{Log}(z)}{1-z} +$$
$$\frac{4\,\mathrm{Log}(1-z)\,\mathrm{Log}(z+1)\,\mathrm{Log}(z)}{1-z} + \frac{2i\pi\,\mathrm{Log}(z+1)\,\mathrm{Log}(z)}{1-z} - \frac{2\,\mathrm{Li}_2(1-z)\,\mathrm{Log}(z)}{1-z} - \frac{2\,\mathrm{Li}_2(-z)\,\mathrm{Log}(z)}{1-z} -$$
$$\frac{2\,\mathrm{Log}^3(z+1)}{3(1-z)} + \frac{i\pi\zeta(2)}{1-z} + \frac{2\zeta(2)\,\mathrm{Log}(1-z)}{1-z} + \frac{2\zeta(2)\,\mathrm{Log}(z+1)}{1-z} - \frac{2\,S_{12}(1-z)}{1-z} + \frac{4\,\mathrm{Log}(1-z)\,\mathrm{Li}_2(-z)}{1-z} +$$
$$\frac{2i\pi\,\mathrm{Li}_2(-z)}{1-z} + \frac{4\,\mathrm{Li}_3(1-z)}{1-z} + \frac{2\,\mathrm{Li}_3(-z)}{1-z} + \frac{4\,\mathrm{Li}_3\left(\frac{1}{z+1}\right)}{1-z} + \frac{4\,\mathrm{Li}_3\left(-\frac{1-z}{z+1}\right)}{1-z} - \frac{4\,\mathrm{Li}_3\left(\frac{1-z}{z+1}\right)}{1-z} - \frac{2\zeta(3)}{1-z}$$

Out[1267]= Apart[Integrate2[x^(OPEm - 1) PolyLog[3, 1 - x], {x, 0, 1}], OPEm]

In[1268]:= $-\frac{\zeta(2)}{m-1} - \frac{\zeta(2)}{m^2} + \frac{-S_1(m-2)\,\zeta(2) + \zeta(2) + S_{12}(m) + \zeta(3)}{m}$

Out[1268]= $\mathrm{Integrate2}\left[x^{(\mathrm{OPEm}-1)}\,\mathrm{Log}[1-x]\,\mathrm{Log}[x]\,\frac{\mathrm{Log}[1+x]}{1+x}, \{x, 0, 1\}\right]$

```
In[1269]:= 1/4 (-1)^m ζ(2) S_-1^2(m-1) + (-1)^m Log(2) S_-2(m-1) S_-1(m-1) - (-1)^m ζ(3) S_-1(m-1) +
           3/2 (-1)^m ζ(2) Log(2) S_-1(m-1) - 1/2 (-1)^m ζ(2) S_-2(m-1) + 1/2 (-1)^m Log^2(2) S_-2(m-1) -
           3/2 (-1)^m ζ(2) Log(2) S_1(m-1) + 3/4 (-1)^m ζ(2) S_2(m-1) - 1/2 (-1)^m Log^2(2) S_2(m-1) +
           (-1)^m Log(2) S_3(m-1) - (-1)^m Log(2) S_-21(m-1) - (-1)^m Log(2) S_-12(m-1) -
           (-1)^m ζ(2) S_1-1(m-1) + (-1)^m S_-2-1-1(m-1) + (-1)^m S_-1-2-1(m-1) + (-1)^m S_-1-1-2(m-1) +
           (-1)^m S_1-21(m-1) + (-1)^m S_1-12(m-1) + (-1)^m S_2-11(m-1) + 13/8 (-1)^m S_1(m-1) ζ(3) -
           21/8 (-1)^m Log(2) ζ(3) - 2 (-1)^m Li_4(1/2) - 1/12 (-1)^m Log^4(2) + 5/4 (-1)^m ζ(2) Log^2(2) + 1/45 (-1)^m π^4
Out[1269]= % /. OPEm → 2

In[1270]:= 5/4 Log^2(2) ζ(2) - 3 Log(2) ζ(2) + 5 ζ(2)/2 - 21/8 Log(2) ζ(3) +
           21 ζ(3)/8 - 2 Li_4(1/2) - Log^4(2)/12 - Log^2(2) + 4 Log(2) + π^4/45 - 6
Out[1270]= N[%]

In[1271]:= 0.0505138
Out[1271]= NIntegrate[x Log[1 - x] Log[x] Log[1 + x]/(1 + x), {x, 0, 1}]

In[1272]:= 0.0505138
Out[1272]= Integrate2[x^(OPEm - 1) (PolyLog[3, (1 - x)/(1 + x)] - PolyLog[3, -(1 - x)/(1 + x)]), {x, 0, 1}]

In[1273]:= (-1)^m S_-1(m) ζ(2)/m - S_-1(m) ζ(2)/(2 m) + (-1)^m S_1(m) ζ(2)/(2 m) - S_1(m) ζ(2)/m + 3 (-1)^m Log(2) ζ(2)/(2 m) -
           3 Log(2) ζ(2)/(2 m) + (-1)^m S_-3(m)/m + (-1)^m S_-2(m) S_1(m)/m + S_1(m) S_2(m)/m + S_3(m)/m -
           (-1)^m S_-21(m)/m - S_-1-2(m)/m - (-1)^m S_-12(m)/m - S_21(m)/m - 7 (-1)^m ζ(3)/(8 m) + 21 ζ(3)/(8 m)
Out[1273]= DataType[OPEm, PositiveInteger]
```

This is the polarized non-singlet spin splitting function whose first moment vanishes.

```
In[1274]:= True
Out[1274]= Integrate2[x^(OPEm - 1) DeltaFunction[1 - x], {x, 0, 1}]

In[1275]:= 1
Out[1275]= t = SplittingFunction[PQQNS]
```

Expanding t with respect to x yields a form already suitable for Integrate3 and therefore the following is faster:

```
In[1276]:= ( -4 (x + 1) Log^2(x) - 8 (2 x + 3/(1 - x)) Log(x) -
              16 (x^2 + 1) Log(1 - x) Log(x)/(1 - x) - 40 (1 - x) + δ(1 - x) (-24 ζ(2) + 48 ζ(3) + 3)) C_F^2 +
            N_f (88 x/9 + (-16 ζ(2)/3 - 2/3) δ(1 - x) - 8 (x^2 + 1) Log(x)/(3 (1 - x)) - 80/9 (1/(1 - x))_+ - 8/9) C_F -
            8 (C_F - C_A/2) (4 (1 - x) + 2 (x + 1) Log(x) + (x^2 + 1) (Log^2(x) - 4 Log(x + 1) Log(x) - 2 ζ(2) - 4 Li_2(-x))/(x + 1)
            C_A (4 (x^2 + 1) Log^2(x)/(1 - x) - 4/3 (5 x - 22/(1 - x) + 5) Log(x) + 4/9 (53 - 187 x) +
              8 (x + 1) ζ(2) + (536/9 - 16 ζ(2)) (1/(1 - x))_+ + δ(1 - x) (88 ζ(2)/3 - 24 ζ(3) + 17/3)) C_F

Out[1276]= Integrate2[t, {x, 0, 1}]//Timing

In[1277]:= {0.52 Second, 0}

In[1278]:= Integrate3[Expand[t, x], {x, 0, 1}]//Expand//Timing
Out[1278]= {0.11 Second, 0}

In[1279]:= Clear[t];
Out[1279]= Integrate2[DeltaFunction[1 - x] f[x], {x, 0, 1}]

In[1280]:= f(1)
Out[1280]= Integrate2[x^5 Log[1 + x]^2, {x, 0, 1}]

In[1281]:= -6959/10800 + 46 Log(2)/45
Out[1281]= N@%

In[1282]:= 0.0641986
Out[1282]= NIntegrate[x^5 Log[1 + x]^2, {x, 0, 1}]
```

Integrate3

Description

Integrate3 contains the integral table used by Integrate2. Integration is performed in a distributional sense. Integrate3 works more effectively on a sum of expressions if they are expanded or collected with respect to the integration variable. See the examples in Integrate2.

See also: Integrate2.

Examples

```
In[1283]:= 0.0641986
Out[1283]= Integrate2[x^(OPEm - 1) Log[1 + x]^2, {x, 0, 1}]
```

```
In[1284]:= -2 (-1)^m S_1^2(m)/m + (-1)^m S_1((m-1)/2) S_1(m)/m - S_1((m-1)/2) S_1(m)/m +
            (-1)^m S_1(m/2) S_1(m)/m + S_1(m/2) S_1(m)/m + 4 (-1)^m Log(2) S_1(m)/m - (-1)^m Log(2) S_1((m-1)/2)/m +
            Log(2) S_1((m-1)/2)/m - (-1)^m Log(2) S_1(m/2)/m - Log(2) S_1(m/2)/m + (-1)^m S_2((m-1)/2)/(2 m) - S_2((m-1)/2)/(2 m) +
            (-1)^m S_2(m/2)/(2 m) + S_2(m/2)/(2 m) - 2 (-1)^m S_2(m)/m - 2 (-1)^m S_-11(m)/m - (-1)^m Log^2(2)/m + Log^2(2)/m
Out[1284]= Integrate3[x^OPEm Log[x], {x, 0, 1}]

In[1285]:= -1/(m+1)^2
Out[1285]= Integrate3[x^OPEm Log[x] Log[1-x]/(1-x), {x, 0, 1}]
In[1286]:= ζ(2) S_1(m) - S_12(m) - S_21(m) + ζ(3)
Out[1286]= Integrate3[a x^OPEm Log[x] Log[1-x]/(1-x) + b x^OPEm PolyLog[3, -x]/(1+x), {x, 0, 1}]

In[1287]:= a (ζ(2) S_1(m) - S_12(m) - S_21(m) + ζ(3)) +
            (-1)^m b (ζ(2)^2/8 + 1/2 S_-2(m) ζ(2) + Log(2) (S_3(m) - S_-3(m)) + S_3-1(m) - 3/4 S_-1(m) ζ(3) - 3/4 Log(2) ζ(3))
Out[1287]= Integrate3[DeltaFunctionPrime[1-x], {x, 0, 1}]

In[1288]:= 0
Out[1288]= Integrate3[f[x] DeltaFunctionPrime[1-x], {x, 0, 1}]
```

IntermediateSubstitutions

Description

IntermediateSubstitutions is an option for OneLoop. All substitutions indicated hereby are done at an intermediate stage of the calculation.

See also: OneLoop.

InverseMellin

Description

InverseMellin[exp, y] performs the inverse Mellin transform of polynomials in OPEm. The inverse transforms are not calculated but a table-lookup is done. WARNING: do not "trust" the results for the inverse Mellin transform involving SumT's; there is an unresolved inconsistency here (related to $f'(1)$

See also: DeltaFunction, Integrate2, OPEm, SumS, SumT.

Examples

```
In[1289]:= Integrate3[1/(1 - x), {x, 0, 1}]
Out[1289]= 0

In[1290]:= (-1)^m).
Out[1290]= InverseMellin[1/OPEm, y]

In[1291]:= y^(m-1)
Out[1291]= InverseMellin[1/(OPEm + 3), y]

In[1292]:= y^(m+2)
Out[1292]= InverseMellin[1, y]

In[1293]:= y^(m-1) δ(1 - y)
Out[1293]= InverseMellin[1/OPEm^4, y]

In[1294]:= -1/6 y^(m-1) Log^3(y)
Out[1294]= InverseMellin[1/OPEm + 1, y]
```

The inverse operation to InverseMellin is done by Integrate2.

```
In[1295]:= δ(1 - y) y^(m-1) + y^(m-1)
Out[1295]= InverseMellin[1/i + 1, y, i]
```

Below is a list of all built-in basic inverse Mellin transforms .

```
In[1296]:= δ(1 - y) y^(i-1) + y^(i-1)

In[1297]:= Integrate2[InverseMellin[1/OPEm, y], {y, 0, 1}]

In[1298]:= 1/m

Out[1298]= list = {1, 1/(OPEm + n), 1/(-OPEm + n), PolyGamma[0, OPEm], SumS[1, -1 + OPEm],
     SumS[1, -1 + OPEm]/(OPEm - 1), SumS[1, -1 + OPEm]/(1 - OPEm), SumS[1, -1 + OPEm]/(OPEm + 1),
     SumS[1, -1 + OPEm]/OPEm^2, SumS[1, -1 + OPEm]/OPEm, SumS[1, -1 + OPEm]^2/OPEm, SumS[2, -1 + OPEm],
     SumS[2, -1 + OPEm]/OPEm, SumS[3, -1 + OPEm], SumS[1, 1, -1 + OPEm], SumS[1, OPEm - 1]^2,
     SumS[1, 2, -1 + OPEm], SumS[2, 1, -1 + OPEm], SumS[1, -1 + OPEm]^3,
     SumS[1, -1 + OPEm] SumS[2, -1 + OPEm], SumS[1, 1, 1, -1 + OPEm]};

In[1299]:= im[z_] := z → InverseMellin[z, y]
Out[1299]= im[OPEm^(-3)]

In[1300]:= 1/m^3 → (1/2 y^(m-1) Log^2(y))
Out[1300]= im[OPEm^(-2)]

In[1301]:= 1/m^2 → (- y^(m-1) Log(y))
```

In[1302]:= **im[PolyGamma[0, OPEm]]**

Out[1302]= $\psi^{(0)}(m) \longrightarrow \left(-\Gamma_E\, \delta(1-y)\, y^{m-1} - \left(\frac{1}{1-y}\right)_+ y^{m-1}\right)$

In[1303]:=

Out[1303]= im[SumS[1, OPEm - 1]]

In[1304]:= $\mathbf{S_1(m-1) \longrightarrow \left(-y^{m-1}\left(\frac{1}{1-y}\right)_+\right)}$

Out[1304]= im[SumS[1, OPEm - 1]/(OPEm - 1)]

In[1305]:= $\mathbf{\frac{S_1(m-1)}{m-1} \longrightarrow (-y^{m-2}\,Log(1-y))}$

Out[1305]= im[SumS[1, OPEm - 1]/(OPEm + 1)]

In[1306]:= $\mathbf{\frac{S_1(m-1)}{m+1} \longrightarrow \left(-y^{m-1} - Log(1-y)\,y^m + Log(y)\,y^m + y^m\right)}$

Out[1306]= im[SumS[1, -1 + OPEm]/OPEm^2]

In[1307]:= $\mathbf{\frac{S_1(m-1)}{m^2} \longrightarrow \left(y^{m-1}\left(-\frac{1}{2}Log^2(y) + \zeta(2) - Li_2(y)\right)\right)}$

Out[1307]= im[SumS[1, -1 + OPEm]/OPEm]

In[1308]:= $\mathbf{\frac{S_1(m-1)}{m} \longrightarrow \left(y^{m-1}\,(Log(y) - Log(1-y))\right)}$

Out[1308]= im[SumS[1, -1 + OPEm]^2/OPEm]

In[1309]:= $\mathbf{\frac{S_1^2(m-1)}{m} \longrightarrow \left(y^{m-1}\left(Log^2(1-y) + \frac{Log^2(y)}{2} - 3\zeta(2) + Li_2(1-y) + 2\,Li_2(y)\right)\right)}$

Out[1309]= im[SumS[2, OPEm - 1]]

In[1310]:= $\mathbf{S_2(m-1) \longrightarrow \left(y^{m-1}\left(\zeta(2)\,\delta(1-y) + \frac{Log(y)}{1-y}\right)\right)}$

Out[1310]= im[SumS[2, OPEm - 1]/OPEm]

In[1311]:= $\mathbf{\frac{S_2(m-1)}{m} \longrightarrow \left(y^{m-1}\left(-\frac{1}{2}Log^2(y) + \zeta(2) - Li_2(1-y)\right)\right)}$

Out[1311]= im[SumS[3, OPEm - 1]]

In[1312]:= $\mathbf{S_3(m-1) \longrightarrow \left(y^{m-1}\left(\delta(1-y)\,\zeta(3) - \frac{Log^2(y)}{2\,(1-y)}\right)\right)}$

Out[1312]= im[SumS[1, 1, OPEm - 1]]

In[1313]:= $\mathbf{S_{11}(m-1) \longrightarrow \left(y^{m-1}\left(\frac{Log(1-y)}{1-y}\right)_+\right)}$

Out[1313]= im[SumS[1, 2, OPEm - 1]]

In[1314]:= $\mathbf{S_{12}(m-1) \longrightarrow \left(y^{m-1}\left(-\zeta(3)\,\delta(1-y) - \zeta(2)\left(\frac{1}{1-y}\right)_+ + \frac{Li_2(1-y)}{1-y}\right)\right)}$

Out[1314]= im[SumS[2, 1, OPEm - 1]]

In[1315]:= $\mathbf{S_{21}(m-1) \longrightarrow \left(y^{m-1}\left(2\zeta(3)\,\delta(1-y) - \zeta(2)\left(\frac{1}{1-y}\right)_+ + \frac{Li_2(y)}{1-y}\right)\right)}$

Isolate

Description

Isolate[expr] substitutes abbreviations KK[i] for all Plus[...] (sub-sums) in expr. The inserted KK[i] have head HoldForm. Isolate[expr, varlist] substitutes KK[i] for all subsums in expr which are free of any occurence of a member of the list varlist. Instead of KK any other head or a list of names of the abbreviations may be specified with the option IsolateNames.

```
In[1316]:= im[SumS[1, 1, 1, OPEm - 1]]
```

$$\text{Out[1316]= } S_{111}(m-1) \longrightarrow \left(-\frac{1}{2}\, y^{m-1} \left(\frac{\text{Log}^2(1-y)}{1-y} \right)_+ \right)$$

Examples

```
In[1317]:= Clear[im, list];
Out[1317]= Options[Isolate]

In[1318]:= {IsolateNames → KK, HighEnergyPhysics`FeynCalc`IsolatePrint`IsolatePrint → False,
             HighEnergyPhysics`FeynCalc`IsolateSplit`IsolateSplit → ∞}
Out[1318]= t0 = Isolate[a + b]

In[1319]:= KK(1)
Out[1319]= t1 = Isolate[(a + b) f + (c + d) f + e, f]

In[1320]:= e + f KK(1) + f KK(2)
Out[1320]= StandardForm[t1]

In[1321]:= e + f KK[1] + f KK[2]
Out[1321]= {t0, t1, ReleaseHold[t1]}

In[1322]:= {KK(1), e + f KK(1) + f KK(2), e + (a + b) f + (c + d) f}
Isolate[a[z] (b + c (y + z)) + d[z] (y + z), {a, d}, IsolateNames → F]
d(z) F(3) + a(z) F(4)

In[1323]:= ??F
F is a fermion field.
  F[3] = y + z


  F[4] = b + c HoldForm[F[3]]
Out[1323]= Isolate[a - b - c - d - e, IsolateNames → L, IsolateSplit → 15]

In[1324]:= L2 :: shdw : Symbol L2 appears in multiple contexts {Global`, HighEnergyPhysics`Phi`Objects`};
             definitions in context Global` may shadow or be shadowed by other definitions.
```

```
Out[1324]= L3 :: shdw : Symbol L3 appears in multiple contexts {Global`, HighEnergyPhysics`Phi`Objects`};
             definitions in context Global` may shadow or be shadowed by other definitions.

In[1325]:= L(3)
```

IsolateHead

Description

IsolateHead is equivalent to IsolateNames.
See also: IsolateNames.

IsolateNames

Description

IsolateNames is an option for Isolate and Collect2. Its default setting is KK. Instead of a symbol the setting may also be a list with the names of the abbrevations.
See also: Isolate, Collect2.

IsolatePrint

Description

IsolatePrint is an option of Isolate. If it is set to OutputForm (or any other *Form) the definitions of the abbreviations are printed during the operation of Isolate. The setting IsolatePrint → False suppresses printing.
See also: Isolate.

IsolateSplit

Description

IsolateSplit is an option for Isolate. Its setting determines the maximum number of characters of FortranForm[expr] which are abbreviated by Isolate. If the expression is larger than the indicated number, it is split into smaller pieces and onto each subsum Isolate is applied. With the default setting IsolateSplit → Infinity no splitting is done.
See also: Isolate.

KeepOnly

Description

KeepOnly is an option of OneLoop. It may be set to B0, C0, D0 keeping only the corresponding coefficients. The default setting is False. If KeepOnly is set to { } then the part of the amplitude which is not coefficient of B0, C0, D0 is kept.

See also: OneLoop, B0, C0, D0.

KK

Description

KK[i] is the default setting of IsolateNames, which is the head of abbreviations used by Isolate. A KK[i] returned by Isolate is given in HoldForm and can be recovered by ReleaseHold[KK[i]].

See also: Isolate, IsolateNames.

Kummer

Description

Kummer[i][exp] applies Kummer relation number i (i =1, ... 24, 94,95,96) to all Hypergeometric2F1 in exp. i = 94 corresponds to eq. 9.131.2, i = 95 to eq. 9.132.1 and i = 96 to eq. 9.132.2 in Gradsteyn & Ryzhik.

See also: HypergeometricAC.

Examples

```
In[1326]:= {L[2], L[1]}
Out[1326]= {a - b, L(1)}

In[1327]:= Clear[t0, t1, L]
Out[1327]= Hypergeometric2F1[a, b, c, z] == Kummer[2][Hypergeometric2F1[a, b, c, z]]

In[1328]:= 2F1(a, b; c; z) == (1 - z)^(-a-b+c) 2F1(c - a, c - b; c; z)
Out[1328]= Hypergeometric2F1[a, b, c, z] == Kummer[3][Hypergeometric2F1[a, b, c, z]]

In[1329]:= 2F1(a, b; c; z) == (1 - z)^(-a) 2F1(a, c - b; c; -z/(1 - z))
Out[1329]= Hypergeometric2F1[a, b, c, z] == Kummer[4][Hypergeometric2F1[a, b, c, z]]

In[1330]:= 2F1(a, b; c; z) == (1 - z)^(-b) 2F1(c - a, b; c; -z/(1 - z))
Out[1330]= Hypergeometric2F1[a, b, c, 1 - z] == Kummer[6][Hypergeometric2F1[a, b, c, 1 - z]]

In[1331]:= 2F1(a, b; c; 1 - z) == z^(-a-b+c) 2F1(c - b, c - a; c; 1 - z)
```

```
Out[1331]= Hypergeometric2F1[a, b, a + b + 1 - c, 1 - z] ==
             Kummer[6][Hypergeometric2F1[a, b, a + b + 1 - c, 1 - z]]
```

$$In[1332]:= {}_2F_1(a,b;a+b-c+1;1-z) == z^{1-c}\,{}_2F_1(a-c+1,b-c+1;a+b-c+1;1-z)$$

```
Out[1332]= Hypergeometric2F1[a, b, c, 1 - z] == Kummer[7][Hypergeometric2F1[a, b, c, 1 - z]]
```

$$In[1333]:= {}_2F_1(a,b;c;1-z) == z^{-a}\,{}_2F_1\left(a,c-b;c;-\frac{1-z}{z}\right)$$

```
Out[1333]= Hypergeometric2F1[a, b, a + b + 1 - c, 1 - z] ==
             Kummer[7][Hypergeometric2F1[a, b, a + b + 1 - c, 1 - z]]
```

$$In[1334]:= {}_2F_1(a,b;a+b-c+1;1-z) == z^{-a}\,{}_2F_1\left(a,a-c+1;a+b-c+1;-\frac{1-z}{z}\right)$$

```
Out[1334]= Hypergeometric2F1[a, b, c, 1 - z] == Kummer[8][Hypergeometric2F1[a, b, c, 1 - z]]
```

$$In[1335]:= {}_2F_1(a,b;c;1-z) == z^{-b}\,{}_2F_1\left(c-a,b;c;-\frac{1-z}{z}\right)$$

```
Out[1335]= Hypergeometric2F1[a, b, a + b + 1 - c, 1 - z] ==
             Kummer[8][Hypergeometric2F1[a, b, a + b + 1 - c, 1 - z]]
```

$$In[1336]:= {}_2F_1(a,b;a+b-c+1;1-z) == z^{-b}\,{}_2F_1\left(b-c+1,b;a+b-c+1;-\frac{1-z}{z}\right)$$

```
Out[1336]= Hypergeometric2F1[a, b, c, z^(-1)] == Kummer[10][Hypergeometric2F1[a, b, c, z^(-1)]]
```

$$In[1337]:= {}_2F_1\left(a,b;c;\frac{1}{z}\right) == (1-z)^{-a-b+c}\,(-z)^{a+b-c}\,{}_2F_1\left(c-a,c-b;c;\frac{1}{z}\right)$$

```
Out[1337]= Hypergeometric2F1[a, a + 1 - c, a + 1 - b, z^(-1)] ==
             Kummer[10][Hypergeometric2F1[a, a + 1 - c, a + 1 - b, z^(-1)]]
```

$$In[1338]:= {}_2F_1\left(a,a-c+1;a-b+1;\frac{1}{z}\right) == (1-z)^{-a-b+c}\,(-z)^{a+b-c}\,{}_2F_1\left(1-b,c-b;a-b+1;\frac{1}{z}\right)$$

```
Out[1338]= Hypergeometric2F1[a, b, c, z^(-1)] == Kummer[11][Hypergeometric2F1[a, b, c, z^(-1)]]
```

$$In[1339]:= {}_2F_1\left(a,b;c;\frac{1}{z}\right) == \left(-\frac{1-z}{z}\right)^{-a}\,(-z)^{a}\,{}_2F_1\left(a,c-b;c;\frac{1}{1-z}\right)$$

```
Out[1339]= Hypergeometric2F1[a, a + 1 - c, a + 1 - b, z^(-1)] ==
             Kummer[11][Hypergeometric2F1[a, a + 1 - c, a + 1 - b, z^(-1)]]
```

$$In[1340]:= {}_2F_1\left(a,a-c+1;a-b+1;\frac{1}{z}\right) == \left(-\frac{1-z}{z}\right)^{-a}\,(-z)^{a}\,{}_2F_1\left(a,c-b;a-b+1;\frac{1}{1-z}\right)$$

```
Out[1340]= Hypergeometric2F1[a, b, c, z^(-1)] == Kummer[12][Hypergeometric2F1[a, b, c, z^(-1)]]
```

$$In[1341]:= {}_2F_1\left(a,b;c;\frac{1}{z}\right) == (1-z)^{-b}\,(-z)^{b-a}\,{}_2F_1\left(b,c-a;c;\frac{1}{1-z}\right)$$

```
Out[1341]= Hypergeometric2F1[a, a + 1 - c, a + 1 - b, z^(-1)] ==
             Kummer[12][Hypergeometric2F1[a, a + 1 - c, a + 1 - b, z^(-1)]]
```

$$In[1342]:= {}_2F_1\left(a,a-c+1;a-b+1;\frac{1}{z}\right) == (1-z)^{-a+c-1}\,(-z)^{1-c}\,{}_2F_1\left(a-c+1,1-b;a-b+1;\frac{1}{1-z}\right)$$

```
Out[1342]= Hypergeometric2F1[a, b, c, z^(-1)] == Kummer[14][Hypergeometric2F1[a, b, c, z^(-1)]]
```

In[1343]:= $${}_2F_1\left(a, b; c; \frac{1}{z}\right) == (1-z)^{-a-b+c}\,(-z)^{a+b-c}\,{}_2F_1\left(c-b, c-a; c; \frac{1}{z}\right)$$

Out[1343]= Hypergeometric2F1[b + 1 - c, b, b + 1 - a, z^(-1)] ==

Kummer[14][Hypergeometric2F1[b + 1 - c, b, b + 1 - a, z^(-1)]]

In[1344]:= $${}_2F_1\left(b-c+1, b; -a+b+1; \frac{1}{z}\right) == (1-z)^{-a-b+c}\,(-z)^{a+b-c}\,{}_2F_1\left(1-a, c-a; -a+b+1; \frac{1}{z}\right)$$

Out[1344]= Hypergeometric2F1[a, b, c, z^(-1)] == Kummer[15][Hypergeometric2F1[a, b, c, z^(-1)]]

In[1345]:= $${}_2F_1\left(a, b; c; \frac{1}{z}\right) == (1-z)^{-b}\,(-z)^{b}\,{}_2F_1\left(b, c-a; c; \frac{1}{1-z}\right)$$

Out[1345]= Hypergeometric2F1[b + 1 - c, b, b + 1 - a, z^(-1)] ==

Kummer[15][Hypergeometric2F1[b + 1 - c, b, b + 1 - a, z^(-1)]]

In[1346]:= $${}_2F_1\left(b-c+1, b; -a+b+1; \frac{1}{z}\right) == (1-z)^{-b}\,(-z)^{b}\,{}_2F_1\left(b, c-a; -a+b+1; \frac{1}{1-z}\right)$$

Out[1346]= Hypergeometric2F1[a, b, c, z^(-1)] == Kummer[16][Hypergeometric2F1[a, b, c, z^(-1)]]

In[1347]:= $${}_2F_1\left(a, b; c; \frac{1}{z}\right) == (1-z)^{-a}\,(-z)^{a}\,{}_2F_1\left(a, c-b; c; \frac{1}{1-z}\right)$$

Out[1347]= Hypergeometric2F1[b + 1 - c, b, b + 1 - a, z^(-1)] ==

Kummer[16][Hypergeometric2F1[b + 1 - c, b, b + 1 - a, z^(-1)]]

In[1348]:= $${}_2F_1\left(b-c+1, b; -a+b+1; \frac{1}{z}\right) == (1-z)^{-b+c-1}\,(-z)^{b-c+1}\,{}_2F_1\left(b-c+1, 1-a; -a+b+1; \frac{1}{1-z}\right)$$

Out[1348]= Hypergeometric2F1[a + 1 - c, b + 1 - c, 2 - c, z] ==

Kummer[18][Hypergeometric2F1[a + 1 - c, b + 1 - c, 2 - c, z]]

In[1349]:= $${}_2F_1(a-c+1, b-c+1; 2-c; z) == (1-z)^{-a-b+c}\,{}_2F_1(1-a, 1-b; 2-c; z)$$

Out[1349]= Hypergeometric2F1[a, b, c, z] == Kummer[18][Hypergeometric2F1[a, b, c, z]]

In[1350]:= $${}_2F_1(a, b; c; z) == (1-z)^{-a-b+c}\,{}_2F_1(c-a, c-b; c; z)$$

Out[1350]= Hypergeometric2F1[a + 1 - c, b + 1 - c, 2 - c, z] ==

Kummer[19][Hypergeometric2F1[a + 1 - c, b + 1 - c, 2 - c, z]]

In[1351]:= $${}_2F_1(a-c+1, b-c+1; 2-c; z) == (1-z)^{-a+c-1}\,{}_2F_1\left(a-c+1, 1-b; 2-c; \frac{z}{z-1}\right)$$

Out[1351]= Hypergeometric2F1[a, b, c, z] == Kummer[19][Hypergeometric2F1[a, b, c, z]]

In[1352]:= $${}_2F_1(a, b; c; z) == (1-z)^{-a}\,{}_2F_1\left(a, c-b; c; \frac{z}{z-1}\right)$$

Out[1352]= Hypergeometric2F1[a + 1 - c, b + 1 - c, 2 - c, z] ==

Kummer[20][Hypergeometric2F1[a + 1 - c, b + 1 - c, 2 - c, z]]

In[1353]:= $${}_2F_1(a-c+1, b-c+1; 2-c; z) == (1-z)^{-b+c-1}\,{}_2F_1\left(b-c+1, 1-a; 2-c; \frac{z}{z-1}\right)$$

Out[1353]= Hypergeometric2F1[a, b, c, z] == Kummer[20][Hypergeometric2F1[a, b, c, z]]

In[1354]:= $${}_2F_1(a, b; c; z) == (1-z)^{-b}\,{}_2F_1\left(b, c-a; c; \frac{z}{z-1}\right)$$

```
Out[1354]= Hypergeometric2F1[c - a, c - b, c + 1 - a - b, 1 - z] ==

             Kummer[22][Hypergeometric2F1[c - a, c - b, c + 1 - a - b, 1 - z]]
```

$$In[1355]:= {}_2F_1(c-a, c-b; -a-b+c+1; 1-z) == z^{1-c}\, {}_2F_1(1-a, 1-b; -a-b+c+1; 1-z)$$

```
Out[1355]= Hypergeometric2F1[a, b, c, 1 - z] == Kummer[22][Hypergeometric2F1[a, b, c, 1 - z]]
```

$$In[1356]:= {}_2F_1(a, b; c; 1-z) == z^{-a-b+c}\, {}_2F_1(c-b, c-a; c; 1-z)$$

```
Out[1356]= Hypergeometric2F1[c - a, c - b, c + 1 - a - b, 1 - z] ==

             Kummer[23][Hypergeometric2F1[c - a, c - b, c + 1 - a - b, 1 - z]]
```

$$In[1357]:= {}_2F_1(c-a, c-b; -a-b+c+1; 1-z) == (1-z)^{a-c}\, {}_2F_1\left(c-a, 1-a; -a-b+c+1; 1-\frac{1}{1-z}\right)$$

```
Out[1357]= Hypergeometric2F1[a, b, c, 1 - z] == Kummer[23][Hypergeometric2F1[a, b, c, 1 - z]]
```

$$In[1358]:= {}_2F_1(a, b; c; 1-z) == (1-z)^{-a}\, {}_2F_1\left(a, c-b; c; 1-\frac{1}{1-z}\right)$$

```
Out[1358]= Hypergeometric2F1[c - a, c - b, c + 1 - a - b, 1 - z] ==

             Kummer[24][Hypergeometric2F1[c - a, c - b, c + 1 - a - b, 1 - z]]
```

$$In[1359]:= {}_2F_1(c-a, c-b; -a-b+c+1; 1-z) == z^{b-c}\, {}_2F_1\left(c-b, 1-b; -a-b+c+1; -\frac{1-z}{z}\right)$$

```
Out[1359]= Hypergeometric2F1[a, b, c, 1 - z] == Kummer[24][Hypergeometric2F1[a, b, c, 1 - z]]
```

$$In[1360]:= {}_2F_1(a, b; c; 1-z) == z^{-b}\, {}_2F_1\left(b, c-a; c; -\frac{1-z}{z}\right)$$

```
Out[1360]= Hypergeometric2F1[a, b, c, z] == Kummer[94][Hypergeometric2F1[a, b, c, z]]
```

$$In[1361]:= {}_2F_1(a, b; c; z) == \frac{\Gamma(a+b-c)\,\Gamma(c)\,{}_2F_1(c-a, c-b; -a-b+c+1; 1-z)\,(1-z)^{-a-b+c}}{\Gamma(a)\,\Gamma(b)} + \frac{\Gamma(c)\,\Gamma(-a-b+c)\,{}_2F_1(a, b; a+b-c+1; 1-z)}{\Gamma(c-a)\,\Gamma(c-b)}$$

```
Out[1361]= Hypergeometric2F1[a, b, c, z] == Kummer[95][Hypergeometric2F1[a, b, c, z]]
```

Lagrangian

Description

Lagrangian["oqu"] gives the unpolarized OPE quark operator. Lagrangian["oqp"] gives the polarized quark OPE operator. Lagrangian["ogu"] gives the unpolarized gluon OPE operator. Lagrangian["ogp"] gives the polarized gluon OPE operator. Lagrangian["ogd"] gives the sigma-term part of the QCD lagrangian. Lagrangian["QCD"] gives the gluon self interaction part of the QCD lagrangian.

See also: FeynRule.

Examples

```
In[1362]:= ₂F₁(a, b; c; z) ==
           Γ(b - a) Γ(c) ₂F₁(a, c - b; a - b + 1; 1/(1-z)) (1 - z)^-a / (Γ(b) Γ(c - a)) + Γ(a - b) Γ(c) ₂F₁(b, c - a; -a + b + 1; 1/(1-z)) (1 - z)^-b / (Γ(a) Γ(c - b))
Out[1362]= Hypergeometric2F1[a, b, c, z] == Kummer[96][Hypergeometric2F1[a, b, c, z]]
```

Twist-2 operator product expansion operators

```
In[1363]:= ₂F₁(a, b; c; z) ==
           (-1)^a Γ(b - a) Γ(c) ₂F₁(a, a - c + 1; a - b + 1; 1/z) z^-a / (Γ(b) Γ(c - a)) + (-1)^b Γ(a - b) Γ(c) ₂F₁(b, b - c + 1; -a + b + 1; 1/z)
Out[1363]= Lagrangian["QCD"]

In[1364]:= -1/4 F^a_αβ . F^a_αβ
Out[1364]= Lagrangian["ogu"]

In[1365]:= 1/2 i^(m-1) F^a_αΔ . (D^ab_Δ)^(m-2) . F^b_αΔ
Out[1365]= Lagrangian["ogp"]

In[1366]:= 1/2 i^m ϵ^αβγΔ . F^a_βγ . (D^ab_Δ)^(m-2) . F^b_αΔ
Out[1366]= Lagrangian["oqu"]
```

LC

Description

LC[m,n,r,s] evaluates to 4-dimensional LeviCivita[m,n,r,s] by virtue of applying FeynCalcInternal. LC[m,...][p, ...] evaluates to 4-dimensional LeviCivita[m,...][p,...] applying FeynCalcInternal.

See also: LeviCivita, LCD.

Examples

```
In[1367]:= i^m ψ̄ . (γ · Δ) . D_Δ^(m-1) . ψ
Out[1367]= Lagrangian["oqp"]

In[1368]:= i^m ψ̄ . γ^5 . (γ · Δ) . D_Δ^(m-1) . ψ
Out[1368]= LC[μ, ν, ρ, σ]

In[1369]:= ϵ^μνρσ
Out[1369]= %//FCI

In[1370]:= ϵ^μνρσ
Out[1370]= %//StandardForm

In[1371]:= Eps[LorentzIndex[μ], LorentzIndex[ν], LorentzIndex[ρ], LorentzIndex[σ]]
```

```
Out[1371]= LC[μ, ν][p, q]

In[1372]:= ϵ^μνpq
Out[1372]= %//FCI//StandardForm
```

LCD

Description

LCD[m,n,r,s] evaluates to D-dimensional LeviCivita[m,n,r,s] by virtue of FeynCalcInternal. LCD[m,...][p, ...] evaluates to D-dimensional LeviCivita[m,...][p,...] applying FeynCalcInternal.

You need to do SetOptions[Eps, Dimension→D] before LCD can be used as D-dimensional Levi-Civita input function.

See also: LeviCivita, LC.

Examples

```
In[1373]:= Eps[LorentzIndex[μ], LorentzIndex[ν], Momentum[p, D], Momentum[q, D]]

In[1374]:= Contract[LC[μ, ν, ρ][p] LC[μ, ν, ρ][q]]
Out[1374]= 18 p · q - 24 p · q

In[1375]:= SetOptions[Eps, Dimension → D];
Out[1375]= LCD[μ, ν, ρ, σ]

In[1376]:= ϵ^μνρσ
Out[1376]= %//FCI

In[1377]:= ϵ^μνρσ
Out[1377]= %//StandardForm

In[1378]:= Eps[LorentzIndex[μ, D], LorentzIndex[ν, D], LorentzIndex[ρ, D], LorentzIndex[σ, D]]
Out[1378]= LCD[μ, ν][p, q]

In[1379]:= ϵ^μνpq
Out[1379]= %//FCI//StandardForm

In[1380]:= Eps[LorentzIndex[μ, D], LorentzIndex[ν, D], Momentum[p, D], Momentum[q, D]]
Out[1380]= Factor2[Contract[LCD[μ, ν, ρ][p] LCD[μ, ν, ρ][q]]]
```

LeftPartialD

Description

LeftPartialD[μ] denotes $(1 - D)\,(2 - D)\,(3 - D)\,p \cdot q$acting to the left.

See also: ExpandPartialD, PartialD, LeftRightPartialD, RightPartialD.

SetOptions[Eps, Dimension → 4]

```
In[1381]:= {Dimension → 4}
Out[1381]= (←∂)_μ

In[1382]:= Examples
Out[1382]= QuantumField[A, LorentzIndex[μ]].LeftPartialD[ν]

In[1383]:= A_μ . (←∂)_ν
Out[1383]= ExpandPartialD[%]

In[1384]:= ∂_ν A_μ
Out[1384]= StandardForm[%]

In[1385]:= QuantumField[PartialD[LorentzIndex[ν]], A, LorentzIndex[μ]]
Out[1385]= StandardForm[LeftPartialD[μ]]

In[1386]:= LeftPartialD[LorentzIndex[μ]]
Out[1386]= QuantumField[A, LorentzIndex[μ]].QuantumField[A, LorentzIndex[ν]].LeftPartialD[ρ]

In[1387]:= A_μ . A_ν . (←∂)_ρ
Out[1387]= ExpandPartialD[%]
```

LeftRightPartialD

Description

LeftRightPartialD[mu] denotes $A_\mu.\partial_\rho A_\nu+\partial_\rho A_\mu.A_\nu$, acting to the left and right. ExplicitPartialD[LeftRightPartialD[StandardF

QuantumField[A, LorentzIndex[μ]].

gives 1/2 (RightPartialD[QuantumField[PartialD[LorentzIndex[ρ]], A, LorentzIndex[ν]]+ QuantumField[PartialD[LorentzIndex[ρ]], A, LorentzIndex[μ]]. QuantumField[A, LorentzIndex[ν]]] - LeftPartialD[$\left(\overleftrightarrow{\partial}\right)_\mu$]).

See also: ExplicitPartialD, ExpandPartialD, PartialD, LeftPartialD, LeftRightPartialD2, RightPartialD.

Examples

```
In[1388]:= μ
Out[1388]= μ

In[1389]:= μ
Out[1389]= LeftRightPartialD[μ]

In[1390]:= (↔∂)_μ
```

```
Out[1390]= ExplicitPartialD[%]

In[1391]:= 1/2 ((→∂)_μ - (←∂)_μ)
Out[1391]= LeftRightPartialD[μ].QuantumField[A, LorentzIndex[ν]]

In[1392]:= (⇆∂)_μ.A_ν
Out[1392]= ExpandPartialD[%]

In[1393]:= ∂_μA_ν/2 - 1/2 (←∂)_μ.A_ν
Out[1393]= QuantumField[A, LorentzIndex[μ]].
             LeftRightPartialD[ν].QuantumField[A, LorentzIndex[ρ]]
```

LeftRightPartialD2

Description

LeftRightPartialD2[mu] denotes $A_\mu.\left(\overleftrightarrow{\partial}\right)_\nu.A_\rho$, acting to the left and right. ExplicitPartialD[LeftRightPartialD2[ExpandPartial

gives (RightPartialD[$\frac{1}{2}$ $A_\mu.\partial_\nu A_\rho - \frac{1}{2}\,\partial_\nu A_\mu.A_\rho$] + LeftPartialD[$\left(\overleftrightarrow{\partial}\right)_\mu$]).

See also: ExplicitPartialD, ExpandPartialD, PartialD, LeftPartialD, RightPartialD.

Examples

```
In[1394]:= μ
Out[1394]= μ

In[1395]:= μ
Out[1395]= LeftRightPartialD2[μ]

In[1396]:= (⇆∂)_μ
Out[1396]= ExplicitPartialD[%]

In[1397]:= (←∂)_μ + (→∂)_μ
Out[1397]= LeftRightPartialD2[μ].QuantumField[A, LorentzIndex[ν]]

In[1398]:= (⇆∂)_μ.A_ν
Out[1398]= ExpandPartialD[%]

In[1399]:= (←∂)_μ.A_ν + ∂_μA_ν
Out[1399]= QuantumField[A, LorentzIndex[μ]].
             LeftRightPartialD2[ν].QuantumField[A, LorentzIndex[ρ]]
```

LeviCivita

Description

LeviCivita[$A_\mu.\left(\overleftrightarrow{\partial}\right)_\nu.A_\rho$] is an input function for the totally antisymmetric Levi-Civita tensor. It evaluates automatically to the internal representation Eps[LorentzIndex[ExpandPartialD[%]], LorentzIndex[$A_\mu.\partial_\nu A_\rho + \partial_\nu A_\mu.A_\rho$], LorentzIndex[$\mu$, ν, ρ, σ], LorentzIndex[σ]] (or with a second argument in LorentzIndex for the Dimension, if the option Dimension of LeviCivita is changed). LeviCivita[μ][ν.] evaluates to Eps[LorentzIndex[ρ], LorentzIndex[μ, ν ...], ..., Momentum[p, ..], ...].

```
In[1400]:= μ
Out[1400]= ν
```

See also: Eps, Contract, LC, LCD.

Examples

```
In[1401]:= p
Out[1401]= Options[LeviCivita]

In[1402]:= {Dimension → D}
Out[1402]= LeviCivita[α, β, γ, δ]

In[1403]:= ϵ^αβγδ
Out[1403]= LeviCivita[μ, ν][OPEDelta, p]

In[1404]:= ϵ^μνΔp
Out[1404]= LeviCivita[][p, q, r, s]

In[1405]:= ϵ^pqrs
Out[1405]= LeviCivita[α, β, β, δ]

In[1406]:= 0
Out[1406]= LeviCivita[μ, ν][OPEDelta, p]//StandardForm

In[1407]:= Eps[LorentzIndex[μ], LorentzIndex[ν], Momentum[OPEDelta], Momentum[p]]
Out[1407]= t1 = LeviCivita[α, β, γ, ρ].LeviCivita[α, β, γ, ρ]

In[1408]:= ϵ^αβγρ.ϵ^αβγρ

In[1409]:= Contract[t1]
Out[1409]= -24

In[1410]:= SetOptions[LeviCivita, Dimension → D]; SetOptions[Eps, Dimension → D];

In[1411]:= Contract[ LeviCivita[α, β, γ, ρ].LeviCivita[α, β, γ, ρ]]//Factor2
Out[1411]= (1 - D) (2 - D) (3 - D) D

In[1412]:= SetOptions[LeviCivita, Dimension → 4]; SetOptions[Eps, Dimension → 4];
```

LeviCivitaSign

Description

LeviCivitaSign is an option for DiracTrace and EpsChisholm. It determines the sign in the result of a Dirac trace of four gamma matrices and gamma5.

See also: DiracTrace, EpsChisholm.

ListIntegrals ***unfinished***

Description

ListIntegrals[exp, {q1, q2}] gives a list of basic integrals (FeynAmpDenominator's multiplied by scalar products involving q1, q2). Any non-integer exponent (e.g. the (OPEm-1) in ScalarProduct[OPEDelta,q1]^(OPEm-1)) is replaced by OPEm. If the options Pair is set to True only integrals involving scalar products (not counting those with OPEDelta) are given; with Pair → False those are excluded.

Examples

```
In[1413]:= LC[α, β, γ, δ]
```

Li2

Description

Li2 is an abbreviation for the dilog function, i.e., Li2 = PolyLog[2, #]&.

See also: Li3.

Examples

```
In[1414]:= ε^(αβγδ)
Out[1414]= Clear[t1]

In[1415]:= {}

In[1416]:= Li2[x]
Out[1416]= Li2(x)
```

Li3

Description

Li3 is an abbreviation for the trilog function, i.e., Li3 = PolyLog[3, #]&.
See also: Li2.

Examples

```
In[1417]:= Li2//StandardForm
Out[1417]= PolyLog[2, #1]&

In[1418]:= -∫ Log[1 - x]/x dx
Out[1418]= Li2(x)

In[1419]:= Li3[x]
Out[1419]= Li3(x)

In[1420]:= Li3//StandardForm
Out[1420]= PolyLog[3, #1]&
```

Loop

Description

Loop is an option indicating the number of (virtual) loops.

LorentzIndex

Description

LorentzIndex[mu] denotes a four dimensional Lorentz index. For other than four dimensions: LorentzIndex[mu, D] or LorentzIndex[mu], etc. LorentzIndex[mu, 4] simplifies to LorentzIndex[mu].
See also: ChangeDimension, Momentum.

Examples

This denotes a four-dimensional Lorentz index.

```
In[1421]:= D[Li3[x], x]
Out[1421]= Li2(x)/x
```

An optional second argument can be given for a dimension different from 4.

```
In[1422]:= ∫ Li2[x]/x dx
Out[1422]= Li3(x)

In[1423]:= LorentzIndex[α]
Out[1423]= α

In[1424]:= LorentzIndex[α, n]
Out[1424]= α

In[1425]:= LorentzIndex[ComplexIndex[α]]
Out[1425]= α*
```

Setting $LorentzIndices=True displays the dimension (if different from 4) as a subindex.

```
In[1426]:= ComplexConjugate[LorentzIndex[μ]]

In[1427]:= μ
Out[1427]= ComplexConjugate[LorentzIndex[ComplexIndex[β]]]

In[1428]:= β*
```

Lower

Description

Lower may be used inside LorentzIndex to indicate an covariant LorentzIndex.

See also: LorentzIndex, Upper, Contract1.

MakeContext

Description

MakeContext[string] constructs the context path of string. MakeContext is invoked at startup of FeynCalc. MakeContext[a, b] construct the context path of b defined in the context of a.

See also: FeynCalc, CheckContext, $FeynCalcStuff, Load.

Mandelstam

Description

Mandelstam is an option for DiracTrace, OneLoop, OneLoopSum, Tr and TrickMandelstam. A typical setting is Mandelstam → {s, t, u, $*LorentzIndices* = True; + {LorentzIndex[α, *D*], LorentzIndex[ν, *n*], LorentzIndex[β, 4]} + {α_D, ν_n, β} + $*LorentzIndices* = False;}, which stands for $s + t + u = m_1^2 + m_2^2 + m_3^2 + m_4^2$. If other than four-particle processes are calculated the setting should be: Mandelstam → { }.

See also: DiracTrace, OneLoop, OneLoopSum, Tr, TrickMandelstam, SetMandelstam.

Map2

Description

Map2[f, exp] is equivalent to Map if Nterms[exp] > 0, otherwise Map2[f, exp] gives f[exp].
See also: NTerms.

Examples

```
In[1429]:= m1^2
Out[1429]= m2^2

In[1430]:= m3^2
Out[1430]= m4^2

In[1431]:= Map2[f, a - b]
Out[1431]= f(a) + f(-b)

In[1432]:= Map2[f, x]
Out[1432]= f(x)
```

MemoryAvailable

Description

MemoryAvailable is an option of MemSet. It can be set to an integer n, where n is the available amount of main memory in Mega Byte. The default setting is $MemoryAvailable.
See also: MemSet, $MemoryAvailable.

MemSet

Description

MemSet[f[x_], body] is like f[x_] := f[x] = body, but dependend on the value of the setting of MemoryAvailable → memorycut (memorycut - MemoryInUse[]/Map2[*f*, {*a, b, c*}]) MemSet[f[x_], body] may evaluate as f[x_] := body.
See also: MemoryAvailable.

MetricTensor

Description

MetricTensor[*f*({*a, b, c*})] is the metric tensor. The default dimension is 4.

See also: FeynCalcExternal, FCE, FCI, MT, MTD.

Examples

```
In[1433]:= Map2[f, 1]
Out[1433]= f(1)

In[1434]:= 10^6
Out[1434]= μ, ν

In[1435]:= MetricTensor[α, β]
Out[1435]= g^αβ

In[1436]:= Contract[%%]
Out[1436]= 4

In[1437]:= MetricTensor[α, β, Dimension → n]
Out[1437]= MetricTensor(α, β, Dimension → n)

In[1438]:= Contract[%%]
Out[1438]= n

In[1439]:= StandardForm[MetricTensor[a, b]]
Out[1439]= MetricTensor[a, b]

In[1440]:= StandardForm[MetricTensor[a, b, Dimension → D]]
Out[1440]= MetricTensor[a, b, Dimension → D]
```

MLimit

Description

MLimit[expr, lims] takes multiple limits of expr using the limits lims.

Examples

```
In[1441]:= StandardForm[FCE[MetricTensor[a, b]]]
Out[1441]= MT[a, b]
```

Momentum

Description

Momentum[p] is the head of a four momentum (p). The internal representation of a four-dimensional p is Momentum[p]. For other than four dimensions: Momentum[p, dim]. Momentum[p, 4] simplifies to Momentum[p].
See also: DiracGamma, Eps, LorentzIndex, MomentumExpand.

Examples

This is a four dimensional momentum.

```
In[1442]:= StandardForm[FCE[MetricTensor[a, b, Dimension → D]]]
Out[1442]= MTD[a, b]
```

As an optional second argument the dimension must be specified if it is different from 4.

```
In[1443]:= MLimit[y Log[y] + Sin[x - 1]/(x - 1), x → 1, y → 0]
Out[1443]= 1
```

The dimension index is supressed in the output.

```
In[1444]:= Momentum[p]
Out[1444]= p

In[1445]:= Momentum[p, D]
Out[1445]= p

In[1446]:= Momentum[p, d]//StandardForm
Out[1446]= Momentum[p, d]

In[1447]:= a1 = Momentum[-q]
Out[1447]= -q

In[1448]:= a1//StandardForm
Out[1448]= -Momentum[q]

In[1449]:= a2 = Momentum[p - q] + Momentum[2q]
Out[1449]= (p - q) + 2 q

In[1450]:= a2//StandardForm
Out[1450]= Momentum[p - q] + 2 Momentum[q]

In[1451]:= a2//MomentumExpand//StandardForm
Out[1451]= Momentum[p] + Momentum[q]

In[1452]:= a2//MomentumCombine//StandardForm
```

MomentumCombine

Description

MomentumCombine[expr] is the inverse operation to MomentumExpand and ExpandScalarProduct. MomentumCombine combines also Pair‘s.

See also: ExpandScalarProduct, Momentum, MomentumExpand, MomentumCombine2.

Examples

```
In[1453]:= Momentum[p + q]
Out[1453]= ChangeDimension[Momentum[p], d]//StandardForm

In[1454]:= Momentum[p, d]
Out[1454]= Clear[a1, a2]

In[1455]:= Momentum[p] - 2 Momentum[q] //MomentumCombine //StandardForm
Out[1455]= Momentum[p - 2 q]

In[1456]:= t1 = FV[p, μ] + 2 FV[q, μ]
Out[1456]= p^μ + 2 q^μ

In[1457]:= MomentumCombine[t1]
Out[1457]= (p + 2 q)^μ

In[1458]:= MomentumCombine[t1]//StandardForm
Out[1458]= Pair[LorentzIndex[μ], Momentum[p + 2 q]]

In[1459]:= MomentumCombine[t1]//ExpandScalarProduct
```

MomentumCombine2

Description

MomentumCombine2[expr] is the inverse operation to MomentumExpand and ExpandScalarProduct. MomentumCombine2 combines also FourVectors.

See also: MomentumCombine, MomentumExpand, ExpandScalarProduct, FourVector.

Examples

```
In[1460]:= p^μ + 2 q^μ
Out[1460]= StandardForm[%]

In[1461]:= Pair[LorentzIndex[μ], Momentum[p]] + 2 Pair[LorentzIndex[μ], Momentum[q]]
Out[1461]= Clear[t1]
```

MomentumExpand

Description

MomentumExpand[expr] expands Momentum[a+b+ ...] in expr into Momentum[a] + Momentum[b] + ...

See also: ExpandScalarProduct, MomentumCombine.

Examples

```
In[1462]:= 3 Pair[LorentzIndex[μ], Momentum[p]] + 2 Pair[LorentzIndex[μ], Momentum[q]]
Out[1462]= 3 p^μ + 2 q^μ

In[1463]:= MomentumCombine2[%]//StandardForm
Out[1463]= Pair[LorentzIndex[μ], Momentum[3 p + 2 q]]

In[1464]:= MomentumExpand[Momentum[p + q]]//StandardForm
Out[1464]= Momentum[p] + Momentum[q]

In[1465]:= ScalarProduct[p + q, r]
Out[1465]= (p + q) · r

In[1466]:= %//StandardForm
Out[1466]= Pair[Momentum[p + q], Momentum[r]]

In[1467]:= MomentumExpand[ScalarProduct[p + q, r]]
Out[1467]= (p + q).(r)

In[1468]:= %//StandardForm
Out[1468]= Pair[Momentum[p] + Momentum[q], Momentum[r]]

In[1469]:= MomentumExpand[ScalarProduct[p + q, r - p]]
Out[1469]= (p + q).(r - p)
```

MT

Description

MT[%//StandardForm] is the metric tensor in 4 dimensions.
See also: FeynCalcExternal, FCE, FCI, MetricTensor, MTD.

Examples

```
In[1470]:= Pair[Momentum[p] + Momentum[q], -Momentum[p] + Momentum[r]]
Out[1470]= Calc[%]

In[1471]:= -p^2 - p · q + p · r + q · r
Out[1471]= μ, ν

In[1472]:= MT[α, β]
Out[1472]= g^αβ

In[1473]:= Contract[MT[α, β] MT[α, β]]
Out[1473]= 4

In[1474]:= MT[a, b]//StandardForm
Out[1474]= MT[a, b]
```

MTD

Description

MTD[FCI[MT[*a, b*]]//StandardForm] is the metric tensor in D dimensions.
See also: FeynCalcExternal, FCE, FCI, MetricTensor, MT.

Examples

```
In[1475]:= Pair[LorentzIndex[a], LorentzIndex[b]]
Out[1475]= FCE[FCI[MT[a, b]]]//StandardForm

In[1476]:= MT[a, b]
Out[1476]= μ, ν

In[1477]:= MTD[α, β]
Out[1477]= g^αβ

In[1478]:= Contract[MTD[α, β] MTD[α, β]]
Out[1478]= D

In[1479]:= MTD[α, β]//StandardForm
Out[1479]= MTD[α, β]
```

NegativeInteger

Description

NegativeInteger is a data type. E.g. DataType[n, NegativeInteger] can be set to True.
See also: DataType.

Nf

Description

Nf denotes the number of flavors.
See also: Amplitude, CounterTerm, Convolute, AnomalousDimension, SplittingFunction.

Nielsen

Description

Nielsen[i,j, x] denotes Nielsen's polylogarithm.

```
In[1480]:= FCI[MTD[α, β]]//StandardForm
Out[1480]= Pair[LorentzIndex[α, D], LorentzIndex[β, D]]
```

See also: SimplifyPolyLog.

Examples

```
In[1481]:= FCE[FCI[MTD[μ, ν]]]//StandardForm
Out[1481]= MTD[μ, ν]
```

Numerical evaluation is done via N[Nielsen[n_,p_,x_]] := (-1)^(n+p-1)/(n-1)!/p! NIntegrate[Log[1-x t]^p Log[t]^(n-1)/t,{t,0,1}];

```
In[1482]:= Options[Nielsen]
Out[1482]= {PolyLog → False}
```

Some special values are built in.

```
In[1483]:= Nielsen[1, 2, x]
Out[1483]= S12(x)

In[1484]:= N[Nielsen[1, 2, 0.45]]
Out[1484]= 0.0728716

In[1485]:= {Nielsen[1, 2, 0], Nielsen[1, 2, -1], Nielsen[1, 2, 1/2], Nielsen[1, 2, 1]}
Out[1485]= {0, ζ(3)/8, ζ(3)/8, ζ(3)}

In[1486]:= Nielsen[1, 2, x, PolyLog → True]
Out[1486]= 1/2 Log(x) Log^2(1 - x) + Li2(1 - x) Log(1 - x) - Li3(1 - x) + ζ(3)
```

NonCommFreeQ

Description

NonCommFreeQ[exp] yields True if exp contains no non-commutative objects (i.e. those objects which are listed in $NonComm) or only non-commutative objects inside DiracTrace's or SUNTrace's.

See also: $NonComm, NonCommQ, DiracTrace, SUNTrace.

NonCommQ

Description

NonCommQ[exp] yields True if exp contains non-commutative objects (i.e. those objects which are listed in $NonComm) not inside DiracTrace's or SUNTrace's.

See also: $NonComm, NonCommFreeQ, DiracTrace, SUNTrace.

NonCommutative

Description

NonCommutative is a data type which may be used, e.g., as: DataType[x, NonCommutative] = True.
See also: DataType, DeclareNonCommutative.

NTerms

Description

NTerms[x] is equivalent to Length if x is a sum; otherwise NTerms[x] returns 1, except NTerms[0] → 0.

Examples

```
In[1487]:= Nielsen[1, 3, x, PolyLog → True]
Out[1487]= -1/6 Log(x) Log^3(1 - x) - 1/2 Li_2(1 - x) Log^2(1 - x) + Li_3(1 - x) Log(1 - x) - Li_4(1 - x) + π^4/90

In[1488]:= Nielsen[3, 1, x, PolyLog → True]
Out[1488]= Li_4(x)

In[1489]:= NTerms[a - b]
Out[1489]= 2

In[1490]:= NTerms[a b c]
Out[1490]= 1
```

NumberOfMetricTensors

Description

NumberOfMetricTensors is an option of Tdec.
See also: Tdec.

NumericalFactor

Description

NumericalFactor[expr] gives the overall numerical factor of expr.

NumericQ1

Description

NumericQ1[x,{a,b,..}] is like NumericQ, but assumes that {a,b,..} are numeric quantities.

Examples

```
In[1491]:= NTerms[9]
Out[1491]= 1

In[1492]:= NTerms[0]
Out[1492]= 0

In[1493]:= NumericQ[3 a + Log[b] + c^2]
Out[1493]= False
```

OneLoop ***unfinished***

See also the old FeynCalc documentation at http://www.mertig.com/oldfc/. NOTICE: While OneLoop is restricted to 't Hooft Feynman gauge the function OneLoopSimplify does not have this restriction (but is usually slower). OneLoop handles selfenergies, vertex and box-graphs (those only up to 3rd rank tensor in the integration variable).

WARNING: If you encounter anomalies: NumericQ1$\left[3\ a + \log[b] + c^2, \{\}\right]$ is calculated in D dimensions has changed compared to the old FeynCalc version. Please keep in mind that the issue of Falseschemes is inherintly tricky.

Description

OneLoop[q, amplitude] calculates the one-loop Feynman diagram amplitude (n-point, where $n \leq 4$ and the highest tensor rank of the integration momenta (after cancellation of scalar products) may be 3; unless OneLoopSimplify is used).

The argument q denotes the integration variable, i.e., the loop momentum. OneLoop[name, q, amplitude] has as first argument a name of the amplitude. If the second argument has head FeynAmp then OneLoop[q, FeynAmp[name, k, expr]] and OneLoop[FeynAmp[name, k, expr]] tranform to OneLoop[name, k, expr].

```
In[1494]:= NumericQ1[3 a + Log[b] + c^2, {a, b, c}]
Out[1494]= True
```

See also: B0, C0, D0, OneLoopSimplify, TID, TIDL, $LimitTo4.

Examples

```
In[1495]:= The default setting of $West is True,
           i.e., the way tr(γ^μ γ^ν γ^ρ γ^σ γ^τ γ^λ γ^5)
Out[1495]= γ^5

In[1496]:= Options[OneLoop]
Out[1496]= {Apart2 → True, CancelQP → True, DenominatorOrder → False, Dimension → D,
             FinalSubstitutions → {}, Factoring → False, FormatType → InputForm, InitialSubstitutions → {}
             IntermediateSubstitutions → {}, IsolateNames → False, Mandelstam → {}, OneLoopSimplify → Fals
             Prefactor → 1, ReduceGamma → False, ReduceToScalars → False, SmallVariables → {},
             WriteOut → False, WriteOutPaVe → /opt/cvs/HighEnergyPhysics/Phi/Storage/, Sum → True}

In[1497]:= -I/π^2 FAD[{q, m}]//FCI
```

Remember that FAD[{q,mf},{q-k,mf}] is a fast possibility to enter $-\frac{i}{\pi^2 (q^2-m^2)}$

```
In[1498]:= OneLoop[q, %]
Out[1498]= A0(m^2)
```

The input to OneLoop may be in 4 dimensions, since the function changes the dimension of the objects automatically to the setting of the Dimension option (D by default).

```
In[1499]:= mf/: MakeBoxes[mf, TraditionalForm] = InterpretationBox[SubscriptBox["m", "f"], mf];
Out[1499]= 1/( (q^2 - mf^2) ( (q - k)^2 - mf^2) ).

In[1500]:= t = I el^2/(16 π^4 (1 - D)) FAD[{q, mf}, {q - k, mf}]
              DiracTrace[(mf + GS[q - k]).GA[μ].(mf + GS[q]).GA[μ]] //FCI
Out[1500]= i el^2 tr((m_f + γ · (q - k)).γ^μ.(m_f + γ · q).γ^μ) / (16 (1 - D) π^4 (q^2 - m_f^2).((q - k)^2 - m_f^2))

In[1501]:= OneLoop[q, t]
Out[1501]= -(el^2 (-8/3 B0(k^2, m_f^2, m_f^2) m_f^2 - 8 m_f^2/3 + 8/3 A0(m_f^2) - 4/3 B0(k^2, m_f^2, m_f^2) k^2 + 4 k^2/9)) / (16 π^2)

In[1502]:= FullSimplify[%]
Out[1502]= el^2 (6 m_f^2 - 6 A0(m_f^2) - k^2 + 3 B0(k^2, m_f^2, m_f^2) (2 m_f^2 + k^2)) / (36 π^2)

In[1503]:= SP[k, r] FAD[{k, m}, k - p]//FCI
```

OneLoopSimplify

Description

OneLoopSimplify[amp, q] simplifies the one-loop amplitude amp. The second argument denotes the integration momentum.

```
In[1504]:= k · r / ((k^2 - m^2) . (k - p)^2)
Out[1504]= OneLoop[k, %]
```

See also: OneLoop, TID, TIDL.

Examples

```
In[1505]:= i π^2 ( - B0(0, m^2, m^2) p · r m^2 / (2 p^2) + B0(p^2, 0, m^2) p · r m^2 / (2 p^2) - p · r m^2 / (2 p^2) + 1/2 B0(p^2, 0, m^2) p · r )
Out[1505]= Clear[t]

In[1506]:= Options[OneLoopSimplify]
Out[1506]= {Collecting → False, Dimension → D, DimensionalReduction → False,
  DiracSimplify → True, FinalSubstitutions → {}, IntegralTable → {}, OPE1Loop → False,
  ScalarProductCancel → True, SUNNToCACF → True, SUNTrace → False}

In[1507]:= SP[k, r] FAD[{k, m}, k - p]//FCI
Out[1507]= k · r / ((k^2 - m^2) . (k - p)^2)

In[1508]:= OneLoopSimplify[%, k]
Out[1508]= (m^2 + p^2) p · r / (2 (k^2 - m^2) . (k - p)^2 p^2) - p · r / (2 (k^2 - m^2) p^2)

In[1509]:= OneLoopSimplify[%/.m → 0, k]
Out[1509]= p · r / (2 k^2 . (k - p)^2)

In[1510]:= FAD[k, k, k - p1, k - p2] FVD[k, μ]//FCI
Out[1510]= k^μ / (k^2 . k^2 . (k - p1)^2 . (k - p2)^2)

In[1511]:= OneLoopSimplify[ %, k]
Out[1511]= p2^μ p1^2 - p1^μ p1 · p2 / (2 k^2 . k^2 . (k - p1)^2 (p1 · p2^2 - p1^2 p2^2)) + (p2^μ p1^2 p1 · p2 + p1^μ p2^2 p1 · p2 - p1^μ p1^2 p2^2 - p2^μ p1^2 p2^2) / (2 k^2 . k^2 . (k - p1)^2 . (k - p2)^2 (p1 · p2^2 - p1^2 p2^2)) -
  (p2^μ p1 · p2 - p1^μ p2^2) / (2 k^2 . k^2 . (k - p2)^2 (p1 · p2^2 - p1^2 p2^2)) - (p2^μ p1^2 - p1^μ p1 · p2 - p2^μ p1 · p2 + p1^μ p2^2) / (2 k^2 . (k - p1)^2 . (k - p2)^2 (p1 · p2^2 - p1^2 p2^2))
```

OneLoopSum

Description

OneLoopSum[FeynAmp[...], FeynAmp[...] , ...] will calculate a list of Feynman amplitudes by replacing FeynAmp step by step by OneLoop.
See also: OneLoop.

OPE

Description

OPE is a convenience variable to separate OPE insertions.
OPE is also an option of several input functions like GluonPropagator.
The OPE variable has the property to vanish for higher powers:

```
In[1512]:= FCE[%]/.SPD[p1] → 0//FCI
```

$$\text{Out[1512]}= \frac{p_1^{\mu} p_2^2}{2\,k^2.k^2.(k-p_1)^2.(k-p_2)^2\,p_1\cdot p_2} - \frac{p_1^{\mu}}{2\,k^2.k^2.(k-p_1)^2\,p_1\cdot p_2} - \frac{p_2^{\mu}\,p_1\cdot p_2 - p_1^{\mu} p_2^2}{2\,k^2.k^2.(k-p_2)^2\,p_1\cdot p_2^2} - \frac{-p_1^{\mu}\,p_1\cdot p_2 - p_2^{\mu}\,p_1\cdot p_2 + p_1^{\mu} p_2^2}{2\,k^2.(k-p_1)^2.(k-p_2)^2\,p_1\cdot p_2^2}$$

See also: OPE1Loop.

OPEDelta

Description

OPEDelta is a lightlike axial vector as used e.g. in the operator product expansion in QCD.
See also: Twist2QuarkOperator.

Examples

```
In[1513]:= OneLoopSimplify[FAD[k - p1, k - p2] SP[k, l]^2, k]
```

$$\text{Out[1513]}= -\frac{D\,l\cdot p_1^2 + 2\,D\,l\cdot p_2\,l\cdot p_1 - 4\,l\cdot p_2\,l\cdot p_1 + D\,l\cdot p_2^2 - l^2 p_1^2 + 2\,l^2\,p_1\cdot p_2 - l^2 p_2^2}{4\,(1-D)\,k^2.(k-p_1+p_2)^2}$$

```
In[1514]:= {OPE, OPE^2, OPE^3}
Out[1514]= {Ω, 0, 0}

In[1515]:= FourVector[OPEDelta, μ]
```

$$\text{Out[1515]}= \Delta_{\mu}$$

OPEi, OPEj, OPEk, OPEl, OPEm, OPEn, OPEo

Description

OPEi, etc. are variables with DataType PositiveInteger which are used in functions relating to the operator product expansion.

Examples

```
In[1516]:= Contract[%%]
Out[1516]= 0

In[1517]:= SP[OPEDelta, OPEDelta]
Out[1517]= 0

In[1518]:= OPEi
Out[1518]= i
```

Re has been changed:

```
In[1519]:= DataType[OPEi, OPEj, OPEk, OPEl, OPEm, OPEn, OPEo, PositiveInteger]
Out[1519]= {True, True, True, True, True, True, True}

In[1520]:= PowerSimplify[{(-1)^(2OPEi), (-1)^(2OPEj), (-1)^(2OPEk),
              (-1)^(2OPEl), (-1)^(2OPEm), (-1)^(2OPEn), (-1)^(2OPEo)}]
Out[1520]= {1, 1, 1, 1, 1, 1, 1}

In[1521]:= {Re[OPEi] > -3, Re[OPEi] > -2, Re[OPEi] > -1,  Re[OPEi] > 0, Re[OPEi] > 1}
Out[1521]= {Re(i) > -3, Re(i) > -2, Re(i) > -1, Re(i) > 0, Re(i) > 1}
```

OPEInsert ***unfinished***

Description

OPEInsert[diagram_String, name_, rhifile_String] or OPEInsert[diagram_String, name_]. The setting of the option EpsContract (4 or D) determines whether the Levi-Civita tensors are contracted in 4 or D dimensions.
See also: RHI.

Examples

```
In[1522]:= {Re[-OPEi + OPEm] > 0, Re[-OPEi + OPEm] > 1, Re[-OPEi + OPEm] > 2}
```

OPEInt ***unfinished***

Description

OPEInt[expr, q, p, x]. The dimension is changed to the one indicated by the option Dimension. The setting of the option EpsContract determines the dimension in which the Levi-Civita tensors are contracted.

See also: RHI.

Examples

```
In[1523]:= {Re(m) - Re(i) > 0, Re(m) - Re(i) > 1, Re(m) - Re(i) > 2}
```

OPEIntDelta ***unfinished***

Description

OPEIntDelta[expr, x, m] introduces the delta(1-x) (DeltaFunction[1-x]). The *Mathematica* Integrate function is called and each integration (from 0 to 1) is recorded for reference (and bug-checking) in the global list $MIntegrate.
Notice that the dimension specified by the option should also be the dimension used in expr. It is replaced in OPEIntDelta by (4+Epsilon).

See also: RHI.

Examples

```
In[1524]:= {Re[OPEm] > -3, Re[OPEm] > -2, Re[OPEm] > -1,  Re[OPEm] > 0, Re[OPEm] > 1}
```

OPEIntegrate ***unfinished***

Description

OPEIntegrate[expr, q, x]. The dimension is changed to the one indicated by the option Dimension. The setting of the option EpsContract determines the dimension in which the Levi-Civita tensors are contracted.

See also: RHI.

Examples

```
In[1525]:= {Re(m) > -3, Re(m) > -2, Re(m) > -1, Re(m) > 0, Re(m) > 1}
```

OPEIntegrateDelta ***unfinished***

Description

OPEIntegrateDelta[expr, x, m] introduces the δ(1-x) (DeltaFunction[1-x]). The *Mathematica* Integrate function is called and each integration (from 0 to 1) is recorded for reference (and bug-checking) in the global list $MIntegrate.
Notice that the dimension specified by the option should also be the dimension used in expr. It is replaced in OPEIntegrateDelta by (4+Epsilon).
See also: RHI.

Examples

```
In[1526]:= {}
```

OPEIntegrate2 ***unfinished***

Description

OPEIntegrate2[exp, k] does special loop (tensorial) integrations. Only the residue is calculated.
See also: OPEIntegrate.

Examples

OPESumExplicit

Description

OPESumExplicit[exp] calculates OPESum's.
See also: OPESum, OPESumSimplify.

Examples

```
In[1527]:= {}
```

```
In[1528]:= {}
```

```
In[1529]:= {}
```

```
In[1530]:= {}
```

```
In[1531]:= {}
```

OPESum

Description

OPESum[exp, {i, 0, m}] denotes a symbolic sum.The syntax is the same as for Sum.
See also: OPESumExplicit, OPESumSimplify.

Examples

```
Out[1531]= t1 = OPESum[A^iB^(m - i - 3), {i, 0, m - 3}]
```

$$In[1532]:= \sum_{i=0}^{m-3} A^i B^{-i+m-3}$$

```
Out[1532]= OPESumExplicit[t1]
```

$$In[1533]:= \frac{A^{m-2}}{A-B} - \frac{B^{m-2}}{A-B}$$

```
Out[1533]= t2 = OPESum[a^ib^(j - i)c^(m - j - 4), {i, 0, j}, {j, 0, m - 4}]
```

$$In[1534]:= \sum_{j=0}^{m-4} (j+1) a^i b^{j-i} c^{-j+m-4}$$

```
Out[1534]= OPESumExplicit[t2]
```

$$In[1535]:= -\frac{c\,a^{m-2}}{(a-b)(a-c)(b-c)} + \frac{b\,a^{m-2}}{(a-b)(a-c)(b-c)} + \frac{c\,b^{m-2}}{(a-b)(a-c)(b-c)} - \frac{a\,b^{m-2}}{(a-b)(a-c)(b-c)} - \frac{b\,c^{m-2}}{(a-b)(a-c)(b-c)} + \frac{a\,c^{m-2}}{(a-b)(a-c)(b-c)}$$

OPESumSimplify

Description

OPESumSimplify[exp] simplifies OPESum's in exp.
See also: OPESum, OPESumExplicit.

Examples

```
In[1536]:= Clear[t1, t2];
Out[1536]= t1 = OPESum[SO[p]^OPEiSO[k]^(OPEm - OPEi - 3), {OPEi, 0, OPEm - 3}]
```

$$In[1537]:= \sum_{i=0}^{m-3} (\Delta\cdot k)^{-i+m-3} (\Delta\cdot p)^i$$

```
Out[1537]= OPESumExplicit[t1]
```

$$In[1538]:= \frac{(\Delta\cdot k)^{m-2}}{\Delta\cdot k - \Delta\cdot p} - \frac{(\Delta\cdot p)^{m-2}}{\Delta\cdot k - \Delta\cdot p}$$

```
Out[1538]= t2 = OPESum[a^ib^(j - i)c^(m - j - 4), {i, 0, j}, {j, 0, m - 4}]
```

```
In[1539]:= Sum[(j + 1) a^i b^(j-i) c^(-j+m-4), {j, 0, m-4}]
Out[1539]= OPESumExplicit[t2]
```

$$\text{In[1540]:=}\quad -\frac{c\,a^{m-2}}{(a-b)\,(a-c)\,(b-c)}+\frac{b\,a^{m-2}}{(a-b)\,(a-c)\,(b-c)}+\frac{c\,b^{m-2}}{(a-b)\,(a-c)\,(b-c)}-\frac{a\,b^{m-2}}{(a-b)\,(a-c)\,(b-c)}-\frac{b\,c^{m-2}}{(a-b)\,(a-c)\,(b-c)}+\frac{a\,c^{m-2}}{(a-b)\,(a-c)\,(b-c)}$$

```
Out[1540]= Clear[t1, t2]
```

OPE1Loop ***unfinished***

Description

OPE1Loop[q1, amp]. OPE1Loop[{q1,q2}, amp] does sub-loop decomposition.
See also: OPESum.

Examples

```
In[1541]:= OPESum[(-SOD[p])^(OPEi + 1) SOD[p - q]^(OPEm - OPEi - 2), {OPEi, 0, OPEm}]
```

OPE2AI ***unfinished***

Description

OPE2AI[tabname, listtodo, q1,q2,p] is ....

```
In[1542]:= Sum[(-(Δ · p))^(i+1) (Δ · (p - q))^(-i+m-2), {i, 0, m}]
Out[1542]= OPESumSimplify[%]
```

See also: OPESum.

Examples

```
In[1543]:= -Sum[(-1)^i (Δ · p)^i (Δ · (p - q))^(-i+m-2) Δ · p, {i, 0, m}]
```

OPE2TID ***unfinished***

Description

OPE2TID[exp, k1, k2, p]. The setting of the option EpsContract determines the dimension in which the Levi-Civita tensors are contracted.

```
In[1544]:= OPESumSimplify[OPESum[{OPEi, 0, OPEm}] a^OPEi]
Out[1544]= ∑_{i=0}^{m} a^i
```

See also: OPESum.

Examples

```
In[1545]:= OPESumSimplify[OPESum[{j, 0, i}, {i, 0, m}] a^(j - i) b^i]
```

OptionsSelect

Description

OptionsSelect[function,opts] returns the option settings of opts accepted by function. When an option occurs several times in opts, the first setting is selected.

Pair

Description

Pair[x, y] is the head of a special pairing used in the internal representation: x and y may have heads LorentzIndex or Momentum. If both x and y have head LorentzIndex, the metric tensor is understood. If x and y have head Momentum, a scalar product is meant. If one of x and y has head LorentzIndex and the other Momentum, a Lorentz vector $\sum_{i=0}^{m} (i+1) a^{j-i} b^i$ is understood.

See also: FourVector, FV, FVD, MetricTensor, MT, MTD, ScalarProduct, SP, SPD.

Examples

This represents a four-dimensional metric tensor.

```
In[1546]:= %//StandardForm
Out[1546]= OPESum[a^(-i+j) b^i, {i, 0, m}, {j, 0, i}]
```

This is a D-dimensional metric tensor.

```
In[1547]:= {}

In[1548]:= Options[OPE2AI]
Out[1548]= {Directory → /home/rolfm}

In[1549]:= {}

In[1550]:= Options[OPE2TID]
Out[1550]= {Uncontract → False, Contract → True, Dimension → D, EpsContract → False}
```

```
In[1551]:= {}

In[1552]:= p^μ
Out[1552]= Pair[LorentzIndex[α],LorentzIndex[β]]

In[1553]:= g^αβ
Out[1553]= Pair[LorentzIndex[α,D],LorentzIndex[β,D]]

In[1554]:= g^αβ
Out[1554]= Pair[LorentzIndex[α],Momentum[p]]

In[1555]:= p^α
Out[1555]= Pair[Momentum[q],Momentum[p]]

In[1556]:= p · q
Out[1556]= Pair[Momentum[p],Momentum[p]]
```

PairCollect

Description

PairCollect is an option for DiracTrace specifying if the result is collected with respect to Pair's.
See also: DiracTrace, Pair.

PairContract

Description

PairContract is like Pair, but with (local) contraction properties.
See also: Pair, Contract.

PairContract2

Description

PairContract2 is like Pair, but with local contraction properties among PairContract2's.
See also: Pair, Contract, PairContract.

PairContract3

Description

PairContract3 is like Pair, but with local contraction properties among PairContract3's.
See also: Pair, Contract, PairContract.

PartialD

Description

PartialD[p^2] denotes the four-dimensional Pair[Momentum[$p - q$], Momentum[p]] PartialD is used to denote derivative fields. PartialD[LorentzIndex[$p \cdot (p - q)$]] denotes the Pair[Momentum[p], Momentum[p]]^2-dimensional p^4
See also: ExpandPartialD, LeftPartialD, LeftRightPartialD, RightPartialD.

Pair[Momentum[p], Momentum[p]]^3

```
In[1557]:= p^6
Out[1557]= Pair[LorentzIndex[α, n - 4], LorentzIndex[β]]//Contract

In[1558]:= 0
Out[1558]= ExpandScalarProduct[Pair[Momentum[p - q], Momentum[p]]]

In[1559]:= p^2 - p · q
Out[1559]= Pair[Momentum[-q], Momentum[p]] + Pair[Momentum[q], Momentum[p]]
```

PartialDRelations

Description

PartialDRelations is an option for ExpandPartialD. It is a list of rules applied by ExpandPartialD at the end.
See also: PartialD, ExpandPartialD, LeftPartialD, LeftRightPartialD, RightPartialD.

0

```
In[1560]:= μ
Out[1560]= ∂_μ.

In[1561]:= μ, D
Out[1561]= D
```

PartialFourVector

Description

PartialFourVector[exp, FourVector[p, mu]] calculates the partial derivative of exp w.r.t. p(mu). PartialFourVector[exp, FourVector[p, mu], FourVector[p,nu], ...] gives the multiple derivative.

See also: FourVector, Pair, Contract.

$\partial_\mu .$

```
In[1562]:= Examples
Out[1562]= QuantumField[A, {μ}].LeftPartialD[ν]
```

PartialIntegrate

Description

PartialIntegrate[exp, ap, t] does a partial integration of the definite integral Integrate[exp,{t,0,1}], with ap the factor that is to be integrated and exp/ap the factor that is to be differentiated.

```
In[1563]:= A_μ . (←∂)_ν
Out[1563]= ExpandPartialD[%]
```

$\partial_\nu A_\mu$

```
In[1564]:= StandardForm[%]

QuantumField[PartialD[LorentzIndex[ν]], A, LorentzIndex[μ]]

Examples
Out[1564]= QuantumField[A, {μ}].QuantumField[B, {μ}].LeftPartialD[ν]

In[1565]:= A_μ . B_μ . (←∂)_ν
Out[1565]= ExpandPartialD[%, PartialDRelations → {A → C}]

In[1566]:= C_μ . ∂_ν B_μ + ∂_ν C_μ . B_μ
Out[1566]= Examples

In[1567]:= PartialFourVector[a ℮^FourVector[p,ν]^2, FourVector[p, μ]]
Out[1567]= 2 a ℮^p^2 p^μ

In[1568]:= Options[PartialIntegrate]
Out[1568]= {Integrate → Integrate}

In[1569]:= Examples
```

PartitHead

Description

PartitHead[expr, h] returns a list {ex1, h[ex2]} with ex1 free of expressions with head h, and h[ex2] having head h.

PauliSigma

Description

PauliSigma denotes the vector of the 3 Pauli matrices. PauliSigma[1], PauliSigma[2], PauliSigma[3] give the explicit Pauli matrices. PauliSigma[] yields {PauliSigma[1], PauliSigma[2], PauliSigma[3]}.

```
In[1570]:= PartialIntegrate[f[x]g[x], g[x], x]
Out[1570]= Integrate :: ilim : Invalid integration variable or limit(s) in 1.
```

PaVe

Description

PaVe[i,j,... {p10,p12,...},{m1^2, mw^2, ...}] denotes the invariant (and scalar) Passarino-Veltman integrals, i.e. the coefficient functions of the tensor integral decomposition. Joining plist and mlist gives the same conventions as for A0, B0, C0, D0. Automatic simlifications are performed for the coefficient functions of two-point integrals and for the scalar integrals.

For a more detailed description see the manual (no changes from there basically): http://www.mertig.com/oldfc

See also: PaVeReduce.

Examples

Some of the PaVe's reduce to special cases.

```
In[1571]:= Integrate :: ilim : Invalid integration variable or limit(s) in 0.
Out[1571]= -f(0) ∫ g(0)d0 + f(1) ∫ g(1)d1 - ( ∫ g(x)dx ) f'(x)

In[1572]:= f[x_] = Integrate[Log[3x + 2], x]
Out[1572]= 1/3 (-3 x - 2) + 1/3 (3 x + 2) Log(3 x + 2)
```

PaVeOrderList

Description

PaVeOrderList is an option for PaVeOrder and PaVeReduce, specifying in which order the arguments of D0 are to be permuted.
See also: PaVeOrder, PaVeReduce, PaVe, D0.

PaVeOrder

Description

PaVeOrder[expr] orders the arguments of all D0 in expr in a standard way. PaVeOrder[expr, PaVeOrderList → { {..., s, u, ...}, {... g[x_] = D$\left[\frac{1}{\log[3x+2]}, x\right]$, $-\frac{3}{(3x+2)\log^2(3x+2)}$ ...}, ...}] orders the arguments of all D0 in expr according to the specified ordering lists. The lists may contain only a subsequence of the D0-variables.
See also: PaVeReduce.

Examples

```
In[1573]:= Integrate[PartialIntegrate[f[x]g[x], f[x], x], {x, 0, 1}]//FullSimplify
```

$$\text{Out[1573]}= -\frac{\text{Log}\left(\frac{32}{25}\right)}{\text{Log(5) Log(8)}}$$

```
In[1574]:= Integrate[f[x]g[x], {x, 0, 1}]//Simplify
```

$$\text{Out[1574]}= -\frac{\text{Log}\left(\frac{32}{25}\right)}{\text{Log(5) Log(8)}}$$

```
In[1575]:= Clear[f, g]
Out[1575]= PauliSigma[]//TableForm
```

PaVeReduce

Description

PaVeReduce[expr] reduces all Passarino-Veltman integrals (i.e. all PaVe's) in expr down to scalar A0, B0, C0 and D0.

```
In[1576]:=  0    1
            1    0

            0    i
            -i   0

            1    0
            0    -1
```

```
Out[1576]= PaVe[0, {a, b, c, d, e, f}, {m1, m2, m3, m4}]

In[1577]:= D0(a, b, c, d, e, f, m1, m2, m3, m4)
PaVe[0, 0, {pp}, {m^2, M^2}]
```

See also: FRH, PaVeOrder.

Examples

$$\text{In[1578]:=}\quad \frac{1}{3}\,B_0(pp, m^2, M^2)\,m^2 + \frac{1}{18}\,(3\,m^2 + 3\,M^2 - pp) + \frac{1}{6}\left(A_0(M^2) + (m^2 - M^2 + pp)\left(\frac{(M^2 - m^2)\,(B_0(pp, m^2, M^2) - B_0(0, m^2, M^2))}{2\,pp} - \frac{1}{2}\,B_0(pp, m^2, M^2)\right)\right)$$

```
Out[1578]= m1^2

In[1579]:= m2^2
Out[1579]= PaVeOrder[D0[me2, me2, mw2, mw2, t, s, me2, 0, me2, 0],
             PaVeOrderList → {me2, me2, 0, 0}]

In[1580]:= D0(me2, s, mw2, t, mw2, me2, me2, 0, 0, me2)
Out[1580]= PaVeOrder[D0[me2, me2, mw2, mw2, t, s, me2, 0, me2, 0],
             PaVeOrderList → {me2, me2, 0, 0}]

In[1581]:= D0(me2, s, mw2, t, mw2, me2, me2, 0, 0, me2)
Out[1581]= PaVeOrder[
             D0[a, b, c, d, e, f, m12, m22, m32, m42] + D0[me2, me2, mw2, mw2, t, s, me2, 0, me2, 0],
             PaVeOrderList → {{me2, me2, 0, 0}, {f, e}}]

In[1582]:= D0(a, d, c, b, f, e, m22, m12, m42, m32) + D0(me2, s, mw2, t, mw2, me2, me2, 0, 0, me2)
Out[1582]= Options[PaVeReduce]

In[1583]:= {Dimension → True, IsolateNames → False, Mandelstam → {},
             PaVeOrderList → {}, WriteOutPaVe → /opt/cvs/HighEnergyPhysics/Phi/Storage/}

In[1584]:= ?WriteOutPaVe

In[1585]:= WriteOutPaVe is an option for PaVeReduce and OneLoopSum. If set to
             a string, the results of all Passarino - Veltman PaVe's are stored in
             files with names generated from this string and the arguments of PaVe.
Out[1585]= PaVeReduce[PaVe[1, 2, {s, m^2, m^2}, {m^2, m^2, M^2}], IsolateHead → L]
```

$$\text{In[1586]:=}\quad \frac{(8\,m^2 - 3\,M^2 - 2\,s)\,C_0(m^2, m^2, s, m^2, M^2, m^2)\,M^2}{(4\,m^2 - s)^2} + \frac{(m^2 - M^2)\,B_0(0, m^2, M^2)}{2\,m^2\,(4\,m^2 - s)} - \frac{(8\,m^4 - 10\,M^2\,m^2 - 2\,s\,m^2 + M^2\,s)\,B_0(m^2, m^2, M^2)}{2\,m^2\,(4\,m^2 - s)^2} + \frac{(4\,m^2 - 6\,M^2 - s)\,B_0(s, m^2, m^2)}{2\,(4\,m^2 - s)^2} + \frac{1}{2\,(4\,m^2 - s)}$$

```
In[1587]:= L[10]
Out[1587]= L(10)

In[1588]:= FRH[%]
```

PD

Description

PD is an abbreviation for PropagatorDenominator.
See also: PropagatorDenominator, IFPD.

PlusDistribution

Description

PlusDistribution[1/(1-x)] denotes a distribution (in the sense of the "+" prescription).
See also: Integrate2.

Examples

```
In[1589]:= L(10)
Out[1589]= PaVeReduce[PaVe[2, {SmallVariable[me2], mw2, t}, {SmallVariable[me2], 0, mw2}],
             WriteOutPaVe → "p"]

In[1590]:= B0(0, mw2, mw2)/(mw2 - t) - B0(t, mw2, me2)/(mw2 - t) + 2/(mw2 - t)
Out[1590]= TableForm[ReadList[If[$OperatingSystem === "MacOS", " : ", ""] <>
                "pPaVe2Csmame2mw2tCsmame20mw2.s", String]]

In[1591]:= (2/(mw2 - t) + B0[0, mw2, mw2]/(mw2 - t) - B0[t, mw2, SmallVariable[me2]]/
            (mw2 - t)
           )
Out[1591]= DeleteFile/@FileNames["pPaVe2Csmame2mw2tCsmame20mw2.s"];

In[1592]:= se = SmallVariable[ME2];
Out[1592]= d122 = PaVeReduce[PaVe[1, 2, 2, {se, MW2, MW2, se, S, T}, {0, se, 0, se}],
             Mandelstam → {S, T, U, 2 MW2}, IsolateHead → F]
```

```
In[1593]:= (MW2 - S) T^2 D0(MW2, MW2, ME2, ME2, T, S, ME2, 0, ME2, 0) S^3 /
             (2 (MW2^2 - S U)^3) +
           (MW2 - S)^2 T C0(MW2, S, ME2, ME2, 0, 0) S^2 /
             (MW2^2 - S U)^3 -
           (MW2 - S) T^2 C0(T, ME2, ME2, ME2, ME2, 0) S^2 /
             (2 (MW2^2 - S U)^3) + (MW2^2 - 4 S MW2 + 2 S^2 + S U) B0(S, 0, 0) S /
             (2 (MW2 - S) (MW2^2 - S U)^2) -
           ((4 MW2^5 - 5 S MW2^4 - U MW2^4 - 16 S^2 MW2^3 + 4 S^3 MW2^2 +
               4 S U^2 MW2^2 - 4 S^2 U MW2^2 + 4 S^4 MW2 + 8 S^3 U MW2 + S^2 U^3 + S^3 U^2)
             B0(MW2, 0, ME2))/(2 (MW2 - S) (4 MW2 - T)^2 (MW2^2 - S U)^2) +
           ((MW2 + S) (4 MW2^3 - 9 S MW2^2 - U MW2^2 - 4 S U MW2 + 2 S^3 + 3 S U^2 + 5 S^2 U) B0(T, ME2, ME2))/
             (2 (4 MW2 - T)^2 (MW2^2 - S U)^2) -
           1/(2 (4 MW2 - T)^2 (MW2^2 - S U)^3) ((MW2 + S) (2 MW2^6 - 8 S MW2^5 - 2 T MW2^5 + 12 S^2 MW2^4 +
               20 S T MW2^4 - 8 S^3 MW2^3 - 6 S T^2 MW2^3 - 36 S^2 T MW2^3 + 2 S^4 MW2^2 +
               6 S^2 T^2 MW2^2 + 20 S^3 T MW2^2 + 4 S^2 T^3 MW2 - 6 S^3 T^2 MW2 - 2 S^4 T MW2 - S^2 T^4)
             C0(MW2, MW2, T, ME2, 0, ME2)) - (MW2 + S)/(2 (4 MW2 - T) (MW2^2 - S U))

Out[1593]= Write2["fctd122.for", d122res == d122, FormatType → FortranForm];

In[1594]:= TableForm[
             ReadList[If[$OperatingSystem === "MacOS", " : ", ""] <> "fctd122.for", String]]
```

```
 d122res = (-5.D-1*(MW2 + S))/
&  ((4D0*MW2 - 1D0*T)*(MW2**2 - 1D0*S*U)) -
& (5.D-1*(4D0*MW2**5 - 5D0*MW2**4*S -
&   16D1*MW2**3*S**2 + 4D0*MW2**2*S**3 +
&   4D0*MW2*S**4 - 1D0*MW2**4*U -
&   4D0*MW2**2*S**2*U + 8D0*MW2*S**3*U +
&   4D0*MW2**2*S*U**2 + S**3*U**2 + S**2*U**3)*
&  B0(MW2, 0D0, ME2))/
& ((MW2 - 1D0*S)*(4D0*MW2 - 1D0*T)**2*
&  (MW2**2 - 1D0*S*U)**2) +
& (5.D-1*S*(MW2**2 - 4D0*MW2*S + 2D0*S**2 + S*U)*
&  B0(S, 0D0, 0D0))/
& ((MW2 - 1D0*S)*(MW2**2 - 1D0*S*U)**2) +
& (5.D-1*(MW2 + S)*
&  (4D0*MW2**3 - 9D0*MW2**2*S + 2D0*S**3 -
&   1D0*MW2**2*U - 4D0*MW2*S*U + 5D0*S**2*U +
&   3D0*S*U**2)*B0(T, ME2, ME2))/
& ((4D0*MW2 - 1D0*T)**2*(MW2**2 - 1D0*S*U)**2) -
& (5.D-1*(MW2 + S)*
&  (2D0*MW2**6 - 8D0*MW2**5*S + 12D1*MW2**4*S**2 -
&   8D0*MW2**3*S**3 + 2D0*MW2**2*S**4 -
&   2D0*MW2**5*T + 20D1*MW2**4*S*T -
&   36D1*MW2**3*S**2*T + 20D1*MW2**2*S**3*T -
&   2D0*MW2*S**4*T - 6D0*MW2**3*S*T**2 +
&   6D0*MW2**2*S**2*T**2 - 6D0*MW2*S**3*T**2 +
&   4D0*MW2*S**2*T**3 - 1D0*S**2*T**4)*
&  C0(MW2, MW2, T, ME2, 0D0, ME2))/
& ((4D0*MW2 - 1D0*T)**2*(MW2**2 - 1D0*S*U)**3) +
& ((MW2 - 1D0*S)**2*S**2*T*C0(MW2, S, ME2, ME2, 0D0, 0D0))/
& (MW2**2 - 1D0*S*U)**3 -
& (5.D-1*(MW2 - 1D0*S)*S**2*T**2*
&  C0(T, ME2, ME2, ME2, ME2, 0D0))/(MW2**2 - 1D0*S*U)**3 +
& (5.D-1*(MW2 - 1D0*S)*S**3*T**2*
&  D0(MW2, MW2, ME2, ME2, T, S, ME2, 0D0, ME2, 0D0))/
& (MW2**2 - 1D0*S*U)**3
```

Out[1594]=

Polarization

Description

Polarization[k] = Polarization[k, I] is the head of a polarization momentum with (incoming) momentum k. A slashed polarization vector (e1(k) slash) has to be entered as DiracSlash[Polarization[k]]. The internal representation for a polarization vector e1 corresponding to a boson with four momentum k is: Momentum[Polarization[k, I]]. With this notation transversality of polarization vectors is provided, i.e. , Pair[Momentum[k], Momentum[Polarization[k, I]]] yields 0. Polarization[k,-I] denotes the complex conjugate polarization. Polarization is also an option of various functions related to the operator product expansion. The setting 0 denotes the unpolarized and 1 the polarized case.

See also: PolarizationVector, PolarizationSum, DoPolarizationSums.

Examples

```
In[1595]:= DeleteFile/@FileNames["fctd122.for"];Clear[d122,se];
Out[1595]= PlusDistribution[1/(1-x)]

In[1596]:= (1/(1-x))+
Out[1596]= PlusDistribution[Log[1-x]/(1-x)]

In[1597]:= (Log(1-x)/(1-x))+
Out[1597]= Integrate2[PlusDistribution[1/(1-x)], {x,0,1}]

In[1598]:= 0
Out[1598]= Integrate2[PlusDistribution[Log[1-x]/(1-x)], {x,0,1}]

In[1599]:= 0
Out[1599]= Integrate2[PlusDistribution[Log[1-x]^2/(1-x)], {x,0,1}]

In[1600]:= 0
Out[1600]= PlusDistribution[Log[x (1-x)]/(1-x)]

In[1601]:= Log(x)/(1-x)+(Log(1-x)/(1-x))+
Out[1601]= Polarization[k]
```

PolarizationSum

Description

PolarizationSum[mu,nu, ...] defines (as abbreviations) several polarization sums. The first two arguments are the interpreted as Lorentz indices, all further ones are momenta. PolarizationSum performs no calculations.

```
In[1602]:= ε(k)
Out[1602]= Polarization[k]//StandardForm
```

See also: Polarization.

Examples

```
In[1603]:= Polarization[k, I]
Out[1603]= Polarization[k]//ComplexConjugate

In[1604]:= ε*(k)
Out[1604]= Polarization[k]//ComplexConjugate//StandardForm

In[1605]:= Polarization[k, -I]
Out[1605]= DiracSlash[Polarization[k]]

In[1606]:= γ · ε(k)
Out[1606]= DiracSlash[Polarization[k]]//StandardForm
```

PolarizationUncontract

Description

PolarizationUncontract does Uncontract on scalar products involving polarization vectors.
See also: Polarization, Uncontract.

PolarizationVector

Description

PolarizationVector[p, mu] gives a polarization vector.
See also: FourVector, Pair, Polarization.

Examples

```
In[1607]:= DiracSlash[Polarization[k, I]]
Out[1607]= Pair[ Momentum[k], Momentum[ Polarization[k, I] ] ]

In[1608]:= 0
Out[1608]= Options[PolarizationSum]
```

A polarization vector {Dimension → 4}is a special four-vector.

```
In[1609]:= PolarizationSum[μ, ν]
Out[1609]= -g^(μν)
```

The transverality property PolarizationSum[μ, ν, k] is built in.

```
In[1610]:= k^μ k^ν / k^2 - g^(μν)
Out[1610]= PolarizationSum[μ, ν, k, n]
```

```
In[1611]:= -g^μν - k^μ k^ν n^2/(k · n^2) + (n^μ k^ν + k^μ n^ν)/(k · n)
Out[1611]= PolarizationSum[μ, ν, k, p1 - p2]

In[1612]:= -g^μν - k^μ k^ν p1^2/(k · p1 - k · p2)^2 + 2 k^μ k^ν p1 · p2/(k · p1 - k · p2)^2 - k^μ k^ν p2^2/(k · p1 - k · p2)^2 + ((p1 - p2)^μ k^ν + k^μ (p1 - p2)^ν)/(k · p1 - k · p2)
Out[1612]= PolarizationVector[k, μ]

In[1613]:= ε_μ(k)
Out[1613]= Conjugate[PolarizationVector[k, μ]]
```

Depending on the alphabetical ordering of the momenta simplifcations are done, e.g., $\varepsilon^*_\mu(k)$

```
In[1614]:= ε_μ(k)
Out[1614]= PolarizationVector[k, μ]//StandardForm

In[1615]:= Pair[LorentzIndex[μ], Momentum[Polarization[k, i]]]
Out[1615]= k^μ ε_μ(k) = 0

In[1616]:= a1 = PolarizationVector[k, μ] FourVector[k, μ]
```

PositiveInteger

Description

PositiveInteger is a data type. E.g. DataType[OPEm, PositiveInteger] gives True.
See also: DataType.

PositiveNumber

Description

PositiveNumber is a data type. E.g. DataType[Epsilon, PositiveNumber] = True (by default).
See also: DataType.

PowerFactor

Description

PowerFactor[exp] replaces $k_\mu\ \varepsilon_\mu(k)$Contract[a1]with 0.
See also: PowerSimplify.

Examples

```
In[1617]:= a2 = PolarizationVector[k - p, μ] FourVector[k, μ]
Out[1617]= k_μ ε_μ(k - p)

In[1618]:= Contract[a2]
Out[1618]= k · (ε(k - p))
```

PowerSimplify

Description

PowerSimplify[exp] simplifies $p^\mu\ \varepsilon_\mu(k-p) = -(k-p-k)^\mu \varepsilon_\mu(k-p) = k^\mu \varepsilon_\mu(k-p) = k\cdot(\varepsilon(k-p))$. to a3 = PolarizationVector[$k - p, \mu$] FourVector[p, μ] $p_\mu\ \varepsilon_\mu(k-p)$ and Contract[a3] to $k\cdot(\varepsilon(k-p))$ Clear[a1, a2, a3]; thus assuming that the exponent is an integer (even if it is symbolic). Furthermore x^a and y^a are expanded and $(x\ y)^a \to x^a y^a$ and (-1)^(n m) → 1 and (-1)^(n m) → $x^a\ y^a$ for n even and odd respectively and PowerFactor[%] → $(x\ y)^a$ and $(-x)^a \to (-1)^a$.
See also: DataType, OPEm.

Examples

```
In[1619]:= x^a
Out[1619]= (y - x)^n

In[1620]:= (-1)^n
Out[1620]= (x - y)^n

In[1621]:= (-1)^(a+n)
Out[1621]= i^(a+n)

In[1622]:= i^(2 m)
Out[1622]= (-1)^m
```

Power2

Description

Power2[x, y] represents x^y. Sometimes Power2 is more useful than the Mathematica Power. Power2[-a,b] simplifies to (-1)^b Power2[a,b] (if no Epsilon is in b ...).
See also: PowerFactor.

Examples

```
In[1623]:= (-1)^m
Out[1623]= (-1)^-n

In[1624]:= (-1)^n
Out[1624]= e^(i m π)
```

PostFortranFile

Description

PostFortranFile is an option for Write2 which may be set to a file name (or a list of file names) or a string, which will be put at the end of the generated Fortran file.

See also: Write2, PreFortranFile.

Prefactor

Description

Prefactor is an option for OneLoop and OneLoopSum. If set as option of OneLoop, the amplitude is multiplied by Prefactor before calculation; if specified as option of OneLoopSum, after calculation in the final result as a global factor.

See also: OneLoop, OneLoopSum.

PreFortranFile

Description

PreFortranFile is an option for Write2 which may be set to a file name (or a list of file names) or a string, which will be put at the beginning of the generated Fortran file.

See also: Write2, PostFortranFile.

PropagatorDenominator

Description

PropagatorDenominator[q, m] is a factor of the denominator of a propagator. If p is supposed to be D-dimensional enter: PropagatorDenominator[Momentum[q, D], m]. What is meant is $(-1)^m$ - PowerSimplify[(−1)^(2OPEm)]). PropagatorDenominator[p] evaluates to PropagatorDenominator[p, 0].

See also: FeynAmpDenominator, PropagatorDenominatorExplicit, IFPD.

Examples

```
In[1625]:= 1
Out[1625]= PowerSimplify[(-1)^(OPEm + 2)]

In[1626]:= (-1)^m
Out[1626]= PowerSimplify[(-1)^(OPEm - 2)]

In[1627]:= (-1)^m
Out[1627]= PowerSimplify[I^(2OPEm)]

In[1628]:= (-1)^m
Out[1628]= Power[-a, b]

In[1629]:= (-a)^b
Out[1629]= Power2[-a, b]

In[1630]:= (-1)^b a^b
Out[1630]= 1/(q^2

In[1631]:= m^2
Out[1631]= PropagatorDenominator[p, m]

In[1632]:=  1
           ------
           2    2
           p  - m
```

PropagatorDenominatorExplicit

Description

PropagatorDenominatorExplicit[exp] changes each occurence of PropagatorDenominator[a,b] in exp into 1/(ScalarProduct[a,a]-b^2) and replaces FeynAmpDenominator by Identity.

See also: FeynAmpDenominator, PropagatorDenominator.

Examples

```
In[1633]:= PropagatorDenominator[p]
Out[1633]=  1
           ---
            2
           p

In[1634]:= t1 = PropagatorDenominator[q, m]
Out[1634]=    1
           -------
            2    2
           q  - m

In[1635]:= StandardForm[FCI[t1]]
Out[1635]= PropagatorDenominator[Momentum[q, D], m]
```

QGV

Description

QGV is equivalent to QuarkGluonVertex.
See also: QuarkGluonVertex.

QO

Description

QO is equivalent to Twist2QuarkOperator.
See also: Twist2QuarkOperator.

QP

Description

QP is an alias for QuarkPropagator. QP[p] is the massless quark propagator. QP[{p,m}] gives the quark propagator with mass m.
See also: QuarkPropagator.

QuantumField

Description

QuantumField is the head of quantized fields and their derivatives. QuantumField[par, ftype, {lorind}, {sunind}] denotes a quantum field of type ftype with (possible) Lorentz-indices lorind and SU(*N*) indices sunind. The optional first argument par denotes a partial derivative acting on the field.
See also: FeynRule, PartialD, ExpandPartialD.

Examples

This denotes a scalar field.

```
In[1636]:= StandardForm[ChangeDimension[t1, D]]
Out[1636]= PropagatorDenominator[Momentum[q, D], m]

In[1637]:= PropagatorDenominatorExplicit[t1]
Out[1637]=      1
           ---------
            2    2
           q  - m
```

```
In[1638]:= StandardForm[%]
Out[1638]= 1/(-m^2 + Pair[Momentum[q], Momentum[q]])
```

This is a field with a Lorentz index.

```
In[1639]:= Clear[t1]
Out[1639]= FAD[{q, m}, {q - p, 0}]//FCI
```

Color indices should be put after the Lorentz ones.

```
In[1640]:= 1/((q^2 - m^2) . (q - p)^2)
Out[1640]= PropagatorDenominatorExplicit[%]//FCE
```

```
In[1641]:= 1/((q^2 - m^2) (m^2 - 2 p · q + q^2))
Out[1641]= %//StandardForm
```

$\frac{1}{(-m^2+\text{SPD}[q,q])\,(m^2-2\,\text{SPD}[p,q]+\text{SPD}[q,q])}$ is a short form for QuantumField[S]

```
In[1642]:= S
Out[1642]= QuantumField[AntiQuarkField]
```

The first list of indices is usually interpreted as type LorentzIndex, except for OPEDelta, which gets converted to type Momentum.

```
In[1643]:= ψ̄
Out[1643]= QuantumField[QuarkField]
```

Derivatives of fields are denoted as follows.

```
In[1644]:= ψ
Out[1644]= QuantumField[B, {μ}]
```

```
In[1645]:= B_μ
Out[1645]= QuantumField[GaugeField, {μ}, {a}]
```

```
In[1646]:= A_μ^a
Out[1646]= %//StandardForm
```

```
In[1647]:= QuantumField[GaugeField, LorentzIndex[μ], SUNIndex[a]]
Out[1647]= A_Δ^a
```

```
In[1648]:= Δ^μ A_μ^a .
Out[1648]= QuantumField[A, {OPEDelta}, {a}]
```

QuarkField

Description

QuarkField is the name of a fermionic field. QuarkField is just a name with no functional properties. Only typesetting rules are attached.

See also: AntiQuarkField, QuantumField.

Examples

```
In[1649]:= A^a_Δ
Out[1649]= QuantumField[A, {OPEDelta}, {a}]//StandardForm
```

QuarkGluonVertex

Description

QuarkGluonVertex[QuantumField[*A*, LorentzIndex[OPEDelta], SUNIndex[*a*]]] gives the Feynman rule for the quark-gluon vertex.

```
In[1650]:= QuantumField[PartialD[μ], A, {μ}]
Out[1650]= ∂_μ A_μ
```

See also: GluonVertex.

Examples

```
In[1651]:= QuantumField[PartialD[OPEDelta], S]
Out[1651]= ∂_Δ S

In[1652]:= QuantumField[PartialD[OPEDelta], A, {OPEDelta}, {a}]
Out[1652]= ∂_Δ A^a_Δ

In[1653]:= QuantumField[PartialD[OPEDelta]^OPEm, A, {OPEDelta}, {a}]
Out[1653]= ∂^m_Δ A^a_Δ

In[1654]:= QuantumField[QuantumField[A]] === QuantumField[A]
Out[1654]= True

In[1655]:= QuarkField
Out[1655]= ψ

In[1656]:= μ, a
Out[1656]= Options[QuarkGluonVertex]
```

QuarkMass

Description

QuarkMass is an option of Amplitude.

See also: Amplitude.

QuarkPropagator

Description

QuarkPropagator[p] is the massless quark propagator. QuarkPropagator[{p,m}] or gives the quark propagator with mass m.

```
In[1657]:= {CounterTerm → False, CouplingConstant → g_s, Dimension → D, Explicit → False, Ω →
False, Polarization → 0}
Out[1657]= QuarkGluonVertex[μ, a]
```

See also: GluonPropagator, QuarkGluonVertex.

Examples

```
In[1658]:= Q_a^μ
Out[1658]= QuarkGluonVertex[μ, a, CounterTerm → 1]

In[1659]:= Q_a^μ
Out[1659]= QuarkGluonVertex[μ, a, CounterTerm → 2]

In[1660]:= Q_a^μ
Out[1660]= QuarkGluonVertex[μ, a, CounterTerm → 3]
```

ReduceGamma

Description

ReduceGamma is an option of OneLoop. If set to True all DiracMatrix[6] and DiracMatrix[7] (i.e. all ChiralityProjector) are reduced to Gamma5.

See also: OneLoop.

ReduceToScalars

Description

ReduceToScalars is an option for OneLoop and OneLoopSum that specifies whether the result will be reduced to scalar A0, B0, C0 and D0 scalar integrals.

See also: OneLoop, OneLoopSum.

Rename

Description

Rename is an option for Contract. If set to True, dummy indices in Eps are renamed, using $MU[i].
See also: Contract.

RHI ***unfinished***

Description

RHI[{v,w,x,y,z},{a,b,c,d,e,f,g}, {al,be,ga,de,ep}]. (sn → 1, mark1 → 1, mark2 → 1, mark3 → 1, eph → Epsilon/2). The exponents of the numerator scalar product are (dl = OPEDelta):
v: k1.k1, w: k2.k2, x: p.k1, y: p.k2, z: k1.k2.
a: dl.k1, b: dl.k2, c: dl.(p-k1), d: dl.(p-k2), e: dl.(k1-k2), f: dl.(p+k1-k2), g: dl.(p-k1-k2)
RHI[any___,{a,b,c,d,e,0,0}, {al,be,ga,de,ep}] simplifies to RHI[any, {a,b,c,d,e}, {al,be,ga,de,ep}];
RHI[{0,0,0,0,0},{a,b,c,d,e}, {al,be,ga,de,ep}] simplifies to RHI[{a,b,c,d,e}, {al,be,ga,de,ep}].
See also: RHI2FC.

Examples

```
In[1661]:= Q_a^μ
```

RHI2FC ***unfinished***

Description

RHI[{v,w,x,y,z},{a,b,c,d,e,f,g}, {al,be,ga,de,ep}]. (sn → 1, mark1 → 1, mark2 → 1, mark3 → 1, eph → Epsilon/2). The exponents of the numerator scalar product are (dl = OPEDelta):
v: k1.k1, w: k2.k2, x: p.k1, y: p.k2, z: k1.k2.
a: dl.k1, b: dl.k2, c: dl.(p-k1), d: dl.(p-k2), e: dl.(k1-k2), f: dl.(p+k1-k2), g: dl.(p-k1-k2)
RHI[any___,{a,b,c,d,e,0,0}, {al,be,ga,de,ep}] simplifies to RHI[any, {a,b,c,d,e}, {al,be,ga,de,ep}];
RHI[{0,0,0,0,0},{a,b,c,d,e}, {al,be,ga,de,ep}] simplifies to RHI[{a,b,c,d,e}, {al,be,ga,de,ep}].
See also: RHI2FC.

Examples

```
In[1662]:= QuarkGluonVertex[{p, μ, a}, {q}, {k}, OPE → True]
```

RHM ***unfinished***

Description

RHM[] is like RHI[], gives Gamma functions.
See also: RHI.

Examples

```
In[1663]:= QuarkGluonVertex({p, μ, a}, {q}, {k}, Ω → True)
```

RHO ***unfinished***

Description

RHO[i] with i from 1 to 4 is an abbreviation for the 4 operators (eq. (3.2.17) – (3.2.20)) defined in R.Hambergs thesis. The Lorentz indices are suppressed. The explicit expressions may be recovered by RHO[i, mu, nu, p].
See also: RHI.

Examples

```
In[1664]:= QuarkGluonVertex[{p, μ, a}, {q}, {k}, OPE → False]
```

RHP ***unfinished***

Description

RHP[i, mu, nu, p] with i from 1 to 4 gives the projectors for RHO[i]. RHP[mu, nu, p] gives Sum[RHP[i,mu, nu, p] RHO[i], {i, 4}] collected with respect to mu and nu.
See also: RHO.

Examples

```
In[1665]:= QuarkGluonVertex({p, μ, a}, {q}, {k}, Ω → False)
```

RTL ***unfinished***

Description

RTL[exp] inserts the list of known TLI integrals into exp, substitutes D→4+Epsilon and expands around Epsilon up to the finite part.

See also: TLI, Epsilon.

Examples

```
In[1666]:= Options[QuarkPropagator]
```

RightPartialD

Description

RightPartialD[{CounterTerm → False, CouplingConstant → g_s, Dimension → D, Explicit → True, Loop → 0, Ω → False, Polarization → 0}] denotes QuarkPropagator[p
acting to the right.
See also: ExpandPartialD, PartialD, LeftPartialD.

Examples

```
In[1667]:= i γ · p / p^2
Out[1667]= QuarkPropagator[{p, m}]

In[1668]:= i (m + γ · p) / (p^2 - m^2)
Out[1668]= QuarkPropagator[p, 1, 2]

In[1669]:= i γ · p / p^2
Out[1669]= {}
Out[1669]= {}
Out[1669]= {}
```

RussianTrick

Description

RussianTrick[exp, k, {q1,q2,p}] (=RussianTrick[exp,p,p,{q1,q2,p}]) does the integration by parts where p is the external momentum. RussianTrick[exp, k,l, {q1,q2,p}] (=RussianTrick[exp,k,l]) does integration by parts where l is the momentum to be differentiated.

The result is an expression which is vanishing.

See also: FourDivergence, FourLaplacian.

Examples

```
Out[1669]= {}
Out[1669]= {}
Out[1669]= {}
Out[1669]= μ

In[1670]:= ∂_μ
Out[1670]= RightPartialD[μ]

In[1671]:= (→∂)_μ
```

ScalarGluonVertex

Description

ScalarGluonVertex[{p}, {q}, {mu,a}] or ScalarGluonVertex[p, q, mu, a] yields the scalar-scalar-gluon vertex (p and q are incoming momenta).
ScalarGluonVertex[{mu,a}, {nu,b}] yields the scalar-scalar-gluon-gluon vertex (p and q are incoming momenta). The dimension and the name of the coupling constant are determined by the options Dimension and CouplingConstant.

```
In[1672]:= RightPartialD[μ].QuantumField[A, LorentzIndex[μ]]
Out[1672]= (→∂)_μ.A_μ
```

Examples

```
In[1673]:= ExpandPartialD[%]
Out[1673]= ∂_μA_μ
```

ScalarProduct

Description

ScalarProduct[p, q] is the input for scalar product. ScalarProduct[p] is equivalent to ScalarProduct[p, p]. Expansion of sums of momenta in ScalarProduct is done with ExpandScalarProduct. Scalar product may be set, e.g., ScalarProduct[a, b] = %//StandardForm; but a and b must not contain sums. Note that ScalarProduct[a, b] = QuantumField[PartialD[LorentzIndex[μ]], *A*, LorentzIndex[μ]] actually sets also: Pair[Momentum[a, ___], Momentum[b, ___]] = RightPartialD[μ]//StandardForm. It is encouraged to always set ScalarProduct's **before** any calculation. This improves the performance of FeynCalc .

See also: Calc, ClearScalarProducts, ExpandScalarProduct, ScalarProductCancel, Pair, SP, SPD.

Examples

```
In[1674]:= RightPartialD[LorentzIndex[μ]]
Out[1674]= t = RHI[{OPEm, 0, 0, 0, 0}, {1, 1, 1, 1, 1}]

In[1675]:= T^m0000_11111
Out[1675]= t//RHI2FC

In[1676]:= (Δ · q1)^m / (q1^2 · q2^2 · (q1 - p)^2 · (q2 - p)^2 · (q1 - q2)^2)
Out[1676]= RussianTrick[%//RHI2FC, q2]

In[1677]:= -(Δ · q1)^m / (q1^2 · (q1 - p)^2 · (q2 - p)^2 · (q2 - q1)^2 · (q2 - q1)^2) -
           (Δ · q1)^m / (q1^2 · (q1 - p)^2 · (q2 - q1)^2 · (q2 - p)^2 · (q2 - p)^2) - (4 - D) (Δ · q1)^m / (q1^2 · q2^2 · (q1 - p)^2 · (q2 - q1)^2 · (q2 - p)^2) +
           (Δ · q1)^m / (q2^2 · (q1 - p)^2 · (q2 - p)^2 · (q2 - q1)^2 · (q2 - q1)^2) + (Δ · q1)^m p^2 / (q2^2 · q1^2 · (q2 - q1)^2 · (q1 - p)^2 · (q2 - p)^2 · (q2 - p)^2)
Out[1677]= FC2RHI[%]

In[1678]:= T^m0000_01112 - T^m0000_10112 - T^m0000_10121 + (D - 4) T^m0000_11111 + p^2 T^m0000_11121
Out[1678]= Solve2[%, t]

In[1679]:= {T^m0000_11111 → (T^m0000_01112 - T^m0000_10112 - T^m0000_10121 + p^2 T^m0000_11121) / (4 - D)}
Out[1679]= Clear[t]

In[1680]:= Options[ScalarGluonVertex]
Out[1680]= {Dimension → D, CouplingConstant → g_s}

In[1681]:= ScalarGluonVertex[{p}, {q}, {μ, a}]
Out[1681]= i g_s (p - q)^μ T_a

In[1682]:= m^2
Out[1682]= m^2

In[1683]:= m^2
```

ScalarProductCancel

Description

ScalarProductCancel[exp, q1, q2, ...] cancels scalar products with propagators. ScalarProductCancel[exp] cancels simple cases.

```
In[1684]:= ScalarProduct[p, q]
Out[1684]= p · q

In[1685]:= ScalarProduct[p + q, -q]
Out[1685]= -(q · (p + q))
```

```
In[1686]:= ScalarProduct[p, p]
```

p^2

See also: Calc, ClearScalarProducts, ExpandScalarProduct, Pair, SP, SPD.

Examples

See also: FeynAmpDenominatorSimplify.

```
In[1687]:= ScalarProduct[q]
Out[1687]= q^2

In[1688]:= ScalarProduct[p, q]//StandardForm
Out[1688]= Pair[Momentum[p], Momentum[q]]

In[1689]:= ScalarProduct[p, q, Dimension → D]//StandardForm
Out[1689]= Pair[Momentum[p, D], Momentum[q, D]]

In[1690]:= ScalarProduct[p1, p2] = s/2
Out[1690]= s/2

In[1691]:= ExpandScalarProduct[ ScalarProduct[p1 - q, p2 - k]]
Out[1691]= s/2 + k · q - k · p1 - q · p2

In[1692]:= Calc[ ScalarProduct[p1 - q, p2 - k]]
Out[1692]= s/2 + k · q - k · p1 - q · p2

In[1693]:= ClearScalarProducts
```

ScaleMu

Description

ScaleMu is the mass scale used for dimensional regularization of loop integrals.

```
In[1694]:= SPC
Out[1694]= ScalarProductCancel
```

Schouten

Description

Schouten[expr] applies the Schouten identity on at most 42 terms in a sum. If Schouten should operate on larger expression you can give a second argument, e.g.: Schouten[expr, 4711] which will work on sums with less than 4711 terms.

Schouten is also an option of Contract and DiracTrace. It may be set to an integer indicating the maximum number of terms onto which the function Schouten will be applied .

See also: Contract, DiracTrace.

Examples

```
In[1695]:= ScalarProductCancel//Options
Out[1695]= {ChangeDimension → D, Collecting → True,

             FeynAmpDenominatorSimplify → False, FeynAmpDenominatorCombine → True}

In[1696]:= ?SPC
Out[1696]= SPC is an abbreviation for ScalarProductCancel.

In[1697]:= t1 = SPD[q, p] FAD[{q, m}, {q - p, 0}]//FCI
```

SD

Description

SD[i, j] is the (FeynCalc-external) Kronecker-delta for SU(*N*) with color indices i and j. SD[i,j] is transformed into SUNDelta[SUNIndex[i],SUNIndex[j]] by FeynCalcInternal.

See also: SUNDelta.

Examples

```
In[1698]:= p · q / ((q^2 - m^2) . (q - p)^2)
Out[1698]= ScalarProductCancel[t1, q]

In[1699]:= -1/(2 (q^2 - m^2)) + 1/(2 (q - p)^2) + (m^2/2 + p^2/2)/((q^2 - m^2) . (q - p)^2)
Out[1699]= FeynAmpDenominatorSimplify[%, q]

In[1700]:= (m^2/2 + p^2/2)/((q^2 - m^2) . (q - p)^2) - 1/(2 (q^2 - m^2))
Out[1700]= SPC[t1, q, FDS → True]
```

SelectFree

Description

SelectFree[expr, a, b, ...] is equivalent to Select[expr, FreeQ2[#, {a,b, ...}]&], except the special cases: SelectFree[a, b] returns a and SelectFree[a,a] returns 1 (where a is not a product or a sum).

SelectFree is equivalent to Select1.

See also: FreeQ2, SelectNotFree.

Examples

```
In[1701]:= (m^2/2 + p^2/2)/((q^2 - m^2).(q - p)^2) - 1/(2 (q^2 - m^2))
Out[1701]= t2 = SPD[q2, p]SPD[q1, p] FAD[{q1, m}, {q2, m}, q1 - p, q2 - p, q2 - q1]//FCI

In[1702]:= p.q1 p.q2/((q1^2 - m^2).(q2^2 - m^2).(q1 - p)^2.(q2 - p)^2.(q2 - q1)^2)
Out[1702]= SPC[t2, q1, q2, FDS → True]

In[1703]:= 1/(4 (q1^2 - m^2).(q2^2 - m^2).(q2 - q1)^2) - 1/(2 (q1^2 - m^2).(q1 - p)^2.(q2 - q1)^2) +
  (-m^2/2 - p^2/2)/((q1^2 - m^2).(q2^2 - m^2).(q1 - p)^2.(q2 - q1)^2) + (m^2/2 + p^2/2)/(q1^2.q2^2.((q1 - p)^2 - m^2).(q2 - q1)^2) +
  (m^4/4 + p^2 m^2/2 + p^4/4)/((q1^2 - m^2).(q2^2 - m^2).(q1 - p)^2.(q2 - p)^2.(q2 - q1)^2)
Out[1703]= Clear[t1, t2];

In[1704]:= ScaleMu
Out[1704]= μ

In[1705]:= t = Sum[∑_{a=1}^{4} (SP[k, q[a]]
    ((1/6) Eps[LorentzIndex[a], LorentzIndex[b], LorentzIndex[c], LorentzIndex[d]]
      Eps[LorentzIndex[μ], Momentum[q[b]], Momentum[q[c]], Momentum[q[d]]])),
  {b, 1, 4}, {c, 1, 4}, {d, 1, 4}] - Eps[Momentum[q[1]], Momentum[q[2]],
  Momentum[q[3]], Momentum[q[4]]] * Pair[LorentzIndex[μ], Momentum[k]]
Out[1705]= -ϵ^{q(1)q(2)q(3)q(4)} k^μ + (1/6) ϵ^{μq(2)q(3)q(4)} k.q(1) - (1/6) ϵ^{μq(2)q(4)q(3)} k.q(1) -
  (1/6) ϵ^{μq(3)q(2)q(4)} k.q(1) + (1/6) ϵ^{μq(3)q(4)q(2)} k.q(1) + (1/6) ϵ^{μq(4)q(2)q(3)} k.q(1) -
  (1/6) ϵ^{μq(4)q(3)q(2)} k.q(1) - (1/6) ϵ^{μq(1)q(3)q(4)} k.q(2) + (1/6) ϵ^{μq(1)q(4)q(3)} k.q(2) +
  (1/6) ϵ^{μq(3)q(1)q(4)} k.q(2) - (1/6) ϵ^{μq(3)q(4)q(1)} k.q(2) - (1/6) ϵ^{μq(4)q(1)q(3)} k.q(2) + (1/6) ϵ^{μq(4)q(3)q(1)} k.q(2) +
  (1/6) ϵ^{μq(1)q(2)q(4)} k.q(3) - (1/6) ϵ^{μq(1)q(4)q(2)} k.q(3) - (1/6) ϵ^{μq(2)q(1)q(4)} k.q(3) + (1/6) ϵ^{μq(2)q(4)q(1)} k.q(3) +
  (1/6) ϵ^{μq(4)q(1)q(2)} k.q(3) - (1/6) ϵ^{μq(4)q(2)q(1)} k.q(3) - (1/6) ϵ^{μq(1)q(2)q(3)} k.q(4) + (1/6) ϵ^{μq(1)q(3)q(2)} k.q(4) +
  (1/6) ϵ^{μq(2)q(1)q(3)} k.q(4) - (1/6) ϵ^{μq(2)q(3)q(1)} k.q(4) - (1/6) ϵ^{μq(3)q(1)q(2)} k.q(4) + (1/6) ϵ^{μq(3)q(2)q(1)} k.q(4)

In[1706]:= t//Schouten
```

```
Out[1706]= 0

In[1707]:= Clear[t]
Out[1707]= SD[a, b]
```

SelectGraphs

Description

SelectGraphs is an option for OneLoopSum indicating that only a slected set of graphs of the list provided to OneLoopSum is to be calculated. Possible settings are: SelectGraphs→{ i, j, ... } or SelectGraphs→{ a, {b, c}, ... } which indicates the graphs to be taken from the list provided to OneLoopSum. In the second setting the list {b, c} indicates that all amplitudes from b to c should be taken.

See also: OneLoopSum.

SelectNotFree

Description

SelectNotFree[expr, x] returns that part of expr which is not free of any occurance of x.

SelectNotFree[expr, a, b, ...] is equivalent to Select[expr, !FreeQ2[#, {a, b, ...}]&], except the special cases: SelectNotFree[a, b] returns 1 and SelectNotFree[a, a] returns a (where a is not a product or a sum).

SelectNotFree is equivalent to Select2.

See also: FreeQ2, SelectFree.

Examples

```
In[1708]:= δab
Out[1708]= %//FCI//StandardForm

In[1709]:= SUNDelta[SUNIndex[a], SUNIndex[b]]
Out[1709]= %//FCE//StandardForm

In[1710]:= SD[a, b]
Out[1710]= SelectFree[a + b + f[a] + d, a]

In[1711]:= b + d
Out[1711]= SelectFree[x y, x]

In[1712]:= y
Out[1712]= SelectFree[2 x y z f[x], {x, y}]

In[1713]:= 2 z
Out[1713]= SelectFree[a, b]
```

```
In[1714]:= a
Out[1714]= SelectFree[a, a]
```

SelectSplit

Description

SelectSplit[l, p] Construct list of mutually exclusive subsets from l in which every element li satisfies a criterium pj[li] with pj from p and appends the subset of remaining unmatched elements.

Examples

```
In[1715]:= 1
Out[1715]= SelectFree[1, c]

In[1716]:= 1
Out[1716]= SelectFree[f[x], x]
```

Select1

Description

Select1[expr, a, b, ...] is equivalent to Select[expr, FreeQ2[#, {a,b, ...}]&], except the special cases: Select1[a, b] returns a and Select1[a,a] returns 1 (where a is not a product or a sum).

See also: Select2.

Select2

Description

Select2[expr, a, b, ...] is equivalent to Select[expr, !FreeQ2[#, {a,b, ...}]&], except the special cases: Select2[a, b] returns 1 and Select2[a,a] returns a (where a is not a product or a sum).

See also: Select1.

Series2

Description

Series2 performs a series expansion around 0. Series2 is (up to the Gamma-bug) equivalent to Series, except that it applies Normal on the result and has an option FinalSubstitutions. Series2[f, e, n] is equivalent to Series2[f, {e, 0, n}].

```
In[1717]:= 1
Out[1717]= SelectNotFree[a + b + f[a], a]
```

Examples

```
In[1718]:= a + f(a)
Out[1718]= SelectNotFree[2 x y f[x] z, {x, y}]

In[1719]:= x y f(x)
Out[1719]= SelectNotFree[a, b]

In[1720]:= 1
Out[1720]= SelectNotFree[a + x, b]

In[1721]:= 0
Out[1721]= SelectNotFree[a, a]

In[1722]:= a
Out[1722]= SelectNotFree[1, c]
```

In earlier *Mathematica* versions the Gamma functions was not expanded far enough, in 5.0 this got finally fixed:

```
In[1723]:= 1
Out[1723]= SelectNotFree[f[x], x]
```

There is a table of expansions of special hypergeometric functions.

```
In[1724]:= f(x)
Out[1724]= SelectSplit[{a^2, b^3, c^4, d^5, e^6, f + g, h^4},
             {MatchQ[#, _^2]&, MatchQ[#, _^4]&, FreeQ[#, Power]&}]

In[1725]:= {{a^2}, {c^4, h^4}, {f + g}, {b^3, d^5, e^6}}
Out[1725]= SelectSplit[{a^2, b^3, c^4, d^5, e^6, f + g, h^4}, {FreeQ[#, Plus]&, FreeQ[#, Power]&}]

In[1726]:= {{a^2, b^3, c^4, d^5, e^6, h^4}, {f + g}, {}}
Out[1726]= Options[Series2]

In[1727]:= {Collecting → False, Factoring → True, FinalSubstitutions → {Γ_E → 0}, Print → False}
Out[1727]= Series2[(x (1 - x))^(δ/2), δ, 1]
```

There are over 100 more special expansions of $\frac{1}{2}\,\delta\,\log((1-x)\,x)+1$ tabulated in Series2.m. The interested user can consult the source code (search for HYPERLIST).

Series3

Description

Series3 performs a series expansion around 0. Series3 is equivalent to Series, except that it applies Normal on the result and that some Series bugs are fixed. Series3[f, e, n] is equivalent to Series3[f, {e, 0, n}].

```
In[1728]:= Series2[Gamma[x], x, 1]
Out[1728]= π^2 x/12 + 1/x
```

See also: Series2.

Examples

```
In[1729]:= Series[Gamma[x], {x, 0, 1}]
Out[1729]= 1/x - Γ_E + 1/2 (Γ_E^2 + π^2/6) x + O(x^2)

In[1730]:= Series2[Gamma[x], x, 2]
Out[1730]= -1/3 ζ(3) x^2 + π^2 x/12 + 1/x
```

SetMandelstam

Description

SetMandelstam[s, t, u, p1, p2, p3, p4, m1, m2, m3, m4] defines the Mandelstam variables s=(p1+p2)^2, t=(p1+p3)^2, u=(p1+p4)^2 and sets the pi on-shell: p1^2=m1^2, p2^2=m2^2, p3^2=m3^2, p4^2=m4^2. Note that p1 + p2 + p3 + p4 = 0 is assumed.
SetMandelstam[x, {p1, p2, p3, p4, p5}, {m1, m2, m3, m4, m5}] defines x[i, j] = (pi+pj)^2 and sets the pi on-shell. The pi satisfy: p1 + p2 + p3 + p4 + p5 = 0.

```
In[1731]:= Series2[Gamma[x], x, 2, FinalSubstitutions → {}]//FullSimplify
Out[1731]= 1/12 (-2 Γ_E^3 x^2 + (π^2 - 4 x ζ(3)) x + 6 Γ_E^2 x - Γ_E (π^2 x^2 + 12) + 12/x)
```

See also: Mandelstam.

SimplifyDeltaFunction

Description

SimplifyDeltaFunction[exp, x] simplifies f[x]*DeltaFunction[1-x] toLimit[f[x],x→1] DeltaFunction[1-x] and applies a list of transformation rules for DeltaFunctionPrime[1-x]*x^(OPEm-1)*f[x] where x^(OPEm-1) is suppressed in exp.

See also: DeltaFunction, DeltaFunctionPrime.

Examples

```
In[1732]:= Series[Gamma[x], {x, 0, If[$VersionNumber < 5, 4, 2]}]//Normal//Expand//
             FullSimplify
```

$$\text{Out[1732]}= \frac{1}{12}\left(-2\,\Gamma_E^{\,3}\,x^2+(\pi^2-4\,x\,\zeta(3))\,x+6\,\Gamma_E^{\,2}\,x-\Gamma_E\,(\pi^2\,x^2+12)+\frac{12}{x}\right)$$

```
In[1733]:= Series2[HypergeometricPFQ[{1, OPEm - 1, Epsilon/2 + OPEm}, {OPEm, OPEm + Epsilon}, 1],
             {Epsilon, 0, 1}]
```

$$\text{Out[1733]}= \frac{2\,m-2}{\varepsilon}+\frac{1}{2}\,\varepsilon\,(m-1)\,\psi^{(1)}(m)+1$$

```
In[1734]:= Series2[HypergeometricPFQ[{1, OPEm, Epsilon/2 + OPEm},
             {1 + OPEm, Epsilon + OPEm}, 1], {Epsilon, 0, 1}]
```

$$\text{Out[1734]}= \frac{1}{4}\,\varepsilon\,m\,\psi^{(0)}(m)^2+\frac{2\,m}{\varepsilon}+\frac{3}{4}\,\varepsilon\,m\,\psi^{(1)}(m)-\frac{1}{2}\,\varepsilon\,m\,S_{11}(m-1)+\frac{1}{24}\,\varepsilon\,m\,\pi^2$$

```
In[1735]:= Hypergeometric2F1[1, Epsilon, 1 + 2 Epsilon, x]
Out[1735]= 2F1(1, ε; 2 ε + 1; x)
```

```
In[1736]:= Series2[%, {Epsilon, 0, 3}]
```

$$\text{Out[1736]}= \left(-\frac{1}{6}\,\text{Log}^3(1-x)+\text{Log}(x)\,\text{Log}^2(1-x)-\zeta(2)\,\text{Log}(1-x)-\frac{1}{6}\,\pi^2\,\text{Log}(1-x)+2\,\text{Li}_3(1-x)+4\,\text{Li}_3(x)-2\,\zeta(3\right.$$
$$\left.\left(-\frac{1}{2}\,\text{Log}^2(1-x)+2\,\text{Log}(x)\,\text{Log}(1-x)-3\,\zeta(2)+2\,\text{Li}_2(1-x)+\frac{\pi^2}{6}\right)\varepsilon^2-\text{Log}(1-x)\,\varepsilon+1\right.$$

```
In[1737]:= 2F1
Out[1737]= Options[Series3]
```

```
In[1738]:= {Factoring → True, FinalSubstitutions → {}}
Out[1738]= Series3[(x (1 - x))^(δ/2), δ, 1]
```

$$\text{In[1739]:=}\ \frac{1}{2}\,\delta\,\text{Log}((1-x)\,x)+1$$

```
Out[1739]= Series3[Gamma[x], x, 1]//FullSimplify
```

$$\text{In[1740]:=}\ -\Gamma_E+1+\frac{1}{x}$$

```
Out[1740]= Options[SetMandelstam]
```

```
In[1741]:= {Dimension → {4, D, ᗞ}}
Out[1741]= g[x] DeltaFunction[1 - x]
```

$$\text{In[1742]:=}\ \delta(1-x)\,g(x)$$

```
Out[1742]= SimplifyDeltaFunction[ %, x]
```

$$\text{In[1743]:=}\ \delta(1-x)\,\left(\underline{\text{lim}}\,g(x)\right)$$

```
Out[1743]= g[x]DeltaFunctionPrime[1 - x]
```

SimplifyGTI ***unfinished***

Description

SimplifyGTI simplifies GTI's.
See also: GTI.

Examples

```
In[1744]:= δ'(1 - x) g(x)
```

SimplifyPolyLog

Description

SimplifyPolyLog[y] performs several simplifications assuming that the variables occuring in the Log and PolyLog functions are between 0 and 1.
See also: Nielsen.

Examples

```
In[1745]:= SimplifyDeltaFunction[ %, x]

In[1746]:= δ'(1 - x) (lim g(x)) + δ(1 - x) (lim g'(x))
Out[1746]= x Log[x] DeltaFunctionPrime[1 - x]

In[1747]:= x δ'(1 - x) Log(x)
Out[1747]= SimplifyDeltaFunction[ %, x]

In[1748]:= δ(1 - x)
Out[1748]= PolyLog[2, 1 - x] DeltaFunctionPrime[1 - x]

In[1749]:= δ'(1 - x) Li2(1 - x)
Out[1749]= SimplifyDeltaFunction[ %, x]

In[1750]:= -δ(1 - x)
Out[1750]= Log[x]PolyLog[2, 1 - x] DeltaFunctionPrime[1 - x]

In[1751]:= δ'(1 - x) Log(x) Li2(1 - x)
Out[1751]= SimplifyDeltaFunction[ %, x]

In[1752]:= 0
Out[1752]= PolyLog[3, 1 - x] DeltaFunctionPrime[1 - x]

In[1753]:= δ'(1 - x) Li3(1 - x)
Out[1753]= SimplifyDeltaFunction[ %, x]
```

```
In[1754]:= -δ(1 - x)
Out[1754]= {}
Out[1754]= sip[y_] := y == SimplifyPolyLog[y]

In[1755]:= sip[PolyLog[2, 1/x]]
```

$$\text{Out[1755]= } \text{Li}_2\left(\frac{1}{x}\right) == -\frac{1}{2}\text{Log}^2(x) + \text{Log}(1-x)\ \text{Log}(x) + i\pi\ \text{Log}(x) + \zeta(2) + \text{Li}_2(1-x)$$

```
In[1756]:= sip[PolyLog[2, x]]
```

$$\text{Out[1756]= } \text{Li}_2(x) == \zeta(2) - \text{Log}(1-x)\ \text{Log}(x) - \text{Li}_2(1-x)$$

```
In[1757]:= sip[PolyLog[2, 1 - x^2]]
```

$$\text{Out[1757]= } \text{Li}_2(1-x^2) == -\zeta(2) - 2\ \text{Log}(x)\ \text{Log}(x+1) + 2\ \text{Li}_2(1-x) - 2\ \text{Li}_2(-x)$$

```
In[1758]:= sip[PolyLog[2, x^2]]
```

$$\text{Out[1758]= } \text{Li}_2(x^2) == 2\zeta(2) - 2\ \text{Log}(1-x)\ \text{Log}(x) - 2\ \text{Li}_2(1-x) + 2\ \text{Li}_2(-x)$$

```
In[1759]:= sip[PolyLog[2, -x/(1 - x)]]
```

$$\text{Out[1759]= } \text{Li}_2\left(-\frac{x}{1-x}\right) == -\frac{1}{2}\text{Log}^2(1-x) + \text{Log}(x)\ \text{Log}(1-x) - \zeta(2) + \text{Li}_2(1-x)$$

```
In[1760]:= sip[PolyLog[2, x/(x - 1)]]
```

$$\text{Out[1760]= } \text{Li}_2\left(\frac{x}{x-1}\right) == -\frac{1}{2}\text{Log}^2(1-x) + \text{Log}(x)\ \text{Log}(1-x) - \zeta(2) + \text{Li}_2(1-x)$$

```
In[1761]:= sip[Nielsen[1, 2, -x/(1 - x)]]
```

$$\text{Out[1761]= } S_{12}\left(-\frac{x}{1-x}\right) == S_{12}(x) - \frac{1}{6}\text{Log}^3(1-x)$$

```
In[1762]:= sip[PolyLog[3, -1/x]]
```

$$\text{Out[1762]= } \text{Li}_3\left(-\frac{1}{x}\right) == \frac{\text{Log}^3(x)}{6} + \zeta(2)\ \text{Log}(x) + \text{Li}_3(-x)$$

```
In[1763]:= sip[PolyLog[3, 1 - x]]
```

Simplify2 ***unfinished***

Description

Simplify2 is a special ordering function.

Examples

```
In[1764]:= True
```

SmallDelta

Description

SmallDelta denotes some small positive number.

SmallEpsilon

Description

SmallEpsilon denotes some small positive number.

SmallVariable

Description

SmallVariable[me] is the head of small (negligible) variables. This means any mass with this head can be neglected if it appears in a sum, but not as an argument of Passarino-Veltman (PaVe) functions or PropagatorDenominator.

See also: PaVe, PaVeReduce, PropagatorDenominator.

SmallVariables

Description

SmallVariables is an option for OneLoop. "SmallVariables→{Melectron}" i.e. will substitute "SmallVariable[Melectron]" for all Melectron's in the calculation.

See also: OneLoop.

SMP

Description

SMP[par] substitutes a symbol for the Standard Model parameter par. SMP[] gives the list of substitutions. par should be a string; e.g., SMP["SW"] gives sw (in the Global‘ context).

```
In[1765]:= sip[PolyLog[3,x^2]]
Out[1765]= Li3(x^2) == -2 Log(1-x) Log^2(x)+4ζ(2) Log(x)-4Li2(1-x) Log(x)-4S12(1-x)+4Li3(-x)+4ζ(3)
```

See also: SMVertex.

Examples

```
In[1766]:= sip[PolyLog[3,-x/(1-x)]]
```

$$\text{Out[1766]= } \text{Li}_3\left(-\frac{x}{1-x}\right) == \frac{1}{6}\text{Log}^3(1-x) - \frac{1}{2}\text{Log}(x)\,\text{Log}^2(1-x) + \frac{1}{2}\text{Log}^2(x)\,\text{Log}(1-x) + \zeta(2)\,\text{Log}(1-x) - \zeta(2)\,\text{Log}(x) + \text{S}_{12}(1-x) + \text{Log}(x)\,\text{Li}_2(1-x) - \text{Li}_3(1-x)$$

SMVertex

Description

SMVertex["AWW", p,mu, q,nu, k,rho] gives the photon-W-W vertex (p,mu correspond to the photon, q,nu to the (incoming) W+ and k,rho to the (incoming) W-. All momenta are flowing into the vertex. SMVertex["HHH", ___] give the three-higgs coupling.

```
In[1767]:= sip[PolyLog[3, 1 - 1/x]]
Out[1767]= Li3(1 - 1/x) ==
             Log^3(x)/6 - 1/2 Log^2(1 - x) Log(x) + Li2(1 - x) Log(x) + S12(1 - x) + S12(x) - Log(1 - x) Li2(1 - x) - ζ(3)
```

Examples

```
In[1768]:= sip[PolyLog[4, -x/(1 - x)]]
Out[1768]= Li4(-x/(1 - x)) == -1/24 Log^4(1 - x) + 1/2 Log(x) Log^3(1 - x) - 1/2 Log^2(x) Log^2(1 - x) -
             1/2 ζ(2) Log^2(1 - x) + 1/2 Li2(1 - x) Log^2(1 - x) + ζ(2) Log(x) Log(1 - x) - S12(1 - x) Log(1 - x) -
             S12(x) Log(1 - x) - Log(x) Li2(1 - x) Log(1 - x) + ζ(3) Log(1 - x) - S13(x) + S22(x) - Li4(x)
```

Smu ***unfinished***

Description

Smu is ...

Sn

Description

Sn is sip[log[$a + b/c$]]/log$\left(a + \frac{b}{c}\right)$ == log$\left(\frac{b+a\,c}{c}\right)$.

SO

Description

SO[q] is a four-dimensional scalar product of OPEDelta with q. It is transformed into Pair[Momentum[q], Momentum[OPEDelta]] by FCI.

See also: FCI, OPEDelta, Pair, ScalarProduct, SOD.

Examples

```
In[1769]:= sip[Log[1/x]]
Out[1769]= Log(1/x) == -Log(x)

In[1770]:= sip[ArcTanh[x]]
Out[1770]= Tanh^-1(x) == 1/2 Log(-(x + 1)/(1 - x))

In[1771]:= sip[ArcSinh[x]]
Out[1771]= Sinh^-1(x) == Log(x + Sqrt(x^2 + 1))
```

SOD

Description

SOD[q] is a D-dimensional scalar product of OPEDelta with q. It is transformed into Pair[Momentum[q,D], Momentum[OPEDelta,D]] by FeynCalcInternal.

See also: OPEDelta, Pair, ScalarProduct, SOD.

Examples

```
In[1772]:= sip[ArcCosh[x]]
Out[1772]= Cosh^-1(x) == Log(x + Sqrt(x^2 - 1))

In[1773]:= Clear[sip]
Out[1773]= {}
Out[1773]= SMP[]//InputForm
```

Solve2

Description

Solve2 is equivalent to Solve, except that it works only for linear equations (and returns just a list) and accepts the options Factoring and FinalSubstitutions. Solve2 uses the "high school algorithm" and factors intermediate results. Therefore it can be drastically more useful than Solve.

```
In[1774]:= {EL :> e, CW :> cw, ME :> me, MH :> mh, MW :> mw, SW :> sw}
Out[1774]= SMP["ME"]
```

Examples

```
In[1775]:= me
Out[1775]= SMVertex//Options
```

If no equation sign is given the polynomials are supposed to be 0.

```
In[1776]:= {Dimension → 4, Explicit → True}
Out[1776]= SMVertex["AWW", p, μ, q, ν, k, ρ]

In[1777]:= -i e ((k - q)^μ g^νρ - g^μρ (k - p)^ν - g^μν (p - q)^ρ)
Out[1777]= π^(n/2)

In[1778]:= (2 π)^n
Out[1778]= SO[p]

In[1779]:= Δ · p
Out[1779]= SO[p - q]
```

Solve3

Description

Solve3 is equivalent to Solve, except that it works only for linear equations (and returns just a list) and uses the "high school algorithm" and is sometimes better than Solve for systems involving rational polyonials.

```
In[1780]:= Δ · (p - q)
Out[1780]= SO[p]//FCI//StandardForm
```

See also: Solve2.

Examples

```
In[1781]:= Pair[Momentum[OPEDelta], Momentum[p]]
Out[1781]= SOD[p]
```

SP

Description

SP[a, b] denotes a four-dimensional scalar product. SP[a, b] is transformed into ScalarProduct[a, b] by FeynCalcInternal. SP[p] is the same as SP[p, p] $\Delta \cdot p$).

See also: Calc, ExpandScalarProduct, ScalarProduct.

Examples

```
In[1782]:= SOD[p - q]
Out[1782]= Δ · (p - q)

In[1783]:= SOD[p]//FCI//StandardForm
Out[1783]= Pair[Momentum[OPEDelta, D], Momentum[p, D]]

In[1784]:= Options[Solve2]
Out[1784]= {Factoring → Factor2, FinalSubstitutions → {}}

In[1785]:= Solve2[{2 x == b - w/2, y - d == p}, {x, y}]
Out[1785]= {x → 1/4 (2 b - w), y → d + p}

In[1786]:= Solve2[x + y, x]
Out[1786]= {x → -y}

In[1787]:= Solve2[x + y, x, FinalSubstitutions → {y → h}]
Out[1787]= {x → -h}

In[1788]:= Solve2[{2 x == b - w/2, y - d == p}, {x, y}, Factoring → Expand]
Out[1788]= {x → b/2 - w/4, y → d + p}
```

SPC

Description

SPC is an abbreviation for ScalarProductCancel.

See also: ScalarProductCancel.

SPD

Description

SPD[a, b] denotes a D-dimensional scalar product. SPD[a, b] is transformed into ScalarProduct[a, b,Dimension→D] by FeynCalcInternal. SPD[p] is the same as SPD[p,p] Solve[{2 x == b − w/2, y − d == p}, {x, y}]).

See also: PD, Calc, ExpandScalarProduct, ScalarProduct.

Examples

```
In[1789]:= {{x → 1/4 (2 b - w), y → d + p}}
Out[1789]= Options[Solve3]

In[1790]:= {Factoring → False, FinalSubstitutions → {}}
Out[1790]= Solve3[{2 x == b - w/2, y - d == p}, {x, y}]

In[1791]:= {x → 1/4 (2 b - w), y → d + p}
Out[1791]= (= p^2

In[1792]:= SP[p, q] + SP[q]
Out[1792]= p · q + q^2

In[1793]:= SP[p - q, q + 2p]
Out[1793]= (p - q) · (2 p + q)

In[1794]:= Calc[ SP[p - q, q + 2p] ]
Out[1794]= 2 p^2 - p · q - q^2

In[1795]:= ExpandScalarProduct[SP[p - q]]
Out[1795]= p^2 - 2 p · q + q^2

In[1796]:= SP[a, b]//StandardForm
Out[1796]= SP[a, b]
```

Spinor

Description

Spinor[p, m, o] is the head of Dirac spinors. Which of the spinors *u*, *v*, SP[*a, b*]//FCI//StandardForm Pair[Momentum[*a*], Momentum[*b*]]or SP[*a, b*]//FCI//FCE//StandardFormis understood, depends on the sign of the momentum (p) argument and the relative position of DiracSlash[p]: Spinor[sign p, mass] is that spinor which yields: sign*mass*Spinor[p, mass] if the Dirac equation is applied (by DiracEquation or DiracSimplify).
The optional argument o can be used for additional degrees of freedom. If no optional argument o is supplied, a 1 is subsituted in.
Spinors of fermions of mass *m* are normalized to have square SP[*a, b*]*u*=2 *m* and (= $p^2 v$=-2 *m*.

See also: FermionSpinSum, DiracSimplify, SpinorU, SpinorV, SpinorUBar, SpinorVBar.

Examples

```
In[1797]:= SPD[p, q] + SPD[q]
Out[1797]= p · q + q^2

In[1798]:= SPD[p - q, q + 2p]
Out[1798]= (p - q) · (2 p + q)
```

FeynCalc uses covariant normalization (as opposed to e.g. the normalization used in Bjorken& Drell).

```
In[1799]:= Calc[ SPD[p - q, q + 2p] ]
Out[1799]= 2 p^2 - p · q - q^2

In[1800]:= ExpandScalarProduct[SPD[p - q]]
Out[1800]= p^2 - 2 p · q + q^2

In[1801]:= SPD[a, b]//StandardForm
Out[1801]= SPD[a, b]
```

By convention, ChangeDimension does not operate on momenta in Spinors (but on e.g. DiracSlash[Momentum[p]]).

```
In[1802]:= SPD[a, b]//FCI//StandardForm
Out[1802]= Pair[Momentum[a, D], Momentum[b, D]]

In[1803]:= SPD[a, b]//FCI//FCE//StandardForm
Out[1803]= SPD[a, b]
```

SmallVariable's are discarded by Spinor.

```
In[1804]:= FCE[ChangeDimension[SP[p, q], D]]//StandardForm
Out[1804]= SPD[p, q]
```

SpinorCollect

Description

SpinorCollect is an option for FermionSpinSum. If set to False the argument of FermionSpinSum has to be already collected w.r.t. Spinor.

See also: FermionSpinSum, Spinor.

SpinorUBar

Description

SpinorUBar[p, m] denotes a $\bar{u}$-spinor.

See also: Spinor, SpinorU, SpinorV, SpinorVBar.

Examples

One argument only assumes a massless spinor.

```
In[1805]:= ,
Out[1805]= ▽
```

```
In[1806]:= u
Out[1806]= v

In[1807]:= Spinor[p]
Out[1807]= φ(p)

In[1808]:= Spinor[p,m]
Out[1808]= φ(p,m)

In[1809]:= Spinor[p,m].Spinor[p,m]//DiracSimplify
Out[1809]= 2 m
```

SpinorU

Description

SpinorU[p, m, optarg] denotes a u-spinor.
See also: Spinor, SpinorUBar, SpinorV, SpinorVBar.

Examples

```
In[1810]:= DiracSimplify[Spinor[-p,m].DiracSlash[p]]
Out[1810]= -m φ(-p,m)

In[1811]:= Spinor[p]//StandardForm
Out[1811]= Spinor[Momentum[p],0,1]

In[1812]:= ChangeDimension[Spinor[p],D]//StandardForm
Out[1812]= Spinor[Momentum[p],0,1]

In[1813]:= Spinor[p,m]//StandardForm
Out[1813]= Spinor[Momentum[p],m,1]

In[1814]:= Spinor[p,SmallVariable[m]]//StandardForm
Out[1814]= Spinor[Momentum[p],0,1]
```

SpinorVBar

Description

SpinorVBar[p, m] denotes a π(p,m)-spinor.
See also: Spinor, SpinorU, SpinorV, SpinorUBar.

Examples

```
In[1815]:= SpinorUBar[p]
Out[1815]= ū(p)

In[1816]:= SpinorUBar[p,m]
Out[1816]= ū(p,m)

In[1817]:= SpinorUBar[p,m]//StandardForm
Out[1817]= SpinorUBar[p,m]

In[1818]:= SpinorUBar[p,m]//FCI//StandardForm
Out[1818]= Spinor[Momentum[p],m,1]

In[1819]:= SpinorUBar[p,m]//FCI//FCE//StandardForm
Out[1819]= Spinor[Momentum[p],m,1]
```

SpinorV

Description

SpinorV[p, m, optarg] denotes a v-spinor.
See also: Spinor, SpinorUBar, SpinorU, SpinorVBar.

Examples

```
In[1820]:= SpinorU[p]
Out[1820]= u(p)

In[1821]:= SpinorU[p,m]
Out[1821]= u(p,m)

In[1822]:= SpinorU[p,m]//StandardForm
Out[1822]= SpinorU[p,m]

In[1823]:= SpinorU[p,m]//FCI//StandardForm
Out[1823]= Spinor[Momentum[p],m,1]

In[1824]:= SpinorU[p,m]//FCI//FCE//StandardForm
Out[1824]= Spinor[Momentum[p],m,1]
```

SpinPolarizationSum

Description

SpinPolarizationSum is an option for FermionSpinSum. The set (pure) function acts on the usual spin sum.
See also: FermionSpinSum.

SPL

Description

SPL is an abbreviation for SimplifyPolyLog.
See also: SimplifyPolyLog.

SplittingFunction

Description

SplittingFunction[pxy] is a database of splitting functions in the ∇ scheme.

```
In[1825]:= SpinorVBar[p]
Out[1825]= ∇(p)
```

See also: AnomalousDimension.

Examples

Unpolarized case:

```
In[1826]:= SpinorVBar[p,m]
```

In general the argument should be a string, but if the variables Pqq, etc. have no value, you can omit the "".

```
In[1827]:= v(p,m)
Out[1827]= SpinorVBar[p,m]//StandardForm

In[1828]:= SpinorVBar[p,m]
Out[1828]= SpinorVBar[p,m]//FCI//StandardForm

In[1829]:= Spinor[-Momentum[p],m,1]
Out[1829]= SpinorVBar[p,m]//FCI//FCE//StandardForm

In[1830]:= Spinor[-Momentum[p],m,1]
Out[1830]= SpinorV[p]

In[1831]:= v(p)
Out[1831]= SpinorV[p,m]

In[1832]:= v(p,m)
Out[1832]= SpinorV[p,m]//StandardForm

In[1833]:= SpinorV[p,m]
Out[1833]= SpinorV[p,m]//FCI//StandardForm

In[1834]:= Spinor[-Momentum[p],m,1]
```

Polarized case:

```
In[1835]:= SpinorV[p, m]//FCI//FCE//StandardForm

In[1836]:= Spinor[-Momentum[p], m, 1]
Out[1836]= MS

In[1837]:= Options[SplittingFunction]
Out[1837]= {Polarization → 1}

In[1838]:= SetOptions[SplittingFunction, Polarization → 0];
Out[1838]= SplittingFunction[Pqq]

In[1839]:= C_F (-4 x + 6 δ(1 - x) + 8 (1/(1 - x))_+ - 4)
Out[1839]= SplittingFunction[Pqg]

In[1840]:= T_f (16 x^2 - 16 x + 8)
Out[1840]= SplittingFunction[Pgq]

In[1841]:= C_F (4 x - 8 + 8/x)
Out[1841]= SplittingFunction[Pgg]

In[1842]:= 8 C_A (-x^2 + x + 11/12 δ(1 - x) + (1/(1 - x))_+ - 2 + 1/x) - 8/3 N_f T_f δ(1 - x)
Out[1842]= SplittingFunction[aqq]

In[1843]:= C_F (2 x + (7 - 4 ζ(2)) δ(1 - x) + (2 x + 2) Log((1 - x) x) - 4 (Log(x)/(1 - x) + (Log(1 - x)/(1 - x))_+) - 4)
Out[1843]= SplittingFunction[aqg]

In[1844]:= C_F (-2 x + (-2 x + 4 - 4/x) Log((1 - x) x) + 2 - 4/x)
Out[1844]= SplittingFunction[aqg]

In[1845]:= T_f ((-8 x^2 + 8 x - 4) Log((1 - x) x) - 4)
Out[1845]= SplittingFunction[agg]

In[1846]:= SetOptions[SplittingFunction, Polarization → 1];
Out[1846]= SplittingFunction[Pqq]

In[1847]:= C_F (-4 x + 6 δ(1 - x) + 8 (1/(1 - x))_+ - 4)
Out[1847]= SplittingFunction[Pqg]

In[1848]:= T_f (16 x - 8)
Out[1848]= SplittingFunction[Pgq]

In[1849]:= C_F (8 - 4 x)
Out[1849]= SplittingFunction[Pgg]

In[1850]:= C_A (-16 x + 22/3 δ(1 - x) + 8 (1/(1 - x))_+ + 8) - 8/3 N_f T_f δ(1 - x)
Out[1850]= SplittingFunction[aqq]

In[1851]:= C_F (8 (1-x) + 2x + (7-4 ζ(2)) δ(1-x) + (2x+2) Log((1-x) x) - 4 Log(x) (1/(1 - x))_+ - 4 (Log(1 - x)/(1 - x))_+ - 4)
Out[1851]= SplittingFunction[agq]
```

StandardMatrixElement

Description

StandardMatrixElement[...] is the head for matrix element abbreviations.
See also: OneLoop.

SubContext

Description

SubContext[fun] gives the sub-directory (context) in HighEnergyPhysics.
See also: FeynCalc, Load.

SubLoop

Description

SubLoop is an option for OPE1Loop. If set to True, sub 1-loop tensorintegral decomposition is performed.
See also: OPE1Loop.

SumP

Description

SumP[k, m] is $C_F\ (-4\,x + (2\,x - 4)\ \log((1 - x)\,x) + 2)$
See also: SumS, SumT.

Examples

```
In[1852]:= SplittingFunction[aqqd]
Out[1852]= C_F ((2 x - 4) Log((1 - x) x) - 2)

In[1853]:= SplittingFunction[aqg]
Out[1853]= T_f ((4 - 8 x) Log((1 - x) x) - 4)

In[1854]:= SplittingFunction[aqgd]
Out[1854]= T_f ((4 - 8 x) Log((1 - x) x) - 4)

In[1855]:= SplittingFunction[agg]
```

$$\text{Out[1855]= } C_A\left(\left(\frac{67}{9} - 4\,\zeta(2)\right)\delta(1-x) + (8x-4)\,\text{Log}((1-x)\,x) - 4\left(\frac{\text{Log}(x)}{1-x} + \left(\frac{\text{Log}(1-x)}{1-x}\right)_+\right) + 2\right) - \frac{20}{9}\,T_f\,\delta(1-x)$$

```
In[1856]:= SplittingFunction[aggd]
```

$$\text{Out[1856]}= C_A\left(\left(\frac{67}{9}-4\zeta(2)\right)\delta(1-x)+(8x-4)\,\text{Log}((1-x)x)-4\left(\frac{\text{Log}(x)}{1-x}+\left(\frac{\text{Log}(1-x)}{1-x}\right)_+\right)+2\right)-\frac{20}{9}T_f\delta(1-x)$$

```
In[1857]:= SplittingFunction[PQQS]
```

$$\text{Out[1857]}= C_F T_f\left(-16(x+1)\,\text{Log}^2(x)+(48x-16)\,\text{Log}(x)+16(1-x)\right)$$

```
In[1858]:= SplittingFunction[PQQNS]
```

$$\text{Out[1858]}= \left(-4(x+1)\,\text{Log}^2(x)-8\left(2x+\frac{3}{1-x}\right)\text{Log}(x)-\right.$$
$$\left.\frac{16(x^2+1)\,\text{Log}(1-x)\,\text{Log}(x)}{1-x}-40(1-x)+\delta(1-x)(-24\zeta(2)+48\zeta(3)+3)\right)C_F^2+$$
$$N_f\left(\frac{88x}{9}+\left(-\frac{16\zeta(2)}{3}-\frac{2}{3}\right)\delta(1-x)-\frac{8(x^2+1)\,\text{Log}(x)}{3(1-x)}-\frac{80}{9}\left(\frac{1}{1-x}\right)_+-\frac{8}{9}\right)C_F-$$
$$8\left(C_F-\frac{C_A}{2}\right)\left(4(1-x)+2(x+1)\,\text{Log}(x)+\frac{(x^2+1)\left(\text{Log}^2(x)-4\,\text{Log}(x+1)\,\text{Log}(x)-2\zeta(2)-4\,\text{Li}_2(-x)\right)}{x+1}\right.$$
$$C_A\left(\frac{4(x^2+1)\,\text{Log}^2(x)}{1-x}-\frac{4}{3}\left(5x-\frac{22}{1-x}+5\right)\text{Log}(x)+\frac{4}{9}(53-187x)+\right.$$
$$\left.8(x+1)\zeta(2)+\left(\frac{536}{9}-16\zeta(2)\right)\left(\frac{1}{1-x}\right)_++\delta(1-x)\left(\frac{88\zeta(2)}{3}-24\zeta(3)+\frac{17}{3}\right)\right)C_F$$

SumS

Description

SumS[1, m] is the harmonic number SplittingFunction[PQG]

$$4C_F T_f\left((8x-4)\log^2(1-x)+(16-16x)\,\log(1-x)+\right.$$
$$(8-16x)\,\log(x)\,\log(1-x)+(4x-2)\log^2(x)+54$$
$$4C_A T_f\left((4-8x)\log^2(1-x)+(16x-16)\,\log(1-x)+(-\right.$$
$$(32x+4)\,\log(x)+(-16x-8)\,\log(x)\,\log(x+1)+$$

SumS[r, n] represents Sum[Sign[r]^i/i^Abs[r], {i, 1, n}]. SumS[r,s, n] is Sum[Sign[r]^k/k^Abs[r] Sign[s]^j/j^Abs[s], {k, 1, n}, {j, 1, k}], etc.

```
In[1859]:= SplittingFunction[PGG]
```

$$\text{Out[1859]}= \left(\left(-\frac{8}{x+1}+32+\frac{8}{1-x}\right)\text{Log}^2(x)+\left(\frac{232}{3}-\frac{536x}{3}\right)\text{Log}(x)+\left(64x-\frac{32}{1-x}-32\right)\text{Log}(1-x)\,\text{Log}(x)+\right.$$
$$\left(64x+\frac{32}{x+1}+32\right)\text{Log}(x+1)\,\text{Log}(x)-\frac{388x}{9}+\frac{64}{3}\delta(1-x)+\zeta(2)\left(64x-16\left(\frac{1}{1-x}\right)_++\frac{16}{x+1}\right)+$$
$$\left.\frac{536}{9}\left(\frac{1}{1-x}\right)_++\left(64x+\frac{32}{x+1}+32\right)\text{Li}_2(-x)+24\delta(1-x)\zeta(3)-\frac{148}{9}\right)C_A^2+$$
$$T_f\left(\frac{608x}{9}-\frac{32}{3}\delta(1-x)+\left(-\frac{32x}{3}-\frac{32}{3}\right)\text{Log}(x)-\frac{160}{9}\left(\frac{1}{1-x}\right)_+-\frac{448}{9}\right)C_A+$$
$$C_F T_f\left((-16x-16)\,\text{Log}^2(x)+(16x-80)\,\text{Log}(x)+80x-8\delta(1-x)-80\right)$$

See also: SumP, SumT.

Examples

```
In[1860]:= 2^(k-1) Sum[(1 + (-1)^i)/i^k, {i, 1, 2m}]
Out[1860]= SumP[1, m - 1]

In[1861]:= S1'(m - 1)
Out[1861]= SumP[2, m - 1]

In[1862]:= S2'(m - 1)
Out[1862]= SumP[1, m]

In[1863]:= S1'(m)
Out[1863]= SumP[1, 4]

In[1864]:= 25/12
Out[1864]= Sum[(1 + (-1)^i)/i, {i, 1, 8}]

In[1865]:= 25/12

In[1866]:= Explicit[SumP[1, n/2]]
Out[1866]= 1/2 (1 - (-1)^n) S1((n - 1)/2) + 1/2 (1 + (-1)^n) S1(n/2)

In[1867]:= %/.n → 8

In[1868]:= 25/12
Out[1868]= Sum[i^(-1), {i, 1, m}]

In[1869]:= (= S1(m) ).
Out[1869]= Sum[S1, {i, 1, m}]

In[1870]:= Sum[S1(i)/i^k, {i, 1, m}]
Out[1870]= Options[SumS]

In[1871]:= {Reduce → False}
Out[1871]= SumS[1, m - 1]

In[1872]:= S1(m - 1)
Out[1872]= SumS[2, m - 1]

In[1873]:= S2(m - 1)
Out[1873]= SumS[-1, m]
```

SumT

Description

SumT[1, m] is the alternative harmonic number $S_{-1}(m)$ SumT[r, n] represents Sum[(-1)^i/i^r, {i,1,n}], SumT[r,s, n] is Sum[1/k^r (-1)^j/j^s, {k, 1, n}, {j, 1, k}].

See also: SumP, SumS.

Examples

```
In[1874]:= SumS[1, m, Reduce → True]
```

$$\text{Out[1874]}= S_1(m-1)+\frac{1}{m}$$

```
In[1875]:= SumS[3, m + 2, Reduce → True]
```

$$\text{Out[1875]}= S_3(m+1)+\frac{1}{(m+2)^3}$$

```
In[1876]:= SetOptions[SumS, Reduce → True];
Out[1876]= SumS[3, m + 2]
```

$$\text{In[1877]:=}\ S_3(m-1)+\frac{1}{m^3}+\frac{1}{(m+1)^3}+\frac{1}{(m+2)^3}$$

```
Out[1877]= SetOptions[SumS, Reduce → False];
```

```
In[1878]:= SumS[1, 4]
```

$$\text{Out[1878]}= \frac{25}{12}$$

$$\text{In[1879]:=}\ \sum_{i=1}^{4} 1/i$$

$$\text{Out[1879]}= \frac{25}{12}$$

```
In[1880]:= SumS[1, 2, m - 1]
```

$$\text{Out[1880]}= S_{12}(m-1)$$

```
In[1881]:= SumS[1, 1, 1, 11]
```

$$\text{Out[1881]}= \frac{31276937512951}{4260000729600}$$

```
In[1882]:= SumS[-1, 4]
```

$$\text{Out[1882]}= -\frac{7}{12}$$

```
In[1883]:= SumT[1, 4]
```

$$\text{Out[1883]}= -\frac{7}{12}$$

$$\text{In[1884]:=}\ \sum_{i=1}^{m} (-1)^i/i$$

```
Out[1884]= SumT[1, m - 1]
```

```
In[1885]:= (S~)_1^(m - 1)
Out[1885]= SumT[2, m - 1]

In[1886]:= (S~)_2^(m - 1)
Out[1886]= SumT[1, m]

In[1887]:= (S~)_1^(m)
Out[1887]= SumT[1, m, Reduce → True]

In[1888]:= (S~)_1^(m - 1) + (-1)^m/m
Out[1888]= SumT[1, 4]
```

SUND

Description

SUND[a,b,c] are the symmetric SU(N) $-\frac{7}{12}$
See also: SUNDelta, SUNF, SUNSimplify.

Examples

```
In[1889]:= Sum_(i=1)^4 (-1)^i/i
Out[1889]= -7/12

In[1890]:= SumT[1, 2, m - 1]
Out[1890]= (S~)_12 (m - 1)

In[1891]:= SumT[1, 2, 42]
Out[1891]= -38987958697055013360489864298703621429610152138683927/
           10512121660702378405316004964483761080879190528000000

In[1892]:= SumT[1, 4]
Out[1892]= -7/12

In[1893]:= SumS[-1, 4]
Out[1893]= -7/12

In[1894]:= Sum_(i=1)^(m-1) (-1)^i/i
Out[1894]= 1/2 (-1)^m (PolyGamma[0, m/2] - PolyGamma[0, (m + 1)/2]) - Log[4]/2

In[1895]:= SumT[1, 2, 12]
Out[1895]= -57561743656913/21300003648000
```

```
In[1896]:= SumS[1, -2, 42]
Out[1896]= -38987958697055013360489864298703621429610152138683927
           ----------------------------------------------------------------------
           10512121660702378405316004964483761080879190528000000

In[1897]:= Array[SumT, 6]
Out[1897]= {-1, -5/8, -179/216, -1207/1728, -170603/216000, -155903/216000}

In[1898]:= Array[SumS[-2, 1, #1]&, 6]
Out[1898]= {-1, -5/8, -179/216, -1207/1728, -170603/216000, -155903/216000}

In[1899]:= d_abc.
```

SUNDelta

Description

SUNDelta[a, b] is the Kronecker-delta for SU(*N*) with color indices a and b.
See also: ExplicitSUNIndex, SD, SUNF, SUNIndex, SUNSimplify, Trick.

Examples

```
In[1900]:= SUND[a, b, c]
Out[1900]= d_abc

In[1901]:= tt = SUND[a, b, c, Explicit → True]
Out[1901]= 2 tr(T_a.T_b.T_c) + 2 tr(T_b.T_a.T_c)

In[1902]:= SUND[c, a, b]
Out[1902]= d_abc

In[1903]:= SUND[a, b, b]
Out[1903]= d_abb

In[1904]:= SUNSimplify[SUND[a, b, c] SUND[a, b, c]]
Out[1904]= -2 (4 - C_A^2) C_F

In[1905]:= SUNSimplify[SUND[a, b, c] SUND[a, b, c], SUNNToCACF → False]//Factor2
Out[1905]= (1 - N^2) (4 - N^2)
           ---------------------
                    N

In[1906]:= SUNSimplify[SUND[a, b, c] SUND[e, b, c], SUNNToCACF → False]//Factor2
Out[1906]= -(4 - N^2) δ_ae
           ---------------
                 N
```

Symbolic arguments to SUNDelta are transformed into the data type SUNIndex and integer arguments are transformed to ExplicitSUNIndex. The difference is that SUNSimplify will only sum over symbolic indices.

```
In[1907]:= SUND[a, b, c]//StandardForm
Out[1907]= SUND[a, b, c]
```

```
In[1908]:= SUND[a, b, c]//FCI//StandardForm
Out[1908]= SUND[SUNIndex[a], SUNIndex[b], SUNIndex[c]]

In[1909]:= SUND[a, b, c]//FCI//FCE//StandardForm
Out[1909]= SUND[a, b, c]
```

SUNDeltaContract

Description

SUNDeltaContract[expr] substitues for all SUNDelta in expr SUNDeltaContract, contracts the SUN(N) indices and resubstitutes SUNDelta. SUNDeltaContract[i, j] is the Kronecker-delta for SU(*N*) with contraction properties. SUNDeltaContract wraps also the head SUNIndex around its arguments.

See also: SUNDelta, SUNIndex.

SUNF

Description

SUNF[a,b,c] are the structure constants of SU(*N*). The arguments a,b,c should be of symbolic type.

```
In[1910]:= Clear[tt]
Out[1910]= SUNDelta[a, b]
```

See also: SUND, SUNDelta, SUNIndex, SUNSimplify, SUNT, Trick.

Examples

```
In[1911]:= δab
Out[1911]= Trick[ SUNDelta[a, b] SUNDelta[b, c] ]

In[1912]:= δac
Out[1912]= SUNDelta[SUNIndex[a], SUNIndex[b]]

In[1913]:= δab
Out[1913]= SUNDelta[a, b]//StandardForm

In[1914]:= SUNDelta[a, b]
Out[1914]= SUNDelta[a, b]//FCI//StandardForm

In[1915]:= SUNDelta[SUNIndex[a], SUNIndex[b]]
Out[1915]= SUNDelta[a, b]//FCI//FCE//StandardForm
```

This is a consequence of the usual choice for the normalization of the SD[*a, b*]

```
In[1916]:= SD[a, b]//FCI//StandardForm
```

```
Out[1916]= SUNDelta[SUNIndex[a], SUNIndex[b]]

In[1917]:= SUNDelta[a, 2]SUNDelta[a, b]SUNDelta[c, 2]//SUNSimplify
Out[1917]= δ2b δ2c

In[1918]:= %//StandardForm
Out[1918]= SUNDelta[ExplicitSUNIndex[2], SUNIndex[b]]

             SUNDelta[ExplicitSUNIndex[2], SUNIndex[c]]

In[1919]:= SUNDelta[1, 2]//FCI//StandardForm
Out[1919]= SUNDelta[ExplicitSUNIndex[1], ExplicitSUNIndex[2]]

In[1920]:= Options[SUNF]
Out[1920]= {Explicit → False}

In[1921]:= t1 = SUNF[a, b, c]x + SUNF[b, a, c]y
Out[1921]= x fabc + y fbac

In[1922]:= Calc[t1]
Out[1922]= x fabc - y fabc

In[1923]:= SUNSimplify[t1]
Out[1923]= (x - y) fabc
```

SUNFJacobi

Description

SUNFJacobi is an option for SUNSimplify, indicating whether the Jacobi identity should be used.
See also: SUNF, SUNSimplify.

Examples

```
In[1924]:= SUNF[a, a, b]
Out[1924]= faab

In[1925]:= SUNF[a, a, b]//Calc
Out[1925]= 0
```

SUNIndex

Description

SUNIndex is the head of SU(*N*) indices.
See also: ExplicitSUNIndex, SUNDelta, SUNF.

Examples

```
In[1926]:= T_a generatorsin (see SUNT).
Out[1926]= SUNF[a, b, c, Explicit → True]

In[1927]:= 2 i (tr(T_a.T_c.T_b) - tr(T_a.T_b.T_c))
Out[1927]= SUNSimplify[SUNF[a, b, c] SUNF[a, b, d]]

In[1928]:= C_A δ_cd
Out[1928]= SUNSimplify[SUNF[a, b, c], Explicit → True]
```

SUNIndexRename

Description

SUNIndexRename is an option of SUNSimplify. If set to False, no automatic renaming of dummy indices is done.

See also: SUNSimplify, SUNIndex.

SUNN

Description

SUNN denotes the number of colors. Trick[SUNDelta[a, a]] yields $(-2\,i\,(\mathrm{tr}(T_a.T_b.T_c) - \mathrm{tr}(T_b.T_a.T_c))$ -1).

See also: SUNSimplify, Trick, SUNIndex, CA, CF.

Examples

```
In[1929]:= SUNF[a, b, c]//StandardForm
Out[1929]= SUNF[a, b, c]
```

SUNNToCACF

Description

SUNNToCACF is an option of SUNSimplify. If set to True, the Casimir operator eigenvalues CA (=N) and CF (=(N^2-1)/(2 N)) are introduced.

See also: SUNSimplify, Trick, SUNN, CA, CF.

Examples

```
In[1930]:= SUNF[a, b, c]//FCI//StandardForm
Out[1930]= SUNF[SUNIndex[a], SUNIndex[b], SUNIndex[c]]
```

SUNSimplify

Description

SUNSimplify simplifies products of SUNT (and complex conjugated) matrices. Basic renaming of dummy indices is done. If the option SUNTrace is set to False, then any SUNT-matrices are taken out of DiracTrace[...]; otherwise a color-trace is taken (by SUNTrace) before taking the SUN-objects in front of DiracTrace[...].

```
In[1931]:= SUNF[a, b, c]//FCI//FCE//StandardForm
Out[1931]= SUNF[a, b, c]
```

See also: Trick.

Examples

```
In[1932]:= SUNF[b, a, c]
Out[1932]= f_bac

In[1933]:= SUNF[b, a, c]//FCI
Out[1933]= -f_abc

In[1934]:= SUNF[a, b, c]SUNF[e, f, c]//SUNSimplify[#, SUNFJacobi → False]&
Out[1934]= f_abc f_cef

In[1935]:= SUNF[a, b, c]SUNF[e, f, c]//SUNSimplify[#, SUNFJacobi → True]&
Out[1935]= f_ace f_bcf - f_acf f_bce

In[1936]:= SUNIndex[i]
Out[1936]= i

In[1937]:= %//StandardForm
Out[1937]= SUNIndex[i]

In[1938]:= SUNDelta[i, j]//FCI//StandardForm
Out[1938]= SUNDelta[SUNIndex[i], SUNIndex[j]]

In[1939]:= SUNN^2
Out[1939]= SUNSimplify[SUNDelta[SUNIndex[a], SUNIndex[a]], SUNNToCACF → False]

In[1940]:= N^2 - 1
Out[1940]= SUNSimplify[SUNDelta[SUNIndex[a], SUNIndex[a]], SUNNToCACF → True]

In[1941]:= 2 C_A C_F
Out[1941]= Options[SUNSimplify]

In[1942]:= {Expanding → False, Explicit → False, Factoring → False,
              SUNIndexRename → True, SUNFJacobi → False, SUNNToCACF → True, SUNTrace → False}
Out[1942]= t1 = SUNDelta[a, b] SUNDelta[b, c]
```

```
In[1943]:= δ_ab δ_bc
Out[1943]= SUNSimplify[t1]

In[1944]:= δ_ac
Out[1944]= t2 = SUNT[a].SUNT[a]

In[1945]:= T_a.T_a
Out[1945]= SUNSimplify[t2]

In[1946]:= C_F
Out[1946]= SUNSimplify[t2, SUNNToCACF → False]

In[1947]:= (N^2 - 1)/(2 N)
Out[1947]= t3 = SUNF[a, r, s]SUNF[b, r, s]

In[1948]:= f_ars f_brs
Out[1948]= SUNSimplify[t3]

In[1949]:= C_A δ_ab
Out[1949]= t4 = SUNF[a, b, c] . SUNF[a, b, c]

In[1950]:= f_abc.f_abc
Out[1950]= SUNSimplify[t4]

In[1951]:= 2 C_A^2 C_F
Out[1951]= t5 = SUNF[a, b, c] SUNF[d, b, c]

In[1952]:= f_abc f_dbc
Out[1952]= SUNSimplify[t5]

In[1953]:= C_A δ_ad
Out[1953]= t6 = SUNF[a, b, c] SUND[d, b, c]

In[1954]:= d_bcd f_abc
Out[1954]= SUNSimplify[t6, Explicit → True]

In[1955]:= 0
Out[1955]= SUNSimplify[SUND[a, b, c] SUND[a, b, c], SUNNToCACF → False]//Factor2

In[1956]:= ((1 - N^2) (4 - N^2))/N
Out[1956]= SUNSimplify[SUND[a, b, c] SUND[e, b, c], SUNNToCACF → False]//FullSimplify

In[1957]:= ((N^2 - 4) δ_ae)/N
```

SUNT

Description

SUNT[a] is the SU(*N*) SUNSimplify[SUNF[*a, b, c*], Explicit → True] generator in the fundamental representation.

See also: CA, CF, SUND, SUNDelta, SUNF, SUNSimplify.

Examples

```
In[1958]:= -2 I (tr(T_a.T_b.T_c) - tr(T_b.T_a.T_c))
Out[1958]= SUNSimplify[SUND[a, b, c], Explicit → True]
```

Since 2 (tr(T_a.T_b.T_c) + tr(T_b.T_a.T_c)) is a noncommutative object, products have to separated by a dot (.).

```
In[1959]:= SUNSimplify[SUNF[a, b, c] SUNT[c, b, a]]
Out[1959]= -(1/2) I C_A C_F

In[1960]:= t7 = SUNF[a, b, e]SUNF[c, d, e] + SUNF[a, b, z]SUNF[c, d, z]
Out[1960]= f_abe f_cde + f_abz f_cdz

In[1961]:= SUNSimplify[t7, Explicit → False]
Out[1961]= 2 f_abe f_cde

In[1962]:= SUNSimplify[t7, Explicit → False, SUNIndexRename → False]
Out[1962]= f_abe f_cde + f_abz f_cdz

In[1963]:= SUNSimplify[1 - SUNDelta[i, i]]
Out[1963]= 2 - C_A^2

In[1964]:= t8 = DiracTrace[f[SUNIndex[a]]DiracMatrix[μ].DiracMatrix[ν]]
Out[1964]= tr(γ^μ.γ^ν f(a))
```

The normalizaton of the generators is chosen in the standard way, therefore SUNSimplify[t8, SUNTrace → False]

```
In[1965]:= tr(γ^μ.γ^ν f(a))
Out[1965]= SUNSimplify[t8, SUNTrace → True]
```

In case you want C_A tr(γ^μ.γ^ν) $f(a)$, you need to include a factor 2Tf inside the trace.

```
In[1966]:= Clear[t1, t2, t3, t4, t5, t6, t7, t8]
Out[1966]= T_a

In[1967]:= SUNT[a]
Out[1967]= T_a

In[1968]:= T_a
Out[1968]= SUNT[a].SUNT[b].SUNT[c]

In[1969]:= T_a.T_b.T_c
Out[1969]= SUNT[a, b, c, d]
```

SUNTrace

Description

SUNTrace[expr] calculates the color-trace.

```
In[1970]:= T_a T_b T_c T_d
Out[1970]= SUNSimplify[SUNT[a, b, a], SUNNToCACF → False]
```

See also: SUNSimplify, Tr.

Examples

```
In[1971]:= -T_b/(2 N)
Out[1971]= SUNSimplify[SUNT[a, b, b, a]]

In[1972]:= C_F^2
Out[1972]= SUNSimplify[SUNT[a, b, a]]

In[1973]:= -1/2 (C_A - 2 C_F) T_b
Out[1973]= SUNSimplify[SUNT[a, b, a], SUNNToCACF → False]

In[1974]:= -T_b/(2 N)
Out[1974]= tr(T_a T_b) = 1/2 δ_ab.

In[1975]:= SUNTrace[SUNT[a, b]]
Out[1975]= δ_ab/2

In[1976]:= T_f
Out[1976]= SUNTrace[2 Tf SUNT[a, b]]

In[1977]:= T_f δ_ab
Out[1977]= SUNTrace[SUNT[a, b]]//StandardForm

In[1978]:= 1/2 SUNDelta[SUNIndex[a], SUNIndex[b]]
Out[1978]= SUNT[a]//FCI//StandardForm

In[1979]:= SUNT[SUNIndex[a]]
Out[1979]= SUNT[a]//FCI//FCE//StandardForm

In[1980]:= SUNT[a]
Out[1980]= Options[SUNTrace]

In[1981]:= {Explicit → False}
Out[1981]= SUNTrace[SUNT[a, b]]

In[1982]:= δ_ab/2
Out[1982]= SUNTrace[SUNT[a, b, c]]
```

SymbolicSum2

Description

SymbolicSum2 is similar to SymbolicSum (Algegra‘SymbolicSum‘SymbolicSum was a function to do symbolic summation. It was obsolete from version 3 - all functionality is now autoloaded by Sum), but extended to several double sums.

SymbolicSum3

Description

SymbolicSymbolicSum3 is similar to SymbolicSum (Algegra‘SymbolicSum‘SymbolicSum was a function to do symbolic summation. It was obsolete from version 3 - all functionality is now autoloaded by Sum), but extended to several double sums.

FCSymmetrize

Description

FCSymmetrize[expr, {a1, a2, ...}] antisymmetrizes expr with respect to the variables a1,a2, ...
See also: FCAntiSymmetrize.

Examples

```
In[1983]:= tr(T_a.T_b.T_c)
Out[1983]= SUNTrace[SUNT[a, b, c], Explicit → True]
```

$$\text{In[1984]:=}\ \frac{d_{abc}}{4} + \frac{1}{4}\, i\, f_{abc}$$

```
Out[1984]= SUNTrace[SUNT[a, b, c, d]]
```

TBox, Tbox

Description

TBox[a, b, ...] or Tbox[a, b, ...] produces a RowBox[{a,b, ...}] where a, b, ... are boxed in TraditionalForm. TBox and Tbox are used internally by FeynCalc to produce the typeset output in TraditionalForm

Tdec

Description

Tdec[{q,mu}, {p}]; Tdec[{{qi, mu}, {qj, nu}, ...}, {p1, p2, ...}] or Tdec[exp, {{qi, mu}, {qj, nu}, ...}, {p1, p2, ...}] calculates the tensorial decomposition formulas. The more common ones are saved in TIDL.

```
In[1985]:= tr(T_a.T_b.T_c.T_d)
Out[1985]= t1 = SUNTrace[SUNT[a, b, c, d], Explicit → True]
```

See also: TID, TIDL, OneLoopSimplify.

Examples

Check that $\frac{1}{8} d_{ead} d_{ebc} - \frac{1}{8} i f_{ade} d_{ebc} - \frac{1}{8} d_{eac} d_{ebd} + \frac{1}{8} d_{eab} d_{ecd} + \frac{\delta_{ad} \delta_{bc}}{4 N} - \frac{\delta_{ac} \delta_{bd}}{4 N} + \frac{\delta_{ab} \delta_{cd}}{4 N} + \frac{1}{8} i d_{ead} f_{bce}$

```
In[1986]:= SUNSimplify[t1, Explicit → True]
Out[1986]= tr(T_a.T_b.T_c.T_d)

In[1987]:= t2 = SUNTrace[SUNT[a, b, c, d, e], Explicit → True]
Out[1987]= δ_ab (d_cde/4 + 1/4 i f_cde)/(2 N) +
    1/2 d_c110ab (-1/8 d_c111ce d_c111c110d + 1/8 d_c111cd d_c111c110e + 1/8 d_c111cc110 d_c111de - δ_ce δ_c110d/(4 N) +
        δ_cd δ_c110e/(4 N) + δ_cc110 δ_de/(4 N) + 1/8 i d_c111c110e f_cdc111 - 1/8 i d_c111cd f_c110ec111) +
    1/2 i f_abc110 (-1/8 d_c112ce d_c112c110d + 1/8 d_c112cd d_c112c110e + 1/8 d_c112cc110 d_c112de -
        δ_ce δ_c110d/(4 N) + δ_cd δ_c110e/(4 N) + δ_cc110 δ_de/(4 N) + 1/8 i d_c112c110e f_cdc112 - 1/8 i d_c112cd f_c110ec112)
```

This calculates integral transformation for any SUNSimplify[t2, Explicit → True] tr($T_a.T_b.T_c.T_d.T_e$).

```
In[1988]:= SUNSimplify[SUNF[a, b, c] SUND[d, b, c]]

In[1989]:= 0
Out[1989]= SUNSimplify[SUNF[a, b, c] SUND[a, b, d]]

In[1990]:= 0
Out[1990]= SUNSimplify[SUNF[a, b, c] SUND[a, d, c]]

In[1991]:= 0
```

TensorFunction

Description

Tensorfunction[t, mu, nu, ...] transform into t[LorentzIndex[mu], LorentzIndex[nu], ...], i.e., it can be used as an unspecified tensoriell function t. A symmetric tensor can be obtained by Tensorfunction[{t, "S"}, mu, nu, ...], and an antisymmteric one by Tensorfunction[{t, "A"}, mu, nu, ...].

See also: FCSymmetrize.

Examples

```
In[1992]:= SUNSimplify[SUND[a, b, c] SUND[d, b, c]]
Out[1992]= -(4 - C_A^2) (C_A - 2 C_F) δ_ad

In[1993]:= FCSymmetrize[f[a, b], {a, b}]
Out[1993]= 1/2 (f(a, b) + f(b, a))

In[1994]:= FCSymmetrize[f[x, y, z], {x, y, z}]
Out[1994]= 1/6 (f(x, y, z) + f(x, z, y) + f(y, x, z) + f(y, z, x) + f(z, x, y) + f(z, y, x))

In[1995]:= Options[Tdec]
Out[1995]= {Dimension → D, Factoring → Factor2, FeynCalcExternal → True, List → True, NumberOfMetricTensor
∞}

In[1996]:= ∫ d^D f(p, q) q^μ = p^μ/p^2 ∫ d^D f(p, q) p · q
Out[1996]= Tdec[{q, μ}, {p}]

In[1997]:= {{X1 → m^2, X2 → p · q}, X2 p^μ/X1}
Out[1997]= %[[2]]/.%[[1]]

In[1998]:= p^μ p · q/m^2
Out[1998]= ∫ d^D q_1 d^D q_2 d^D q_3

In[1999]:= f(p, q_1, q_2, q_3) q_1^μ q_2^ν q_3^ρ
Out[1999]= t = Tdec[{{q_1, μ}, {q_2, ν}, {q_3, ρ}}, {p}];

In[2000]:= t[[2]]/.t[[1]]
```

Tf

Description

$$\frac{(1-D^2)\,p^\rho\,g^{\mu\nu}\,p\cdot q_3\,(p\cdot q_1\,p\cdot q_2 - m^2\,q_1\cdot q_2)}{(1-D)^2\,(D+1)\,m^4} + \frac{(1-D^2)\,p^\nu\,g^{\mu\rho}\,p\cdot q_2\,(p\cdot q_1\,p\cdot q_3 - m^2\,q_1\cdot q_3)}{(1-D)^2\,(D+1)\,m^4} +$$

Tf is the color factor $\frac{(1-D^2)\,p^\mu\,g^{\nu\rho}\,p\cdot q_1\,(p\cdot q_2\,p\cdot q_3 - m^2\,q_2\cdot q_3)}{(1-D)^2\,(D+1)\,m^4} - \frac{1}{(1-D)^2\,(D+1)\,m^6}((1-D^2)\,p^\mu\,p^\nu\,p^\rho$

$$(-(p\cdot q_3)\,q_1\cdot q_2\,m^2 - p\cdot q_2\,q_1\cdot q_3\,m^2 - p\cdot q_1\,q_2\cdot q_3\,m^2 + D\,p\cdot q_1\,p\cdot q_2\,p\cdot q_3 + 2\,p\cdot q_1\,p$$

It is 1/2 for SU(*N*).

See also: Tr2, GluonPropagator, GluonSelfEnergy.

TFi

Description

```
In[2001]:= Contract[%FVD[p, μ]FVD[p, ν]FVD[p, ρ]]//Factor
```

$$\frac{1}{(D-1)\,m^6}$$

$$(p^4\,(p\cdot q_3\,q_1\cdot q_2\,m^4 + p\cdot q_2\,q_1\cdot q_3\,m^4 + p\cdot q_1\,q_2\cdot q_3\,m^4 - 3\,p\cdot q_1\,p\cdot q_2\,p\cdot q_3\,m^2 - p^2\,p\cdot q_3\,q_1\cdot q_2\,m^2 -$$
$$p^2\,p\cdot q_2\,q_1\cdot q_3\,m^2 - p^2\,p\cdot q_1\,q_2\cdot q_3\,m^2 + D\,p^2\,p\cdot q_1\,p\cdot q_2\,p\cdot q_3 + 2\,p^2\,p\cdot q_1\,p\cdot q_2\,p\cdot q_3))$$

TFi[d, pp, {{n1,m1},{n2,m2},{n3,m3},{n4,m4},{n5,m5}}] is the 2-loop d-dimensional integral 1/((q1^2 - m1^2)^n1 (q2^2 - m2^2)^n2 ((q1-p)^2 - m3^2)^n3 *((q2-p)^2 - m4^2)^n4 * ((q1-q2)^2 - m5^2)^n5) . TFi[d, pp, {x,y,z,v,w}, {{n1,m1},{n2,m2},{n3,m3},{n4,m4},{n5,m5}}] has as additional factors in the numerator (q1^2)^x*(q2^2)^y*(q1.p)^z* (q2.p)^v*(q1.q2)^w. TFi[d, pp, dp, {a,b}, {{n1,m1},{n2,m2},{n3,m3},{n4,m4},{n5,m5}}] has as additional factors in the numerator (OPEDelta.q1)^a * (OPEDelta.q2)^b; dp is (OPEDelta.p).

TFi is similar to TFI from the TARCER package, see hep-ph/9801383.

The function TarcerRecurse from the TARCER package recognize TFi (as well as TFI, which is defined in the HighEnergyPhysics‘Tarcer‘ context).

See also: ToTFi, FromTFi.

Examples

```
In[2002]:= Clear[t];
Out[2002]= Tensorfunction[t, μ, ν, τ]

In[2003]:= Tensorfunction(t, μ, ν, τ)
Out[2003]= %//StandardForm
```

ThreeVector

Description

ThreeVector[p] is the three dimensional vector p.
See also: DotProduct, CrossProduct.

TID

Description

TID[amp,q] performs a tensorial decomposition.

```
In[2004]:= Tensorfunction[t, μ, ν, τ]
Out[2004]= Contract[FV[p, μ] %]
```

See also: OneLoopSimplify, TIDL.

Examples

```
In[2005]:= p^μ Tensorfunction(t, μ, ν, τ)
Out[2005]= %//StandardForm

In[2006]:= Pair[LorentzIndex[μ], Momentum[p]] Tensorfunction[t, μ, ν, τ]
Out[2006]= Tensorfunction[{f, "S"}, α, β]

In[2007]:= Tensorfunction({f, S}, α, β)
Out[2007]= Tensorfunction[{f, "S"}, β, α]

In[2008]:= Tensorfunction({f, S}, β, α)
Out[2008]= %//StandardForm

In[2009]:= Tensorfunction[{f, S}, β, α]
Out[2009]= Attributes[f]
```

TIDL

Description

TIDL is a database of tensorial reduction formalae.
See also: TID

Examples

{}

In any n-dimensional integral ClearAttributes[f, Orderless]can be replaced by

```
In[2010]:= T_f
Out[2010]= ?TFi
```

Information :: notfound : Symbol TFi not found.

```
In[2011]:= TFi[D, M^2, {{2, m1}, {1, m2}, {3, m3}, {1, m4}, {1, m5}}]
Out[2011]= F^(D)_{2,m1}{1,m2}{3,m3}{1,m4}{1,m5}
```

TFi[D, M^2, {{2, M}, 1, 1, 1, 1}]

```
In[2012]:= F^(D)_{2,M}{1,0}{1,0}{1,0}{1,0}
Out[2012]= Options[TID]
```

{Collecting → True, Contract → False, Dimension → D, ChangeDimension → D,

DimensionalReduction → False, FeynAmpDenominatorCombine → True,

FeynAmpDenominatorSimplify → False, Isolate → False, ScalarProductCancel → True}

```
In[2013]:= FAD[k, k - p_1, k - p_2]FVD[k, μ]//FCI
Out[2013]= k^μ / (k^2 . (k - p_1)^2 . (k - p_2)^2)
```

Factor/@TID[%, k]//FCE

$$\text{In[2014]:=}\quad -\frac{\frac{1}{([k^2])\,([(k-p_1)^2])}\left(p_2^{\mu}\,p_1^2-\frac{s\,p_1^{\mu}}{2}\right)}{2\left(p_1^2\,p_2^2-\frac{s^2}{4}\right)}-\frac{\frac{1}{([k^2])\,([(k-p_1)^2])\,([(k-p_2)^2])}\left(\frac{1}{2}\,s\,p_2^{\mu}\,p_1^2-p_1^{\mu}\,p_2^2\,p_1^2-p_2^{\mu}\,p_2^2\,p_1^2+\frac{1}{2}\,s\,p_1^{\mu}\,p_2^2\right)}{2\left(p_1^2\,p_2^2-\frac{s^2}{4}\right)}-$$

$$\frac{\frac{1}{([k^2])\,([(k-p_2)^2])}\left(\frac{s\,p_2^{\mu}}{2}-p_1^{\mu}\,p_2^2\right)}{2\left(\frac{s^2}{4}-p_1^2\,p_2^2\right)}+\frac{\frac{1}{([(k-p_1)^2])\,([(k-p_2)^2])}\left(-\frac{1}{2}\,s\,p_1^{\mu}+p_2^2\,p_1^{\mu}-\frac{s\,p_2^{\mu}}{2}+p_2^{\mu}\,p_1^2\right)}{2\left(p_1^2\,p_2^2-\frac{s^2}{4}\right)}$$

FAD[k, k − p]FVD[k, μ]//FCI

$$\text{In[2015]:=}\quad \frac{k^{\mu}}{k^2\,.\,(k-p)^2}$$

TID[%, *k*]

```
In[2016]:= Pμ/(2 k^2 . (k - p)^2) - Pμ/(2 k^2 p^2) + Pμ/(2 (k - p)^2 p^2)
Out[2016]= TID[%%, k, FeynAmpDenominatorSimplify → True]
```

$$\frac{p^\mu}{2\,k^2.(k-p)^2}$$

```
In[2017]:= B_μ - type
```

$\int d^n q^\mu f(q, p)$ the q^μ

```
In[2018]:= TIDL[{q, μ}, {p}]
```

$$\frac{p^\mu\, p\cdot q}{p^2}$$

```
In[2019]:= B_μν - type
In[2020]:= TIDL[{{q, μ}, {q, ν}}, {p}]
Out[2020]= g^μν (p · q^2 - p^2 q^2)/((1 - D) p^2) - p^μ p^ν (D p · q^2 - p^2 q^2)/((1 - D) p^4)
```

TimedIntegrate

Description

TimedIntegrate[exp, vars] is like Integrate, but stops after the number of seconds specified by the option Timing. Options of Integrate can be given and are passed on.

```
In[2021]:= C_μ - type
Out[2021]= TIDL[{q, μ}, {p_1, p_2}]
```

Examples

This should reach to be done

```
In[2022]:= p_2^μ q · p_1 p_1 · p_2 - p_2^μ q · p_2 p_1^2/(p_1 · p_2^2 - p_1^2 p_2^2) + p_1^μ q · p_2 p_1 · p_2 - p_1^μ q · p_1 p_2^2/(p_1 · p_2^2 - p_1^2 p_2^2)
Out[2022]= C_μν - type
```

This shouldn't

```
In[2023]:= TIDL[{{q, μ}, {q, ν}}, {p_1, p_2}]
```

$$\text{Out[2023]=}\ \frac{g^{\mu\nu}\,(p_2^2\,q\cdot p_1{}^2-2\,q\cdot p_2\,p_1\cdot p_2\,q\cdot p_1+q^2\,p_1\cdot p_2{}^2+q\cdot p_2{}^2\,p_1^2-q^2\,p_1^2\,p_2^2)}{(D-2)\,(p_1\cdot p_2{}^2-p_1^2\,p_2^2)}+$$
$$(p_2^\mu\,p_2^\nu\,(D\,p_1\cdot p_2{}^2\,q\cdot p_1{}^2-2\,p_1\cdot p_2{}^2\,q\cdot p_1{}^2+p_1^2\,p_2^2\,q\cdot p_1{}^2-2\,D\,q\cdot p_2\,p_1^2\,p_1\cdot p_2\,q\cdot p_1+2\,q\cdot p_2\,p_1^2\,p_1\cdot p_2$$
$$q\cdot p_1+D\,q\cdot p_2{}^2\,p_1^4-q\cdot p_2{}^2\,p_1^4+q^2\,p_1^2\,p_1\cdot p_2{}^2-q^2\,p_1^4\,p_2^2))\big/\big((D-2)\,(p_1\cdot p_2{}^2-p_1^2\,p_2^2)^2\big)+$$
$$(p_2^\mu\,p_1^\nu\,(-q^2\,p_1\cdot p_2{}^3+D\,q\cdot p_1\,q\cdot p_2\,p_1\cdot p_2{}^2-D\,q\cdot p_2{}^2\,p_1^2\,p_1\cdot p_2+q\cdot p_2{}^2\,p_1^2\,p_1\cdot p_2-$$
$$D\,q\cdot p_1{}^2\,p_2^2\,p_1\cdot p_2+q\cdot p_1{}^2\,p_2^2\,p_1\cdot p_2+q^2\,p_1^2\,p_2^2\,p_1\cdot p_2+D\,q\cdot p_1\,q\cdot p_2\,p_1^2\,p_2^2-2\,q\cdot p_1\,q\cdot p_2\,p_1^2\,p_2^2))/$$
$$\big((D-2)\,(p_1\cdot p_2{}^2-p_1^2\,p_2^2)^2\big)+(p_1^\mu\,p_2^\nu\,(-q^2\,p_1\cdot p_2{}^3+D\,q\cdot p_1\,q\cdot p_2\,p_1\cdot p_2{}^2-D\,q\cdot p_2{}^2\,p_1^2\,p_1\cdot p_2+$$
$$q\cdot p_2{}^2\,p_1^2\,p_1\cdot p_2-D\,q\cdot p_1{}^2\,p_2^2\,p_1\cdot p_2+q\cdot p_1{}^2\,p_2^2\,p_1\cdot p_2+q^2\,p_1^2\,p_2^2\,p_1\cdot p_2+$$
$$D\,q\cdot p_1\,q\cdot p_2\,p_1^2\,p_2^2-2\,q\cdot p_1\,q\cdot p_2\,p_1^2\,p_2^2))\big/\big((D-2)\,(p_1\cdot p_2{}^2-p_1^2\,p_2^2)^2\big)+$$
$$(p_1^\mu\,p_1^\nu\,(D\,p_1\cdot p_2{}^2\,q\cdot p_2{}^2-2\,p_1\cdot p_2{}^2\,q\cdot p_2{}^2+p_1^2\,p_2^2\,q\cdot p_2{}^2-2\,D\,q\cdot p_1\,p_1\cdot p_2\,p_2^2\,q\cdot p_2+2\,q\cdot p_1\,p_1\cdot p_2$$
$$p_2^2\,q\cdot p_2+D\,q\cdot p_1{}^2\,p_2^4-q\cdot p_1{}^2\,p_2^4-q^2\,p_1^2\,p_2^4+q^2\,p_1\cdot p_2{}^2\,p_2^2))\big/\big((D-2)\,(p_1\cdot p_2{}^2-p_1^2\,p_2^2)^2\big)$$

TLI ***unfinished***

Description

TLI[{v,w,x,y,z},{a,b,c,d,e}, {al,be,ga,de,ep}]. The exponents of the numerator scalar product are (dl = OPE-Delta): v: k1.k1, w: k2.k2, x: p.k1, y: p.k2, z: k1.k2. a: dl.k1, b: dl.k2, c: dl.(p-k1), d: dl.(p-k2), e: dl.(k1-k2).
TLI[{mu1, ...}, {nu1, ...}][{v,w,x,y,z},{a,b,c,d,e}, {al,be,ga,de,ep}] is a tensor integral, where mu1 belongs to k1 and nu1 to k2.
TLI[any___,{a,b,c,d,e,0,0}, {al,be,ga,de,ep}] simplifies to TLI[any, {a,b,c,d,e}, {al,be,ga,de,ep}].
TLI[{0,0,0,0,0},{a,b,c,d,e}, {al,be,ga,de,ep}] simplifies to TLI[{a,b,c,d,e}, {al,be,ga,de,ep}].

```
In[2024]:= C_μνρ - type
Out[2024]= TIDL[{{q, μ}, {q, ν}, {q, ρ}}, {p, k}];
```

See also: TLI2FC.

Examples

```
In[2025]:= C_μνρσ - type
```

TLIFP ***unfinished***

Description

TLIFP[exp] does Feynman-Parametrizations of TLI's in exp.

```
In[2026]:= TIDL[{{q, μ}, {q, ν}, {q, ρ}, {q, σ}}, {p, k}];
Out[2026]= D_μ - type
```

See also: TLI.

Examples

```
In[2027]:= TIDL[{q, μ}, {p1, p2, p3}]
```

TLIHYP ***unfinished***

Description

TLIHYP[exp] expresses TLI's in exp. in terms of hypergeometric functions, where possible.

```
In[2028]:= (p3^μ (q · p3 p1 · p2^2 - q · p2 p1 · p3 p1 · p2 -
                q · p1 p2 · p3 p1 · p2 - q · p3 p1^2 p2^2 + q · p1 p1 · p3 p2^2 + q · p2 p1^2 p2 · p3)) /
          (p3^2 p1 · p2^2 - 2 p1 · p3 p2 · p3 p1 · p2 + p1^2 p2 · p3^2 + p1 · p3^2 p2^2 - p1^2 p2^2 p3^2) -
         (p2^μ (-(q · p2) p1 · p3^2 + q · p3 p1 · p2 p1 · p3 + q · p1 p2 · p3 p1 · p3 -
                q · p3 p1^2 p2 · p3 + q · p2 p1^2 p3^2 - q · p1 p1 · p2 p3^2)) /
          (p3^2 p1 · p2^2 - 2 p1 · p3 p2 · p3 p1 · p2 + p1^2 p2 · p3^2 + p1 · p3^2 p2^2 - p1^2 p2^2 p3^2) +
         (p1^μ (q · p1 p2 · p3^2 - q · p3 p1 · p2 p2 · p3 - q · p2 p1 · p3 p2 · p3 +
                q · p3 p1 · p3 p2^2 + q · p2 p1 · p2 p3^2 - q · p1 p2^2 p3^2)) /
          (p3^2 p1 · p2^2 - 2 p1 · p3 p2 · p3 p1 · p2 + p1^2 p2 · p3^2 + p1 · p3^2 p2^2 - p1^2 p2^2 p3^2)
Out[2028]= D_μν - type
```

See also: TLI.

Examples

```
In[2029]:= TIDL[{{q, μ}, {q, ν}}, {p, k, r}];
```

TLI2

Description

like TLI, but without any functional properties.
See also: TLI.

TLI2FC ***unfinished***

Description

TLI2FC[exp] transforms all TLI-integrals in exp to the FAD form.

```
In[2030]:= D_μνρ - type
Out[2030]= TIDL[{{q, μ}, {q, ν}, {q, ρ}}, {p, k, r}];
```

See also: TLI. FC2TLI.

Examples

```
In[2031]:= D_μνρσ - type
```

ToDistribution

Description

ToDistribution[exp,x] replaces (1-x)^(a Epsilon - 1) in exp by 1/(a Epsilon) DeltaFunction[1-x] + 1/(1-x) + a Epsilon Log[1-x]/(1-x) + 1/2 a^2 Epsilon^2 Log[1-x]^2/(1-x)] and (1-x)^(a Epsilon - 2) in exp by -1/(a Epsilon) DeltaFunctionPrime[1-x] + 1/(1-x)^2 + (a Epsilon) Log[1-x]/(1-x)^2 + a^2 Epsilon^2/2 Log[1-x]^2/(1-x)^2 + a^3 Epsilon^3/6 Log[1-x]^3/(1-x)^2.

```
In[2032]:= TIDL[{{q, μ}, {q, ν}, {q, ρ}, {q, σ}}, {p, k, r, s}];
Out[2032]= {Length[%], LeafCount[%], ByteCount[%]}
```

See also: PlusDistribution.

Examples

```
In[2033]:= {4, 29, 424}
Out[2033]= Options[TimedIntegrate]

In[2034]:= {Timing → 10, Assumptions → ε > 0, Integrate → Integrate, Expand → True}
Out[2034]= TimedIntegrate[Log[x^5], {x, 0, 1}, Timing → 1]

In[2035]:= (∫_0^1 dx) . Log (x^5)
Out[2035]= TimedIntegrate[Log[Cos[x^5]], {x, 0, 1}, Timing → 10, Integrate → int]

In[2036]:= int (Log (Cos (x^5))), {x, 0, 1}, Assumptions → ε > 0)
Out[2036]= Options[TLI]

In[2037]:= {EpsilonOrder → 0, Momentum → p}
Out[2037]= {}
Out[2037]= Options[TLIFP]

In[2038]:= {FeynmanParameterNames → {x, t, u, s, y}, GammaExpand → True, Momentum → p, Print → True}
Out[2038]= {}
```

ToHypergeometric

Description

ToHypergeometric[, t] returns Options[TLIHYP]. Remember that Re b >0 and Re (c-b) > 0 should hold (need not be set in *Mathematica*).

See also: HypergeometricAC, HypergeometricIR, HypergeometricSE.

Examples

```
In[2039]:= {FeynmanParameterNames → x, Momentum → p}
Out[2039]= {}
Out[2039]= Options[TLI2FC]

In[2040]:= {Dimension → D, FeynCalcExternal → False, Momentum → {q1, q2, p}}
Out[2040]= {}
Out[2040]= Options[ToDistribution]
```

ToLarin

Description

ToLarin[exp] translates all {PlusDistribution → Identity} in exp into SetOptions[ToDistribution, PlusDistribution → PlusDistribution]

See also: Eps.

Examples

```
In[2041]:= {PlusDistribution → PlusDistribution}
Out[2041]= ToDistribution[(1 - x)^(Epsilon - 1), x]

In[2042]:= 1/6 (Log^3(1 - x)/(1 - x))+ ε^3 + 1/2 (Log^2(1 - x)/(1 - x))+ ε^2 + (Log(1 - x)/(1 - x))+ ε + (1/(1 - x))+ + δ(1 - x)/ε
Out[2042]= SetOptions[ToDistribution, PlusDistribution → Identity]
```

ToTFi

Description

ToTFi[expr, q1, q2, p] translates FeynCalc 2-loop self energy type integrals into the TFi notatation, which can be used as input for the function TarcerRecurse from the TARCER package. See TFi for details on the conventions.

See also: TFi, FromTFi.

Examples

```
In[2043]:= {PlusDistribution → Identity}
Out[2043]= ToDistribution[(1 - x)^(Epsilon - 2), x]

In[2044]:= ε^3 Log^3(1 - x)/(6 (1 - x)^2) + ε^2 Log^2(1 - x)/(2 (1 - x)^2) + ε Log(1 - x)/(1 - x)^2 - δ'(1 - x)/ε + 1/(1 - x)^2
Out[2044]= Series2[Integrate[(1 - x)^(Epsilon-2), {x, 0, 1}, GenerateConditions → False], Epsilon, 3]

In[2045]:= -ε^3 - ε^2 - ε - 1
Out[2045]= Options@Integrate

In[2046]:= {Assumptions :→ $Assumptions, GenerateConditions → Automatic, PrincipalValue → False}
Out[2046]= Integrate2[ToDistribution[(1 - x)^(Epsilon - 2), x], {x, 0, 1}]

In[2047]:= -ε^3 - ε^2 - ε - 1
Out[2047]= t^b (1 - t)^c (1 + t z)^a

In[2048]:= Γ(b + 1) Γ(c + 1)/Γ(b + c + 2)

             2F1(-a, b + 1; b + c + 2; -z)
Out[2048]= ToHypergeometric[t^b (1 - t)^c (1 + t z)^a, t]
```

Tr

Description

Tr[exp] calculates the Dirac trace of exp. Depending on the setting of the option SUNTrace also a trace over SU(*N*) objects is performed.

```
In[2049]:= Γ(b + 1) Γ(c + 1) 2F1(-a, b + 1; b + c + 2; -z)/Γ(b + c + 2)
Out[2049]= ToHypergeometric[w t^(b-1) (1 - t)^(c-b-1) (1 - t z)^-a, t]
```

See also: DiracSimplify, DiracTrace, FermionSpinSum, SUNTrace.

Examples

```
In[2050]:= w Γ(b) Γ(c - b) 2F1(a, b; c; z)/Γ(c)
Out[2050]= ToHypergeometric[t^b (1 - t)^c (u + t z)^a, t]

In[2051]:= u^a Γ(b + 1) Γ(c + 1) 2F1(-a, b + 1; b + c + 2; -z/u)/Γ(b + c + 2)
Out[2051]= ToHypergeometric[w t^(b-1) (1 - t)^(c-b-1) (u - t z)^-a, t]

In[2052]:= u^-a w Γ(b) Γ(c - b) 2F1(a, b; c; z/u)/Γ(c)
Out[2052]= γ^μ γ^5
```

```
In[2053]:= -I/6 ε^μνλσ γ^ν γ^λ γ^σ .
Out[2053]= DiracMatrix[ν].DiracMatrix[μ].DiracMatrix[5]

In[2054]:= γ^ν.γ^μ.γ^5
Out[2054]= ToLarin[%]

In[2055]:= -1/6 I γ^ν.γ^du1.γ^du2.γ^du3 ϵ^μdu1du2du3
Out[2055]= FAD[q1, q1 - p, {q2, M}, {q2 - p, m}, q1 - q2]

In[2056]:= 1/([q1^2]) ([(q1 - p)^2]) ([(q1 - q2)^2]) ([q2^2 - M^2]) ([(q2 - p)^2 - m^2])
ToTFi[%, q1, q2, p]
```

$West=False should be set before any calculation is done (because intermediate results from trace calculations are cached).

```
In[2057]:= F^(D)_{1,0}{1,M}{1,0}{1,m}{1,0}
Out[2057]= %//StandardForm

In[2058]:= TFi[D, m^2, {{1, 0}, {1, M}, {1, 0}, {1, m}, {1, 0}}]
Out[2058]= SOD[q1] SOD[q2]FAD[q1, q1 - p, {q2, M}, {q2 - p, m}, q1 - q2]//FCI

In[2059]:= Δ · q1 Δ · q2 / (q1^2 . (q1 - p)^2 . (q2^2 - M^2) . ((q2 - p)^2 - m^2) . (q1 - q2)^2)
Out[2059]= ToTFi[%]

In[2060]:= F^(D) 11_{1,0}{1,M}{1,0}{1,m}{1,0}
Out[2060]= %//StandardForm

In[2061]:= TFi[D, m^2, SOD[p], {1, 1}, {{1, 0}, {1, M}, {1, 0}, {1, m}, {1, 0}}]
Out[2061]= Options[Tr]

In[2062]:= {DiracTraceEvaluate → True, Factoring → False, FeynCalcExternal → False,
             LeviCivitaSign → -1, Mandelstam → {}, PairCollect → False, Schouten → 442,
             SUNTrace → False, TraceOfOne → 4, SUNNToCACF → False}
Out[2062]= GA[μ, ν]

In[2063]:= γ^μ.γ^ν
Out[2063]= Tr[%]

In[2064]:= 4 g^μν
Out[2064]= Tr[(GSD[p] + m).GAD[μ].(GSD[q] - m).GAD[ν]]

In[2065]:= 4 (-g^μν m^2 + q^μ p^ν + p^μ q^ν - g^μν p · q)
Out[2065]= Tr[GA[μ, ν, ρ, σ, 5]]

In[2066]:= -4 I ϵ^μνρσ
Out[2066]= Tr[GAD[μ, ν, ρ, σ, τ, ξ, 5]]
```

```
In[2067]:= 4 (i ε^(ξρστ) g^(μν) - i ε^(νρστ) g^(μξ) + i ε^(νξστ) g^(μρ) - i ε^(νξρτ) g^(μσ) +
              i ε^(νξρσ) g^(μτ) + i ε^(μρστ) g^(νξ) - i ε^(μξστ) g^(νρ) + i ε^(μξρτ) g^(νσ) - i ε^(μξρσ) g^(ντ) -
              i ε^(μνστ) g^(ξρ) + i ε^(μνρτ) g^(ξσ) - i ε^(μνρσ) g^(ξτ) + i ε^(μνξτ) g^(ρσ) - i ε^(μνξσ) g^(ρτ) + i ε^(μνξρ) g^(στ))
Out[2067]= $West

In[2068]:= True
Out[2068]= ?$West

In[2069]:= If $West is set to True (which is the default), traces involving
           more than 4 Dirac matrices and gamma5 are calculated recursively
           according to formula (A.5) from Comp. Phys. Comm 77 (1993) 286-
           298, which is based on the Breitenlohner Maison gamma5 - scheme.
Out[2069]= $West = False

In[2070]:= False
```

TraceDimension

Description

TraceDimension is an option for FeynCalc2FORM. If set to 4: trace, if set to n: tracen.
See also: FeynCalc2FORM.

TraceOfOne

Description

TraceOfOne is an option for Tr and DiracTrace. Its setting determines the value of the unit trace.
See also: Tr, DiracTrace.

Examples

```
In[2071]:= Tr[GAD[ν1, ν2, ν3, ν4, ν5, ν6].GA[5]]
Out[2071]= 0

In[2072]:= Tr[GS[p, q, r, s]]
Out[2072]= 4 (p · s q · r - p · r q · s + p · q r · s)
```

Trick

Description

Trick[exp] performs several basic simplifications without expansion. Trick[exp] uses Contract, DotSimplify and SUNDeltaContract.
See also: Calc, Contract, DiracTrick, DotSimplify, DiracTrick.

Tr[(GS[p] + m).GA[μ].(GS[q] + m).GA[μ], Factoring → True]

This calculates 8 ($2\,m^2 - p \cdot q$)and Tr[GA[α, β], FCE → True] in D dimensions.

```
In[2073]:= 4 g^αβ
Out[2073]= %//StandardForm

In[2074]:= 4 MT[α, β]
Out[2074]= Tr[GAD[μ].GA[α].GA[β].GAD[μ], FCE → True]

In[2075]:= 4 D g^αβ
Out[2075]= %//StandardForm

In[2076]:= 4 D MT[α, β]
Out[2076]= t = GA[μ, ν] SUNT[b].SUNT[c] SUNDelta[c, b]

In[2077]:= γ^μ.γ^ν T_b.T_c δ_bc
Out[2077]= Tr[t, SUNTrace → False, SUNNToCACF → True]
```

TrickIntegrate

Description

TrickIntegrate[(1-t)^(a Epsilon -1) g[t], t] does an integration trick for the definite integral of ((1-t)^(a Epsilon -1) g[t]) from 0 to 1. TrickIntegrate[(1-t)^(a Epsilon -1) g[t], t] gives g[1]/a/Epsilon + Hold[Integrate][(1-t)^(a Epsilon-1) (g[t]-g[1]),{t,0,1}]. TrickIntegrate[t^(a Epsilon -1) f[t], t] gives f[0]/a/Epsilon + Hold[Integrate][t^(a Epsilon-1) (f[t]-f[0]),{t,0,1}], provided g[1] and f[0] do exist.

Examples

```
In[2078]:= 4 C_F g^μν
Out[2078]= Tr[t, SUNTrace → True, SUNNToCACF → True]
```

TrickMandelstam

Description

TrickMandelstam[expr, {s, t, u, 4 C_A C_F $g^{\mu\nu}$ + Tr[1, SUNTrace → False, SUNNToCACF → True]+ 4+ Tr[1, SUNTrace → True, SUNNToCACF → True]}] simplifies all sums in expr so that one of the Mandelstam variables s, t or u is eliminated by the relation $s + t + u = 4\ C_A$ + Clear[t];+ Tr[1, SUNTrace → False, SUNNToCACF → True, TraceOfOne → tr1]+ tr1 . The trick is that the resulting sum has the most short number of terms.

Examples

```
In[2079]:= Tr[1, SUNTrace → True, SUNNToCACF → True, TraceOfOne → tr2]
Out[2079]= C_A tr2

In[2080]:= Examples
Out[2080]= g^μν γ_μ
```

Tr2

Description

If exp contains DiracTrace's, Tr2[exp] simplifies exp and does the Dirac traces unless more that 4 gamma matrices and DiracGamma[5] occur. Tr2[exp] also separates the color-strucure, and takes the color trace if Tf occurs in exp. If exp does not contain DiracTrace's, Tr2[exp] takes the Dirac trace.

```
In[2081]:= g^ν_ν
Out[2081]= Trick[{GA[μ] MT[μ, ν], MTD[ν, ν]}]
```

See also: Tr, Tf, DiracTrace, DiracSimplify, SUNTrace.

Examples

```
In[2082]:= {γ^ν, D}
Out[2082]= FV[p + r, μ] MT[μ, ν] FV[q - p, ν]

In[2083]:= (q - p^ν) (p + r^μ) g^μν
Out[2083]= Trick[%]

In[2084]:= -p^2 + p · q - p · r + q · r
Out[2084]= Trick[c.b.a . GA[d].GA[e]]

In[2085]:= a b c γ^d.γ^e
Out[2085]= %//FCE//StandardForm

In[2086]:= a b c GA[d].GA[e]
```

```
Out[2086]= TrickIntegrate[(1 - t)^(a Epsilon - 1) g[t], t]

In[2087]:= lim g(1 - t) / (a ε) + Hold[Integrate][t^(a ε-1) (g(1 - t) - lim g(1 - t)), {t, 0, 1}]
Out[2087]= m1^2
```

Twist2AlienOperator ***unfinished*** (unclear usage def.)

Description

Twist2AlienOperator[p, 0] : (7); Twist2AlienOperator[p1,p2,{p3,mu,a}, 0] (p1: incoming quark momentum, p3: incoming gluon (count1)).

Examples

```
In[2088]:= m2^2
Out[2088]= m3^2

In[2089]:= m4^2
Out[2089]= m1^2
```

Twist2CounterOperator ***unfinished*** (unclear usage def.)

Description

Twist2CounterOperator[p,mu,nu,a,b,5]; Twist2CounterOperator[p, 7] : (7); Twist2CounterOperator[p1,p2,{p3,mu,a}, 1] (p1: incoming quark momentum, p3: incoming gluon (count1)).

Examples

```
In[2090]:= m2^2
Out[2090]= m3^2

In[2091]:= m4^2
Out[2091]= TrickMandelstam[(s + t - u) (2MW^2 - t - u), {s, t, u, 2MW^2}]//Factor2

In[2092]:= -2 s (u - MW^2)
Out[2092]= TrickMandelstam[M^2 s - s^2 + M^2 t - st + M^2 u - su, {s, t, u, 2M^2}]
```

Twist2GluonOperator

Description

Twist2GluonOperator[{p, mu, a}, {nu, b}] or Twist2GluonOperator[p, {mu, a}, {nu, b}] or Twist2GluonOperator[p, mu,a, nu,b] yields the 2-gluon operator (p is ingoing momentum corresponding to Lorentz index mu). Twist2GluonOperator[{p,mu,a}, {q,nu,b}, {k,la,c}] or Twist2GluonOperator[p,mu,a , q,nu,b , k,la,c] gives the 3-gluon operator. Twist2GluonOperator[{p,mu,a}, {q,nu,b}, {k,la,c}, {s,si,d}] or Twist2GluonOperator[p,mu,a , q,nu,b , k,la,c , s,si,d] yields the 4-Gluon operator. The dimension is determined by the option and Dimension. The setting of the option Polarization (unpolarized: 0; polarized: 1) determines whether the unpolarized or polarized feynman rule is returned. With the setting Explicit to False the color-structure and the (1+(-1)^OPEm) (for polarized: (1-(-1)^OPEm)) is extracted and the color indices are omitted in the arguments of Twist2GluonOperator.

See also: Twist2QuarkOperator.

Polarized case, zero-momentum insertion

```
In[2093]:= 2 M^2 (M^2 - s)
Out[2093]= Options[Tr2]
```

2-gluon legs

```
In[2094]:= {Factoring → False}
Out[2094]= DiracTrace[a]//Tr2

In[2095]:= 4 a
Out[2095]= Tr[a]

In[2096]:= 4 a
Out[2096]= DiracTrace[a[SUNIndex[i]]Tf]//Tr2

In[2097]:= 4 C_A T_f a(i)
Out[2097]= DiracTrace[a[SUNIndex[i]]]//Tr2
```

3-gluon legs

```
In[2098]:= 4 a(i)
Out[2098]= DiracTrace[a Tf]//Tr2

In[2099]:= 4 a T_f
Out[2099]= Tr[a Tf, SUNTrace → True]

In[2100]:= 4 a C_A T_f
Out[2100]= Twist2AlienOperator[p, 0]
```

4-gluon legs

$$\text{In[2101]:=}\ \frac{2\,C_F\,g_s^2\left(\frac{2}{m}-\frac{1}{m+1}-\frac{2}{m-1}\right)S_n\,\gamma\cdot\Delta\,(\Delta\cdot p)^{m-1}}{\varepsilon}$$

```
Out[2101]= Twist2AlienOperator[p1, p2, {p3, μ, a}, 0]
```

```
In[2102]:= 1/ε (i (1 + (-1)^m) g_s^3 S_n T_a . (γ · Δ) Δ^μ
                ((1/(m - 1) - 1/m) (-(Δ · p_3))^(m-2) + (2/m - 1/(m + 1) - 2/(m - 1)) ((Δ · p_1)^(m-1)/(Δ · p_1 + Δ · p_3) - (-(Δ · p_3))^(m-1)/(Δ · p_1 + Δ · p_3))))
Out[2102]= Twist2CounterOperator[p, μ, ν, a, b, 5]

In[2103]:= -1/(2 ε) ((1 + (-1)^m) C_A g_s^2 S_n
                ((8/m - 24/(m + 1) + 24/(m + 2) - 4/(m - 1)) Δ^μ Δ^ν p^2 (Δ · p)^(m-2) + (-6/m + 16/(m + 1) - 16/(m + 2) + 6/(m - 1))
                (p^μ Δ^ν + Δ^μ p^ν) (Δ · p)^(m-1) + (8/m - 12/(m + 1) + 8/(m + 2) - 8/(m - 1)) g^(μν) (Δ · p)^m) δ_ab)
Out[2103]= Twist2CounterOperator[p, 7]
```

Unpolarized case, zero-momentum insertion

```
In[2104]:= ((1 + (-1)^m) C_F g_s^2 (2/m - 1/(m+1) - 2/(m-1)) S_n γ · Δ (Δ · p)^(m-1))/ε
Out[2104]= Twist2CounterOperator[p1, p2, {p3, μ, a}, 1]
```

2-gluon legs

```
In[2105]:= -((1 + (-1)^m) (C_A - 2 C_F) g_s^3 (2/m - 1/(m+1) - 2/(m-1)) S_n γ · Δ Δ^μ ((Δ·p_1)^(m-1)/(Δ·p_1+Δ·p_2) - (-(Δ·p_2))^(m-1)/(Δ·p_1+Δ·p_2)) T_a)/(2 ε)
Out[2105]= SetOptions[Twist2GluonOperator, Polarization → 1,
             Explicit → False, ZeroMomentumInsertion → True]

In[2106]:= {CouplingConstant → g_s, Dimension → D, Polarization → 1, Explicit → False, ZeroMomentumInsertio
True}
Out[2106]= Twist2GluonOperator[p, {μ, a}, {ν, b}]

In[2107]:= i (1 - (-1)^m) δ_ab (O^G2_μν(p))
Out[2107]= Twist2GluonOperator[p, {μ, a}, {ν, b}, Explicit → True]

In[2108]:= i ε^(μνΔp) (1 - (-1)^m) (Δ · p)^(m-1) δ_ab
Out[2108]= Twist2GluonOperator[p, {μ}, {ν}]
```

3-gluon legs

```
In[2109]:= O^G2_μν(p)
Out[2109]= Twist2GluonOperator[p, {μ}, {ν}, Explicit → True]

In[2110]:= ε^(μνΔp) (Δ · p)^(m-1)
Out[2110]= Twist2GluonOperator[{p, μ, a}, {q, ν, b}, {r, ρ, c}]

In[2111]:= g_s (1 - (-1)^m) f_abc (O^G3_νρμ(q, r, p))
Out[2111]= Twist2GluonOperator[{p, μ}, {q, ν}, {r, ρ}]
```

4-gluon legs

```
In[2112]:= O^G3_νρμ(q, r, p)
```

```
Out[2112]= Twist2GluonOperator[{p, μ}, {q, ν}, {r, ρ}, Explicit → True]

In[2113]:= -(ϵ^(μρpΔ) Δ^ν - ϵ^(μνpΔ) Δ^ρ) (Δ · p)^(m-2) - (ϵ^(νρΔq) Δ^μ + ϵ^(μνΔq) Δ^ρ) (Δ · q)^(m-2) -
             (ϵ^(νρΔr) Δ^μ - ϵ^(μρΔr) Δ^ν) (Δ · r)^(m-2) + (Σ_(i=0)^(m-3) (-1)^i (Δ · p)^i (Δ · q)^(-i+m-3)) Δ^ρ (ϵ^(νpΔq) Δ^μ + ϵ^(μνΔq) Δ · p) -
             (Σ_(i=0)^(m-3) (-1)^i (Δ · p)^i (Δ · r)^(-i+m-3)) Δ^ν (ϵ^(ρpΔr) Δ^μ + ϵ^(μρΔr) Δ · p) -
             (Σ_(i=0)^(m-3) (-1)^i (Δ · q)^i (Δ · r)^(-i+m-3)) Δ^μ (ϵ^(ρΔqr) Δ^ν - ϵ^(νρΔr) Δ · q)
Out[2113]= Twist2GluonOperator[{p, μ, a}, {q, ν, b}, {r, ρ, c}, {s, σ, d}]

In[2114]:= i (1 - (-1)^m) g_s^2
             ( - f_(abc135) f_(cc135d) (O^(G4)_(μνρσ)(p, q, r, s)) - f_(acc135) f_(bc135d) (O^(G4)_(μρνσ)(p, r, q, s)) + f_(ac135d) f_(bcc135) (O^(G4)_(ρνμσ)(r, q, p, s)
Out[2114]= Twist2GluonOperator[{p, μ}, {q, ν}, {r, ρ}, {s, σ}]
```

Advanced Examples

The setting All for Explicit performs the sums.

```
In[2115]:= O^(G4)_(μνρσ)(p, q, r, s)

In[2116]:= Twist2GluonOperator[{p, μ}, {q, ν}, {r, ρ}, {s, σ}, Explicit → True]
```

Polarized case, non-zero-momentum insertion

```
In[2117]:= (ϵ^(Δνρσ) Δ^μ - ϵ^(Δμρσ) Δ^ν) (Δ · r + Δ · s)^(m-2) - (Σ_(i=0)^(m-3) (-(Δ · p) - Δ · q)^i (Δ · s)^(-i+m-3)) (ϵ^(νσΔs) Δ^μ - ϵ^(μσΔs) Δ^ν) Δ^ρ +
           (Σ_(i=0)^(m-3) (Δ · p + Δ · q)^(-i+m-3) (-(Δ · r))^i) (ϵ^(ρνrΔ) Δ^μ - ϵ^(ρμrΔ) Δ^ν) Δ^σ -
           (Σ_(i=0)^(m-3) (-(Δ · p))^i (Δ · r + Δ · s)^(-i+m-3)) Δ^ν (ϵ^(μσΔp) Δ^ρ - ϵ^(μρΔp) Δ^σ) +
           (Σ_(i=0)^(m-3) (-(Δ · q))^i (Δ · r + Δ · s)^(-i+m-3)) Δ^μ (ϵ^(νσΔq) Δ^ρ - ϵ^(νρΔq) Δ^σ) -
           (Σ_(j=0)^(m-4) (j + 1) (Δ · p)^(-j+m-4) (-(Δ · r) - Δ · s)^(j-i) (-(Δ · s))^i) Δ^ν Δ^ρ (-ϵ^(Δσps) Δ^μ - ϵ^(μσΔs) Δ · p) +
           (Σ_(j=0)^(m-4) (j + 1) (Δ · q)^(-j+m-4) (-(Δ · r) - Δ · s)^(j-i) (-(Δ · s))^i) Δ^μ Δ^ρ (-ϵ^(Δσqs) Δ^ν - ϵ^(νσΔs) Δ · q) +
           (Σ_(j=0)^(m-4) (j + 1) (Δ · p)^(-j+m-4) (-(Δ · r))^i (-(Δ · r) - Δ · s)^(j-i)) Δ^ν Δ^σ (-ϵ^(Δμpr) Δ^ρ - ϵ^(μρΔp) Δ · r) -
           (Σ_(j=0)^(m-4) (j + 1) (Δ · q)^(-j+m-4) (-(Δ · r))^i (-(Δ · r) - Δ · s)^(j-i)) Δ^μ Δ^σ (-ϵ^(Δνqr) Δ^ρ - ϵ^(νρΔq) Δ · r)
Out[2117]= SetOptions[Twist2GluonOperator, Polarization → 0, ZeroMomentumInsertion → True]
```

2-gluon legs

```
In[2118]:= {CouplingConstant → g_s, Dimension → D, Polarization → 0, Explicit → False, ZeroMomentumInsertio
           True}
Out[2118]= Twist2GluonOperator[p, {μ, a}, {ν, b}]
```

3-gluon legs

```
In[2119]:= 1/2 ((-1)^m + 1) δ_ab (O^G2_μν(p))
Out[2119]= Twist2GluonOperator[p, {μ, a}, {ν, b}, Explicit → True]
```

4-gluon legs

```
In[2120]:= 1/2 (g^μν Δ · p^2 - (p^μ Δ^ν + Δ^μ p^ν) Δ · p + Δ^μ Δ^ν p^2) ((-1)^m + 1) (Δ · p)^(m-2) δ_ab
Out[2120]= Twist2GluonOperator[p, {μ}, {ν}]
```

Unpolarized case, non-zero-momentum insertion

```
In[2121]:= O^G2_μν(p)
Out[2121]= Twist2GluonOperator[p, {μ}, {ν}, Explicit → True]
```

2-gluon legs

```
In[2122]:= (g^μν Δ·p^2 - (p^μ Δ^ν + Δ^μ p^ν) Δ·p + Δ^μ Δ^ν p^2) (Δ·p)^(m-2)
Out[2122]= Twist2GluonOperator[{p, μ, a}, {q, ν, b}, {r, ρ, c}]
```

3-gluon legs

```
In[2123]:= -(1/2) i g_s ((-1)^m + 1) f_abc (O^G3_νρμ(q, r, p))
In[2124]:= Twist2GluonOperator[{p, μ}, {q, ν}, {r, ρ}]
Out[2124]= O^G3_νρμ(q, r, p)
In[2125]:= Twist2GluonOperator[{p, μ}, {q, ν}, {r, ρ}, Explicit → True]
Out[2125]= (Sum_{i=0}^{m-3} (-(Δ·p))^i (Δ·q)^(-i+m-3)) Δ^ρ (q^μ Δ^ν Δ·p - g^μν Δ·q Δ·p + Δ^μ p^ν Δ·q - Δ^μ Δ^ν p·q) +
           (Sum_{i=0}^{m-3} (Δ·p)^(-i+m-3) (-(Δ·r))^i) Δ^ν (r^μ Δ^ρ Δ·p - g^μρ Δ·r Δ·p + Δ^μ p^ρ Δ·r - Δ^μ Δ^ρ p·r) +
           (Sum_{i=0}^{m-3} (-(Δ·q))^i (Δ·r)^(-i+m-3)) Δ^μ (r^ν Δ^ρ Δ·q - g^νρ Δ·r Δ·q + Δ^ν q^ρ Δ·r - Δ^ν Δ^ρ q·r) +
           (Δ^μ (p^ν Δ^ρ - Δ^ν p^ρ) + (g^μρ Δ^ν - g^μν Δ^ρ) Δ·p) (Δ·p)^(m-2) +
           (Δ^ν (Δ^μ q^ρ - q^μ Δ^ρ) + (g^μν Δ^ρ - Δ^μ g^νρ) Δ·q) (Δ·q)^(m-2) + ((r^μ Δ^ν - Δ^μ r^ν) Δ^ρ + (Δ^μ g^νρ - g^μρ Δ^ν) Δ·r) (Δ·r)^(m-2
```

4-gluon legs

```
In[2126]:= Twist2GluonOperator[{p, μ, a}, {q, ν, b}, {r, ρ, c}, {s, σ, d}]
Out[2126]= (1/2) (1 + (-1)^m) g_s^2
           ( - f_abc136 f_cc136d (O^G4_μνρσ(p, q, r, s)) - f_acc136 f_bc136d (O^G4_μρνσ(p, r, q, s)) + f_ac136d f_bcc136 (O^G4_ρνμσ(r, q, p, s)
```

Suppress the lengthy output.

```
In[2127]:= Twist2GluonOperator[{p, μ}, {q, ν}, {r, ρ}, {s, σ}]
Out[2127]= O^G4_μνρσ(p, q, r, s)
```

The setting Explicit → All performs the sums.

```
In[2128]:= Twist2GluonOperator[{p, μ}, {q, ν}, {r, ρ}, {s, σ}, Explicit → True]
```

```
In[2129]:= (Δ^μ g^νσ Δ^ρ - g^μσ Δ^ν Δ^ρ - Δ^μ g^νρ Δ^σ + g^μρ Δ^ν Δ^σ) (Δ · r + Δ · s)^(m-2) +
           (Σ_{i=0}^{m-3} (-(Δ · p))^i (Δ · r + Δ · s)^(-i+m-3)) Δ^ν (Δ^μ p^ρ Δ^σ - g^μρ Δ · p Δ^σ - Δ^μ Δ^ρ p^σ + g^μσ Δ^ρ Δ · p) -
           (Σ_{i=0}^{m-3} (-(Δ · q))^i (Δ · r + Δ · s)^(-i+m-3)) Δ^μ (Δ^ν q^ρ Δ^σ - g^νρ Δ · q Δ^σ - Δ^ν Δ^ρ q^σ + g^νσ Δ^ρ Δ · q) +
           (Σ_{i=0}^{m-3} (Δ · p + Δ · q)^(-i+m-3) (-(Δ · r))^i) Δ^σ (r^μ Δ^ν Δ^ρ - Δ^μ r^ν Δ^ρ + Δ^μ g^νρ Δ · r - g^μρ Δ^ν Δ · r) +
           (Σ_{i=0}^{m-3} (-(Δ · p) - Δ · q)^i (Δ · s)^(-i+m-3)) Δ^ρ (s^μ Δ^ν Δ^σ - Δ^μ s^ν Δ^σ + Δ^μ g^νσ Δ · s - g^μσ Δ^ν Δ · s) -
           (Σ_{j=0}^{m-4} (j + 1) (Δ · p)^(-j+m-4) (-(Δ · r))^i (-(Δ · r) - Δ · s)^(j-i))
             Δ^ν Δ^σ (-r^μ Δ^ρ Δ · p + g^μρ Δ · r Δ · p - Δ^μ p^ρ Δ · r + Δ^μ Δ^ρ p · r) +
           (Σ_{j=0}^{m-4} (j + 1) (Δ · p)^(-j+m-4) (-(Δ · r) - Δ · s)^(j-i) (-(Δ · s))^i) Δ^ν Δ^ρ
             (-s^μ Δ^σ Δ · p + g^μσ Δ · s Δ · p - Δ^μ p^σ Δ · s + Δ^μ Δ^σ p · s) +
           (Σ_{j=0}^{m-4} (j + 1) (Δ · q)^(-j+m-4) (-(Δ · r))^i (-(Δ · r) - Δ · s)^(j-i)) Δ^μ Δ^σ
             (-r^ν Δ^ρ Δ · q + g^νρ Δ · r Δ · q - Δ^ν q^ρ Δ · r + Δ^ν Δ^ρ q · r) - (Σ_{j=0}^{m-4} (j + 1) (Δ · q)^(-j+m-4) (-(Δ · r) - Δ · s)^(j-i) (-(Δ
             Δ^μ Δ^ρ (-s^ν Δ^σ Δ · q + g^νσ Δ · s Δ · q - Δ^ν q^σ Δ · s + Δ^ν Δ^σ q · s)
Out[2129]= SetOptions[Twist2GluonOperator, Polarization → 1, Explicit → All];
```

Twist2QuarkOperator

Description

Twist2QuarkOperator[p] or Twist2QuarkOperator[p,_,_] yields the quark-antiquark operator (p is momentum in the direction of the incoming quark). Twist2QuarkOperator[{p,q}] yields the 2-quark operator for non-zero momentum insertion (p is momentum in the direction of the incoming quark). Twist2QuarkOperator[{p1,___}, {p2,___}, {p3, mu, a}] or Twist2QuarkOperator[p1,_,_, p2,_,_, p3,mu,a] is the quark-antiquark-gluon operator, where p1 is the incoming quark, p2 the incoming antiquark and p3 denotes the incoming gluon momentum. Twist2QuarkOperator[{p1}, {p2}, {p3, mu, a}, {p4, nu, b}] or Twist2QuarkOperator[{p1,___}, {p2,___}, {p3, mu, a}, {p4, nu, b}] or Twist2QuarkOperator[p1,_,_, p2,_,_, p3,mu,a, p4, nu, b] gives the Quark-Quark-Gluon-Gluon-operator. The setting of the option Polarization (unpolarized: 0; polarized: 1) determines whether the unpolarized or polarized operator is returned

See also: Twist2GluonOperator.

Examples

Polarized case, zero-momentum insertion

```
In[2130]:= Twist2GluonOperator[{p, μ, a}, {q, ν, b}, {r, ρ, c}];
Out[2130]= SetOptions[Twist2GluonOperator, Polarization → 1, ZeroMomentumInsertion → False]
```

Quark-antiquark operator.

```
In[2131]:= {CouplingConstant → g_s, Dimension → D, Polarization → 1, Explicit → All, ZeroMomentumInsertion →
False}
Out[2131]= Twist2GluonOperator[{p, q}, {μ, a}, {ν, b}]
```

Quark-antiquark-gluon operator.

$$\text{In[2132]:=}\ \frac{1}{2}\, i\, (1-(-1)^m)\, ((\epsilon^{\nu\Delta pq}\,\Delta^\mu-\epsilon^{\mu\nu\Delta q}\,\Delta\cdot p)\,(\Delta\cdot p)^{m-2}+(\epsilon^{\mu\nu\Delta p}\,\Delta\cdot q-\epsilon^{\mu\Delta pq}\,\Delta^\nu)\,(\Delta\cdot q)^{m-2})\,\delta_{ab}$$

```
Out[2132]= Twist2GluonOperator[{p, μ, a}, {q, ν, b}, {r, ρ, c}]
```

Quark-antiquark-gluon-gluon operator.

```
In[2133]:= g_s ( - ε^(νρΔp) Δ^μ (Δ · p)^(m-2) + ε^(μρΔq) Δ^ν (Δ · q)^(m-2) - ε^(μνΔr) Δ^ρ (Δ · r)^(m-2) -
    ε^(ρΔpr) (Σ_(i=0)^(m-3) (Δ · p)^(-i+m-3) (Δ · p + Δ · q)^i) Δ^μ Δ^ν + ε^(ρΔqr) (Σ_(i=0)^(m-3) (Δ · q)^(-i+m-3) (Δ · p + Δ · q)^i) Δ^μ Δ^ν +
    ε^(νΔpq) (Σ_(i=0)^(m-3) (Δ · p)^(-i+m-3) (Δ · p + Δ · r)^i) Δ^μ Δ^ρ + ε^(νΔqr) (Σ_(i=0)^(m-3) (Δ · r)^(-i+m-3) (Δ · p + Δ · r)^i) Δ^μ Δ^ρ +
    ε^(μΔpq) (Σ_(i=0)^(m-3) (Δ · q)^(-i+m-3) (Δ · q + Δ · r)^i) Δ^ν Δ^ρ - ε^(μΔpr) (Σ_(i=0)^(m-3) (Δ · r)^(-i+m-3) (Δ · q + Δ · r)^i) Δ^ν Δ^ρ +
    1/(Δ · p + Δ · q)^2 (ε^(μρΔr) Δ^ν ((Δ · p + Δ · q)^m + (Σ_(i=0)^(m-3) (Δ · p)^(-i+m-3) (Δ · p + Δ · q)^i) Δ · p^3 +
        (Σ_(i=0)^(m-3) (Δ · p)^(-i+m-3) (Δ · p + Δ · q)^i) Δ · p Δ · q^2 + 2 (Σ_(i=0)^(m-3) (Δ · p)^(-i+m-3) (Δ · p + Δ · q)^i) Δ · p^2 Δ · q
    1/(Δ · p + Δ · q)^2 (ε^(νρΔr) Δ^μ ((Δ · p + Δ · q)^m + (Σ_(i=0)^(m-3) (Δ · q)^(-i+m-3) (Δ · p + Δ · q)^i) Δ · q^3 +
        2 (Σ_(i=0)^(m-3) (Δ · q)^(-i+m-3) (Δ · p + Δ · q)^i) Δ · p Δ · q^2 + (Σ_(i=0)^(m-3) (Δ · q)^(-i+m-3) (Δ · p + Δ · q)^i) Δ · p^2 Δ · q
    1/(Δ · p + Δ · r)^2 (ε^(μνΔq) Δ^ρ ((Δ · p + Δ · r)^m + (Σ_(i=0)^(m-3) (Δ · p)^(-i+m-3) (Δ · p + Δ · r)^i) Δ · p^3 +
        (Σ_(i=0)^(m-3) (Δ · p)^(-i+m-3) (Δ · p + Δ · r)^i) Δ · p Δ · r^2 + 2 (Σ_(i=0)^(m-3) (Δ · p)^(-i+m-3) (Δ · p + Δ · r)^i) Δ · p^2 Δ · r
    1/(Δ · p + Δ · r)^2 (ε^(νρΔq) Δ^μ ((Δ · p + Δ · r)^m + (Σ_(i=0)^(m-3) (Δ · r)^(-i+m-3) (Δ · p + Δ · r)^i) Δ · r^3 +
    2 (Σ_(i=0)^(m-3) (Δ · r)^(-i+m-3) (Δ · p + Δ · r)^i) Δ · p Δ · r^2 + (Σ_(i=0)^(m-3) (Δ · r)^(-i+m-3) (Δ · p + Δ · r)^i) Δ · p^2 Δ · r)) -
1/(Δ · q + Δ · r)^2 (ε^(μνΔp) Δ^ρ ((Δ · q + Δ · r)^m + (Σ_(i=0)^(m-3) (Δ · q)^(-i+m-3) (Δ · q + Δ · r)^i) Δ · q^3 +
    (Σ_(i=0)^(m-3) (Δ · q)^(-i+m-3) (Δ · q + Δ · r)^i) Δ · q Δ · r^2 + 2 (Σ_(i=0)^(m-3) (Δ · q)^(-i+m-3) (Δ · q + Δ · r)^i) Δ · q^2 Δ · r)) +
1/(Δ · q + Δ · r)^2 (ε^(μρΔp) Δ^ν ((Δ · q + Δ · r)^m + (Σ_(i=0)^(m-3) (Δ · r)^(-i+m-3) (Δ · q + Δ · r)^i) Δ · r^3 +
    2 (Σ_(i=0)^(m-3) (Δ · r)^(-i+m-3) (Δ · q + Δ · r)^i) Δ · q Δ · r^2 + (Σ_(i=0)^(m-3) (Δ · r)^(-i+m-3) (Δ · q + Δ · r)^i) Δ · q^2 Δ · r)) -
ε^(μνρΔ) (Δ · q Δ · r (Δ · p)^m + Δ · q (Δ · r)^m Δ · p + (Δ · q)^m Δ · r Δ · p)) / (Δ · q Δ · r Δ · p) ) f_abc
```

```
Out[2133]= Twist2GluonOperator[{p, μ, a}, {q, ν, b}, {r, ρ, c}, {s, σ, d}, Explicit → False]
```

Unpolarized case, zero-momentum insertion

```
In[2134]:= i (1 - (-1)^m) g_s^2
             ( - f_abc137 f_cc137d (O^G4_μνρσ(p, q, r, s)) - f_acc137 f_bc137d (O^G4_μρνσ(p, r, q, s)) + f_ac137d f_bcc137 (O^G4_ρνμσ(r, q, p, s)
Out[2134]= SetOptions[Twist2GluonOperator, Polarization → 0, ZeroMomentumInsertion → False]
```

Quark-antiquark operator.

```
In[2135]:= {CouplingConstant → g_s, Dimension → D, Polarization → 0, Explicit → All, ZeroMomentumInsertion →
False}
Out[2135]= Twist2GluonOperator[{p, q}, {μ, a}, {ν, b}]
```

Quark-antiquark-gluon operator.

```
In[2136]:= -1/2 (1 + (-1)^m) (q^μ Δ^ν Δ · p - g^μν Δ · q Δ · p + Δ^μ (p^ν Δ · q - Δ^ν p · q)) ((Δ · p)^(m-2) + (Δ · q)^(m-2)) δ_ab
Out[2136]= Twist2GluonOperator[{p, μ, a}, {q, ν, b}, {r, ρ, c}];
```

Quark-antiquark-gluon-gluon operator.

```
In[2137]:= Twist2GluonOperator[{p, μ}, {q, ν}, {r, ρ}]
Out[2137]= -1/2 (1 + (-1)^m) (μ^q Δ^r Δ · p - g^qr Δ · μ Δ · p + Δ^q (p^r Δ · μ - Δ^r p · μ)) ((Δ · p)^(m-2) + (Δ · μ)^(m-2)) δ_νρ
```

This shows the FeynCalcExternal (FCE) form.

```
In[2138]:= Twist2GluonOperator[{p, μ}, {q, ν}, {r, ρ}, Explicit → True]
Out[2138]= -1/2 (1 + (-1)^m) (μ^q Δ^r Δ · p - g^qr Δ · μ Δ · p + Δ^q (p^r Δ · μ - Δ^r p · μ)) ((Δ · p)^(m-2) + (Δ · μ)^(m-2)) δ_νρ
In[2139]:= Twist2GluonOperator[{p, μ, a}, {q, ν, b}, {r, ρ, c}, {s, σ, d}, Explicit → False]
Out[2139]= 1/2 (1 + (-1)^m) g_s^2
             ( - f_abc138 f_cc138d (O^G4_μνρσ(p, q, r, s)) - f_acc138 f_bc138d (O^G4_μρνσ(p, r, q, s)) + f_ac138d f_bcc138 (O^G4_ρνμσ(r, q, p, s)
In[2140]:= Short[Twist2GluonOperator[{p, μ}, {q, ν}, {r, ρ}, {s, σ}, Explicit → True], 6]
```

```
Out[2140]= (Δ^μ g^νσ Δ^ρ - g^μσ Δ^ν Δ^ρ - Δ^μ g^νρ Δ^σ + g^μρ Δ^ν Δ^σ) (Δ · r + Δ · s)^(m-2) +

  (Σ_{i=0}^{m-3} (-(Δ · p))^i (Δ · r + Δ · s)^(-i+m-3)) Δ^ν (Δ^μ p^ρ Δ^σ - g^μρ Δ · p Δ^σ - Δ^μ Δ^ρ p^σ + g^μσ Δ^ρ Δ · p) -

  (Σ_{i=0}^{m-3} (-(Δ · q))^i (Δ · r + Δ · s)^(-i+m-3)) Δ^μ (Δ^ν q^ρ Δ^σ - g^νρ Δ · q Δ^σ - Δ^ν Δ^ρ q^σ + g^νσ Δ^ρ Δ · q) +

  (Σ_{i=0}^{m-3} (Δ · p + Δ · q)^(-i+m-3) (-(Δ · r))^i) Δ^σ (r^μ Δ^ν Δ^ρ - Δ^μ r^ν Δ^ρ + Δ^μ g^νρ Δ · r - g^μρ Δ^ν Δ · r) +

  (Σ_{i=0}^{m-3} (<<1>>)^i (Δ · <<1>>)^(-i+m-3)) <<1>>^<<1>> (<<1>>) - <<1>> +

  (Σ_{j=0}^{m-4} (j + 1) (Δ · p)^(-j+m-4) (-(Δ · r) - Δ · s)^(j-i) (-(Δ · s))^i)

   Δ^ν Δ^ρ (-s^μ Δ^σ Δ · p + g^μσ Δ · s Δ · p - Δ^μ p^σ Δ · s + Δ^μ Δ^σ p · s) +

  (Σ_{j=0}^{m-4} (j + 1) (Δ · q)^(-j+m-4) (-(Δ · r))^i (-(Δ · r) - Δ · s)^(j-i)) Δ^μ Δ^σ

   (-r^ν Δ^ρ Δ · q + g^νρ Δ · r Δ · q - Δ^ν q^ρ Δ · r + Δ^ν Δ^ρ q · r) - (Σ_{j=0}^{m-4} (j + 1) (Δ · q)^(-j+m-4) (-(Δ · r) - Δ · s)^(j-i) (-(Δ

   Δ^μ Δ^ρ (-s^ν Δ^σ Δ · q + g^νσ Δ · s Δ · q - Δ^ν q^σ Δ · s + Δ^ν Δ^σ q · s)
```

Polarized case, non-zero momentum insertion

```
In[2141]:= SetOptions[Twist2GluonOperator, Polarization → 1, Explicit → All];
Out[2141]= Short[Twist2GluonOperator[{p, μ, a}, {q, ν, b}, {r, ρ, c}], 5]
```

With the setting ZeroMomentumInsertion → False a non-zero momentum is assumed to flow into the operator vertex.

This is the Feynman rule associated with the quark-antiquark operator, where p is the momentum of the incoming quark and q the momentum of the antiquark. The momentum flowing into the operator vertex is thus -p-q.

```
In[2142]:= g_s ( - ε^νρΔp Δ^μ (Δ · p)^(m-2) + <<20>> +

   1/(Δ · q + Δ · r)^2 (ε^μρΔp Δ^ν ((Δ · q + Δ · r)^m + (Σ_{i=0}^{m-3} (Δ · r)^(-i+m-3) (Δ · q + Δ · r)^i) Δ · r^3 +

      2 (Σ_{i=0}^{m-3} (Δ · r)^(-i+m-3) (Δ · q + Δ · r)^i) Δ · q Δ · r^2 + (Σ_{i=0}^{m-3} (Δ · r)^(-i+m-3) (Δ · q + Δ · r)^i) Δ · q^2 Δ · r

   ε^μνρΔ (Δ · q Δ · r (Δ · p)^m + Δ · q (Δ · r)^m Δ · p + (Δ · q)^m Δ · r Δ · p)) / (Δ · q Δ · r Δ · p)) f_abc
Out[2142]= SetOptions[Twist2QuarkOperator, Polarization → 1, ZeroMomentumInsertion → True]
```

This is the quark-antiquark-gluon operator vertex.

```
In[2143]:= {CouplingConstant → g_s, Dimension → D, Explicit → True, Polarization → 1, ZeroMomentumInsertion
True}
Out[2143]= t1 = Twist2QuarkOperator[p]
```

This shows the FeynCalcExternal form.

```
In[2144]:= -(γ · Δ).γ^5 (Δ · p)^(m-1)
Out[2144]= t2 = Twist2QuarkOperator[{p}, {q}, {k, μ, a}]
```

Unpolarized case, non-zero momentum insertion

```
In[2145]:= -g_s (γ · Δ).γ^5.T_a (Sum_{i=0}^{m-2} (-1)^i (Δ · p)^(-i+m-2) (Δ · q)^i) Δ^μ
Out[2145]= t3 = Twist2QuarkOperator[{p}, {q}, {k, μ, a}, {r, ν, b}]
```

Quark-antiquark operator.

```
In[2146]:= -g_s^2 (γ · Δ).γ^5.(T_b.T_a (Sum_{j=0}^{m-3} (j + 1) (-1)^j (Δ · p)^i (Δ · q)^(-j+m-3) (Δ · p + Δ · r)^(j-i)) -
                (-1)^m T_a.T_b (Sum_{j=0}^{m-3} (j + 1) (-1)^j (Δ · p)^(-j+m-3) (Δ · q)^i (Δ · q + Δ · r)^(j-i))) Δ^μ Δ^ν
Out[2146]= SetOptions[Twist2QuarkOperator, Polarization → 0, ZeroMomentumInsertion → True]
```

Quark-antiquark-gluon operator.

```
In[2147]:= {CouplingConstant → g_s, Dimension → D, Explicit → True, Polarization → 0, ZeroMomentumInsertion
True}
Out[2147]= t4 = Twist2QuarkOperator[p]

In[2148]:= γ · Δ (Δ · p)^(m-1)
```

Twist3QuarkOperator

Description

Twist3QuarkOperator[p] or Twist3QuarkOperator[p,_,_] yields the 2-quark operator (p is momentum in the direction of the fermion number flow). Twist3QuarkOperator[{p1,___}, {p2,___}, {p3, mu, a}] or Twist3QuarkOperator[p1,_,_, p2,_,_, p3,mu,a] Quark-Quark-Gluon-operator, where p1 is the incoming quark, p2 the incoming antiquark and p3 denotes the (incoming) gluon momentum. Twist3QuarkOperator[{p1,___}, {p2,___}, {p3, mu, a}, {p4, nu, b}] or Twist3QuarkOperator[p1,_,_, p2,_,_, p3,mu,a, p4, nu, b] gives the Quark-Quark-Gluon-Gluon-operator. The setting of the option Polarization (unpolarized: 0; polarized: 1) determines whether the uppolarized or polarized operator is returned.

```
In[2149]:= t5 = Twist2QuarkOperator[{p}, {q}, {k, μ, a}]
Out[2149]= g_s (γ · Δ).T_a (Sum_{i=0}^{m-2} (-1)^i (Δ · p)^(-i+m-2) (Δ · q)^i) Δ^μ
```

See also: Twist2QuarkOperator, Twist2GluonOperator.

Examples

```
In[2150]:= t6 = Twist2QuarkOperator[{p}, {q}, {k, μ, a}, {r, ν, b}]
```

$$\text{Out[2150]}= -(-1)^m g_s^2 (\gamma\cdot\Delta).\left(T_a.T_b\left(\sum_{i=0}^{m-3}(i+1)(-(\Delta\cdot p))^{-i+m-3}(\Delta\cdot q)^j(k\cdot\Delta+\Delta\cdot q)^{i-j}\right)+ T_b.T_a\left(\sum_{i=0}^{m-3}(i+1)(-(\Delta\cdot p))^{-i+m-3}(\Delta\cdot q)^j(\Delta\cdot q+\Delta\cdot r)^{i-j}\right)\right)\Delta^\mu\Delta^\nu$$

Twist4GluonOperator

Description

Twist4GluonOperator[{oa, ob, oc, od}, {p1,la1,a1}, {p2,la2,a2}, {p3,la3,a3}, {p4,la4,a4}].
See also: Twist2QuarkOperator, Twist3QuarkOperator, Twist2GluonOperator.

Examples

```
In[2151]:= Twist2QuarkOperator[p]//FCE//StandardForm
Out[2151]= GSD[OPEDelta] SOD[p]^(-1+OPEm)
```

TwoLoopSimplify ***unfinished*** (examples missing)

Description

TwoLoopSimplify[amplitude,{qu1,qu2}] simplifies the 2-loop amplitude (qu1 and qu2 denote the integration momenta). TwoLoopSimplify[amplitude] transforms to TwoLoopSimplify[amplitude, {Global‘q1, Global‘q2}], i.e., the integration momenta in amplitude must be named q1 and q2.
See also: OneLoopSimplify.

Examples

```
In[2152]:= Twist2QuarkOperator[{p}, {q}, {k, μ, a}, Explicit → All, Polarization → 0]
```

Uncontract

Description

Uncontract[exp, q1, q2, ...] uncontracts Eps and DiracGamma. Uncontract[exp, q1, q2, Pair → {p}] uncontracts also p.q1 and p.q2; the option Pair → All uncontracts all momenta except OPEDelta.

```
In[2153]:= -g_s (γ·Δ).T_a Δ^μ ((Δ·p)^(m-1)/(Δ·p+Δ·q) + (-1)^m (Δ·q)^(m-1)/(Δ·p+Δ·q))
```

```
Out[2153]= Twist2QuarkOperator[{p}, {q}, {k, μ, a}, {r, ν, b}, Explicit → All, Polarization → 0]
```

See also: Contract.

Examples

$$\text{In[2154]:=}\quad -g_s^2\,(\gamma\cdot\Delta)\,.\left(T_a.T_b\left(\frac{(-1)^{m-1}\,(\Delta\cdot p)^{m-1}}{(\Delta\cdot p+\Delta\cdot q)\,(k\cdot\Delta+\Delta\cdot p+\Delta\cdot q)}-\frac{(\Delta\cdot q)^{m-1}}{k\cdot\Delta\,(\Delta\cdot p+\Delta\cdot q)}+\frac{(k\cdot\Delta+\Delta\cdot q)^{m-1}}{k\cdot\Delta\,(k\cdot\Delta+\Delta\cdot p+\Delta\cdot q)}\right)+ T_b.T_a\left(\frac{(-1)^{m-1}\,(\Delta\cdot p)^{m-1}}{(\Delta\cdot p+\Delta\cdot q)\,(\Delta\cdot p+\Delta\cdot q+\Delta\cdot r)}-\frac{(\Delta\cdot q)^{m-1}}{(\Delta\cdot p+\Delta\cdot q)\,\Delta\cdot r}+\frac{(\Delta\cdot q+\Delta\cdot r)^{m-1}}{\Delta\cdot r\,(\Delta\cdot p+\Delta\cdot q+\Delta\cdot r)}\right)\right)\Delta^\mu\,\Delta^\nu$$

```
Out[2154]= SetOptions[Twist2QuarkOperator, Polarization → 1, ZeroMomentumInsertion → False]

In[2155]:= {CouplingConstant → g_s, Dimension → D, Explicit → True, Polarization → 1, ZeroMomentumInsertion
False}
Out[2155]= t1 = Twist2QuarkOperator[{p, q}]

In[2156]:= 2^(1-m) γ^5 . (γ · Δ) (Δ · p - Δ · q)^(m-1)
Out[2156]= t2 = Twist2QuarkOperator[{p}, {q}, {k, μ, a}]
```

$$\text{In[2157]:=}\quad 2^{2-m}\,g_s\,T_a.\gamma^5.(\gamma\cdot\Delta)\left(\sum_{i=0}^{m-2}(-(k\cdot\Delta)+\Delta\cdot p-\Delta\cdot q)^{-i+m-2}\,(k\cdot\Delta+\Delta\cdot p-\Delta\cdot q)^i\right)\Delta^\mu$$

```
Out[2157]= StandardForm[FCE[t2]]

In[2158]:= 2^(2-OPEm) Gstrong SUNT[a] . GA[5] . GSD[OPEDelta] FVD[OPEDelta, μ]
            OPESum[(-SOD[k] + SOD[p] - SOD[q])^(-2-OPEi+OPEm) (SOD[k] + SOD[p] - SOD[q])^OPEi,
              {OPEi, 0, -2 + OPEm}]
Out[2158]= SetOptions[Twist2QuarkOperator, Polarization → 1, ZeroMomentumInsertion → False]
```

By default scalar products are not uncontracted.

```
In[2159]:= {CouplingConstant → g_s, Dimension → D, Explicit → True, Polarization → 1, ZeroMomentumInsertion
False}
Out[2159]= t4 = Twist2QuarkOperator[{p, q}]
```

With the option Pair→All they are “uncontracted ”.

```
In[2160]:= 2^(1-m) γ^5 . (γ · Δ) (Δ · p - Δ · q)^(m-1)
Out[2160]= t5 = Twist2QuarkOperator[{p}, {q}, {k, μ, a}]
```

$$\text{In[2161]:=}\quad 2^{2-m}\,g_s\,T_a.\gamma^5.(\gamma\cdot\Delta)\left(\sum_{i=0}^{m-2}(-(k\cdot\Delta)+\Delta\cdot p-\Delta\cdot q)^{-i+m-2}\,(k\cdot\Delta+\Delta\cdot p-\Delta\cdot q)^i\right)\Delta^\mu$$

```
Out[2161]= Clear[t1, t2, t3, t4, t5, t6];

In[2162]:= Twist3QuarkOperator//Options
```

UnDeclareNonCommutative

Description

UnDeclareNonCommutative[a, b, ...] undeclares a,b, ... to be noncommutative, i.e., DataType[a,b, ..., NonCommutative] is set to False.

See also: DataType, DeclareNonCommutative.

Examples

```
In[2163]:= {CouplingConstant → g_s, Dimension → D, Polarization → 1}
```

As a side-effect of DeclareNonCommutative x is declared to be of DataType NonCommutative.

```
In[2164]:= Twist3QuarkOperator[p]
Out[2164]= (-1)^m (γ · Δ).γ^5 (Δ · p)^(m-1)
```

The inverse operation is UnDeclareNonCommutative.

```
In[2165]:= Twist4GluonOperator[{oa, ob, oc, od},
             {p1, la1, a1}, {p2, la2, a2}, {p3, la3, a3}, {p4, la4, a4}]
```

```
In[2166]:= (p2^la1 Δ^la2 Δ · p1 - g^la1la2 Δ · p2 Δ · p1 + Δ^la1 p1^la2 Δ · p2 - Δ^la1 Δ^la2 p1 · p2)
             (p4^la3 Δ^la4 Δ · p3 - g^la3la4 Δ · p4 Δ · p3 + Δ^la3 p3^la4 Δ · p4 - Δ^la3 Δ^la4 p3 · p4) δa1od δa2oc δa3ob δa4oa +
           (p2^la1 Δ^la2 Δ · p1 - g^la1la2 Δ · p2 Δ · p1 + Δ^la1 p1^la2 Δ · p2 - Δ^la1 Δ^la2 p1 · p2)
             (p4^la3 Δ^la4 Δ · p3 - g^la3la4 Δ · p4 Δ · p3 + Δ^la3 p3^la4 Δ · p4 - Δ^la3 Δ^la4 p3 · p4) δa1oc δa2od δa3ob δa4oa +
           (p3^la1 Δ^la3 Δ · p1 - g^la1la3 Δ · p3 Δ · p1 + Δ^la1 p1^la3 Δ · p3 - Δ^la1 Δ^la3 p1 · p3)
             (p4^la2 Δ^la4 Δ · p2 - g^la2la4 Δ · p4 Δ · p2 + Δ^la2 p2^la4 Δ · p4 - Δ^la2 Δ^la4 p2 · p4) δa1od δa2ob δa3oc δa4oa +
           (p3^la1 Δ^la3 Δ · p1 - g^la1la3 Δ · p3 Δ · p1 + Δ^la1 p1^la3 Δ · p3 - Δ^la1 Δ^la3 p1 · p3)
             (p4^la2 Δ^la4 Δ · p2 - g^la2la4 Δ · p4 Δ · p2 + Δ^la2 p2^la4 Δ · p4 - Δ^la2 Δ^la4 p2 · p4) δa1ob δa2od δa3oc δa4oa +
           (p4^la1 Δ^la4 Δ · p1 - g^la1la4 Δ · p4 Δ · p1 + Δ^la1 p1^la4 Δ · p4 - Δ^la1 Δ^la4 p1 · p4)
             (p3^la2 Δ^la3 Δ · p2 - g^la2la3 Δ · p3 Δ · p2 + Δ^la2 p2^la3 Δ · p3 - Δ^la2 Δ^la3 p2 · p3) δa1oc δa2ob δa3od δa4oa +
           (p4^la1 Δ^la4 Δ · p1 - g^la1la4 Δ · p4 Δ · p1 + Δ^la1 p1^la4 Δ · p4 - Δ^la1 Δ^la4 p1 · p4)
             (p3^la2 Δ^la3 Δ · p2 - g^la2la3 Δ · p3 Δ · p2 + Δ^la2 p2^la3 Δ · p3 - Δ^la2 Δ^la3 p2 · p3) δa1ob δa2oc δa3od δa4oa +
           (p2^la1 Δ^la2 Δ · p1 - g^la1la2 Δ · p2 Δ · p1 + Δ^la1 p1^la2 Δ · p2 - Δ^la1 Δ^la2 p1 · p2)
             (p4^la3 Δ^la4 Δ · p3 - g^la3la4 Δ · p4 Δ · p3 + Δ^la3 p3^la4 Δ · p4 - Δ^la3 Δ^la4 p3 · p4) δa1od δa2oc δa3oa δa4ob +
           (p2^la1 Δ^la2 Δ · p1 - g^la1la2 Δ · p2 Δ · p1 + Δ^la1 p1^la2 Δ · p2 - Δ^la1 Δ^la2 p1 · p2)
             (p4^la3 Δ^la4 Δ · p3 - g^la3la4 Δ · p4 Δ · p3 + Δ^la3 p3^la4 Δ · p4 - Δ^la3 Δ^la4 p3 · p4) δa1oc δa2od δa3oa δa4ob +
         (p4^la1 Δ^la4 Δ · p1 - g^la1la4 Δ · p4 Δ · p1 + Δ^la1 p1^la4 Δ · p4 - Δ^la1 Δ^la4 p1 · p4)
           (p3^la2 Δ^la3 Δ · p2 - g^la2la3 Δ · p3 Δ · p2 + Δ^la2 p2^la3 Δ · p3 - Δ^la2 Δ^la3 p2 · p3) δa1od δa2oa δa3oc δa4ob +
         (p4^la1 Δ^la4 Δ · p1 - g^la1la4 Δ · p4 Δ · p1 + Δ^la1 p1^la4 Δ · p4 - Δ^la1 Δ^la4 p1 · p4)
           (p3^la2 Δ^la3 Δ · p2 - g^la2la3 Δ · p3 Δ · p2 + Δ^la2 p2^la3 Δ · p3 - Δ^la2 Δ^la3 p2 · p3) δa1oa δa2od δa3oc δa4ob +
         (p3^la1 Δ^la3 Δ · p1 - g^la1la3 Δ · p3 Δ · p1 + Δ^la1 p1^la3 Δ · p3 - Δ^la1 Δ^la3 p1 · p3)
           (p4^la2 Δ^la4 Δ · p2 - g^la2la4 Δ · p4 Δ · p2 + Δ^la2 p2^la4 Δ · p4 - Δ^la2 Δ^la4 p2 · p4) δa1oc δa2oa δa3od δa4ob +
         (p3^la1 Δ^la3 Δ · p1 - g^la1la3 Δ · p3 Δ · p1 + Δ^la1 p1^la3 Δ · p3 - Δ^la1 Δ^la3 p1 · p3)
           (p4^la2 Δ^la4 Δ · p2 - g^la2la4 Δ · p4 Δ · p2 + Δ^la2 p2^la4 Δ · p4 - Δ^la2 Δ^la4 p2 · p4) δa1oa δa2oc δa3od δa4ob +
         (p3^la1 Δ^la3 Δ · p1 - g^la1la3 Δ · p3 Δ · p1 + Δ^la1 p1^la3 Δ · p3 - Δ^la1 Δ^la3 p1 · p3)
           (p4^la2 Δ^la4 Δ · p2 - g^la2la4 Δ · p4 Δ · p2 + Δ^la2 p2^la4 Δ · p4 - Δ^la2 Δ^la4 p2 · p4) δa1od δa2ob δa3oa δa4oc +
         (p3^la1 Δ^la3 Δ · p1 - g^la1la3 Δ · p3 Δ · p1 + Δ^la1 p1^la3 Δ · p3 - Δ^la1 Δ^la3 p1 · p3)
           (p4^la2 Δ^la4 Δ · p2 - g^la2la4 Δ · p4 Δ · p2 + Δ^la2 p2^la4 Δ · p4 - Δ^la2 Δ^la4 p2 · p4) δa1ob δa2od δa3oa δa4oc +
         (p4^la1 Δ^la4 Δ · p1 - g^la1la4 Δ · p4 Δ · p1 + Δ^la1 p1^la4 Δ · p4 - Δ^la1 Δ^la4 p1 · p4)
           (p3^la2 Δ^la3 Δ · p2 - g^la2la3 Δ · p3 Δ · p2 + Δ^la2 p2^la3 Δ · p3 - Δ^la2 Δ^la3 p2 · p3) δa1od δa2oa δa3ob δa4oc +
         (p4^la1 Δ^la4 Δ · p1 - g^la1la4 Δ · p4 Δ · p1 + Δ^la1 p1^la4 Δ · p4 - Δ^la1 Δ^la4 p1 · p4)
           (p3^la2 Δ^la3 Δ · p2 - g^la2la3 Δ · p3 Δ · p2 + Δ^la2 p2^la3 Δ · p3 - Δ^la2 Δ^la3 p2 · p3) δa1oa δa2od δa3ob δa4oc +
         (p2^la1 Δ^la2 Δ · p1 - g^la1la2 Δ · p2 Δ · p1 + Δ^la1 p1^la2 Δ · p2 - Δ^la1 Δ^la2 p1 · p2)
           (p4^la3 Δ^la4 Δ · p3 - g^la3la4 Δ · p4 Δ · p3 + Δ^la3 p3^la4 Δ · p4 - Δ^la3 Δ^la4 p3 · p4) δa1ob δa2oa δa3od δa4oc +
         (p2^la1 Δ^la2 Δ · p1 - g^la1la2 Δ · p2 Δ · p1 + Δ^la1 p1^la2 Δ · p2 - Δ^la1 Δ^la2 p1 · p2)
           (p4^la3 Δ^la4 Δ · p3 - g^la3la4 Δ · p4 Δ · p3 + Δ^la3 p3^la4 Δ · p4 - Δ^la3 Δ^la4 p3 · p4) δa1oa δa2ob δa3od δa4oc +
         (p4^la1 Δ^la4 Δ · p1 - g^la1la4 Δ · p4 Δ · p1 + Δ^la1 p1^la4 Δ · p4 - Δ^la1 Δ^la4 p1 · p4)
           (p3^la2 Δ^la3 Δ · p2 - g^la2la3 Δ · p3 Δ · p2 + Δ^la2 p2^la3 Δ · p3 - Δ^la2 Δ^la3 p2 · p3) δa1oc δa2ob δa3oa δa4od +
         (p4^la1 Δ^la4 Δ · p1 - g^la1la4 Δ · p4 Δ · p1 + Δ^la1 p1^la4 Δ · p4 - Δ^la1 Δ^la4 p1 · p4)
           (p3^la2 Δ^la3 Δ · p2 - g^la2la3 Δ · p3 Δ · p2 + Δ^la2 p2^la3 Δ · p3 - Δ^la2 Δ^la3 p2 · p3) δa1ob δa2oc δa3oa δa4od +
```

```
Out[2166]= {}

In[2167]:= Options[Uncontract]
Out[2167]= {Dimension → Automatic, DimensionalReduction → False, Pair → {}, Unique → True}

In[2168]:= t1 = LeviCivita[μ, ν][p, q]

In[2169]:= ϵ^μνpq
Out[2169]= Uncontract[t1, p]
```

Upper

Description

Upper may be used inside LorentzIndex to indicate an contravariant LorentzIndex.
See also: LorentzIndex, Lower, Contract1.

UVPart ***unfinished*** (examples missing)

Description

UVPart[exp, q] discards ultraviolet finite integrals (q = integration momentum).

Examples

```
In[2170]:= ϵ^μν$AL$8091(1)q p^$AL$8091(1)
```

Vectors

Description

Vectors is an option for FORM2FeynCalc. Its default setting is Automatic. It may be set to a list, if the FORM-file does not contain a V(ectors) statement.
See also: FORM2FeynCalc.

WriteOut

Description

WriteOut is an option for OneLoop. If set to True, the result of OneLoop will be written to a file called "name.res", where name is the first argument of OneLoop.
See also: OneLoop.

WriteOutPaVe

Description

WriteOutPaVe is an option for PaVeReduce and OneLoopSum. If set to a string, the results of all Passarino-Veltman PaVe's are stored in files with names generated from this string and the arguments of PaVe.

See also: PaVeReduce, PaVe, OneLoopSum.

Write2

Description

Write2[file, val1 = expr1, val2 = expr2, ...] writes the settings val1 = expr1, val2 = expr2 in sequence followed by a newline, to the specified output file. Setting the option FormatType of Write2 to FortranForm results in FORTRAN syntax output.

The continuation character for FORTRAN output can be controlled by changing $FortranContinuationCharacter.

```
In[2171]:= t2 = DiracSlash[p]
Out[2171]= γ · p

In[2172]:= Uncontract[t2, p]
Out[2172]= γ^$AL$8092(1) p^$AL$8092(1)

In[2173]:= Uncontract[t1, p, q]
Out[2173]= ϵ^μν$AL$8094(1)$AL$8093(1) q^$AL$8093(1) p^$AL$8094(1)
```

See also: Isolate, PaVeReduce.

Examples

```
In[2174]:= Uncontract[ScalarProduct[p, q], q]
Out[2174]= p · q

In[2175]:= Uncontract[ScalarProduct[p, q], q, Pair → All]
```

This writes the assignment r=t to a file.

```
In[2176]:= p^$AL$8096(1) q^$AL$8096(1)
```

This shows the contents of the file.

```
In[2177]:= Uncontract[ScalarProduct[p, q]^2, q, Pair → All]
Out[2177]= p^$AL$8097(1) q^$AL$8097(1) p^$AL$8097(2) q^$AL$8097(2)

In[2178]:= Clear[t1, t2]

In[2179]:= DeclareNonCommutative[x]
Out[2179]= DataType[x, NonCommutative]
```

```
In[2180]:= True

In[2181]:= UnDeclareNonCommutative[x]
Out[2181]= DataType[x, NonCommutative]

In[2182]:= True
```

This is how to write out the expression t2 in Fortran format.

```
In[2183]:= DeclareNonCommutative[y, z]

In[2184]:= DataType[y, z, NonCommutative]
Out[2184]= {True, True}

In[2185]:= UnDeclareNonCommutative[y, z]
```

XYT

Description

XYT[exp, x,y] transforms DataType[*y, z,* NonCommutative] away ...

ZeroMomentumInsertion

Description

ZeroMomentumInsertion is an option of FeynRule, Twist2GluonOperator and Twist2QuarkOperator.
See also: FeynRule, Twist2GluonOperator, Twist2QuarkOperator.

Zeta2

Description

Zeta2 denotes Zeta[2]. N[Zeta[2]] evaluates to N[Zeta[2]].
See also: SimplifyPolyLog.

Examples

```
In[2186]:= {True, True}
Out[2186]= {}
Out[2186]= FullForm[$FortranContinuationCharacter]

In[2187]:= "&"
Out[2187]= Options[Write2]
```

$Abbreviations

Description

$Abbreviations are a list of string substitution rules used when generating names for storing intermediate results. It is used by OneLoop and PaVeReduce.The elements of the list should be of the form "name" → "abbreviation".

```
In[2188]:= {D0Convention → 0, FinalSubstitutions → {}, FormatType → InputForm, FortranFormatDoublePrecis
             PageWidth → 62, PostFortranFile →, PreFortranFile →, StringReplace → {}}
Out[2188]= Attributes[Write2]
```

See also: Abbreviation, OneLoop, PaVeReduce, WriteOut, WriteOutPaVe.

$AL

Description

$AL is the head for dummy indices which may be introduced by Uncontract.

```
In[2189]:= {HoldRest}
Out[2189]= t = Collect[((a - c)^2 + (a - b)^2)^2, a, Factor]

In[2190]:= 4 a^4 - 8 (b + c) a^3 + 8 (b^2 + c b + c^2) a^2 - 4 (b + c) (b^2 + c^2) a + (b^2 + c^2)^2
Out[2190]= tempfilename = ToString[$SessionID] <> ".s";

In[2191]:= Write2[tempfilename, r = t];

In[2192]:= TableForm[
             ReadList[If[$OperatingSystem === "MacOS", " : ", ""] <> tempfilename, String]]
           r = ( 4 * a^4 - 8 * a^3 * (b + c) - 4 * a * (b + c) * (b^2 + c^2) +
Out[2192]=  (b^2 + c^2)^2 + 8 * a^2 * (b^2 + b * c + c^2)
             );

In[2193]:= DeleteFile[If[$OperatingSystem === "MacOS", " : ", ""] <> tempfilename]
```

$BreitMaison

Description

The Breitenlohner-Maison t2 = x + Isolate[t, a, IsolateNames → W] scheme is currently not supported by the Dirac algebra functions.Use Tracer if you need it (PHI supplies the functions FCToTracer and TracerToFC).

See also: Tr, $Larin, $West.

$Color

Description

$Color is False by default. If set to True, some special variables will be colored.

$Covariant

Description

The boolean setting of $Covariant determines whether lorentz indices are displayed as lower indices (True) or as upper ones (False).

```
In[2194]:= 4 a^4 - 8 W(1) a^3 + 8 W(3) a^2 - 4 W(1) W(2) a + W(2)^2 + x
Out[2194]= Write2[tempfilename, r = t2];

In[2195]:= TableForm[
             ReadList[If[$OperatingSystem === "MacOS", " : ", ""] <> tempfilename, String]]
           W[1] = (b + c
              );
           W[2] = (b^2 + c^2
              );
Out[2195]= W[3] = (b^2 + b * c + c^2
              );
           r = ( 4 * a^4 + x - 8 * a^3 * HoldForm[W[1]] - 4 * a * HoldForm[W[1]]*
            HoldForm[W[2]] + HoldForm[W[2]]^2 + 8 * a^2 * HoldForm[W[3]]
              );

In[2196]:= DeleteFile[If[$OperatingSystem === "MacOS", " : ", ""] <> tempfilename]

In[2197]:= Write2[tempfilename, r = t2, FormatType → FortranForm];
Out[2197]= TableForm[
             ReadList[If[$OperatingSystem === "MacOS", " : ", ""] <> tempfilename, String]]

In[2198]:=
             W(1D0) = W(1D0)
             W(2D0) = W(2D0)
             W(3D0) = W(3D0)
             r = 4D0 * a * *4 + x - 8D0 * a * *3 * W(1D0) -
           & 4D0 * a * W(1D0) * W(2D0) + W(2D0) * *2 + 8D0 * a * *2 * W(3D0)
```

$FCS

Description

$FCS is a list of functions with a short name. E.g. GA[nu] can be used instead of DiracGamma[nu].

```
In[2199]:= DeleteFile[If[$OperatingSystem === "MacOS", " : ", ""] <> tempfilename];
           Clear[t, t2, tempfilename];
Out[2199]= (x y)^m
```

$FCT

Description

Setting the global variable $FCT to True changes the default typesetting rules for the noncommutative multiplication operator Dot[].

```
In[2200]:= Zeta2

In[2201]:= ζ(2)
Out[2201]= N[Zeta2]

In[2202]:= 1.64493

In[2203]:= SimplifyPolyLog[π^2]
Out[2203]= 6 ζ(2)
```

$FeynCalcStuff

Description

$FeynCalcStuff is the list of availabe stuff in FeynCalc.

$FeynCalcVersion

Description

$FeynCalcVersion is a string that represents the version of FeynCalc.

```
In[2204]:= $Abbreviations//StandardForm
```

```
Out[2204]= {^ →, * →, {→, [→, } →, ] →, / →,

           →,

           →, →, , →, AxialVector → AV, Fermion → F, Momentum →,

             Pair →, ParticleMass → m, PseudoScalar → PS, RenormalizationState →,

             Scalar → S, SmallVariable → sma, Subscript → su, Vector → V}
```

$FortranContinuationCharacter

Description

$FortranContinuationCharacter is the continuation character used in Write2.

```
In[2205]:= $AL
Out[2205]= $AL
```

See also: Write2.

$Gauge

Description

$Gauge(= 1/xi) is a constant specifying the gauge fixing parameter of QED in Lorentz gauge. The usual choice is Feynman gauge, $Gauge=1. Notice that $Gauge is used by some functions, the option Gauge by others.

See also: Gauge.

$Larin

Description

If set to True, the Larin-Gorishny-Atkyampo-DelBurgo-scheme for Uncontract[ScalarProduct[*p, q*], *q,* Pair → All] in *D*-dimensions is used, i.e. before evaluating traces (but after moving $p^{\$AL\$8098(1)}$ $q^{\$AL\$8098(1)}$ anticommuting in all dimensions to the right of the Dirac string) a product gamma[mu].gamma5 is substituted to -I/6 Eps[mu,al,be,si] gamma[al,be,si], where all indices live in *D*-dimensions now. Especially the Levic-Civita tensor is taken to be *D*-dimensional, i.e., contraction of two Eps's results in *D*'s. This has (FOR ONE AXIAL-VECTOR-CURRENT ONLY, it is not so clear if this scheme also works for more than one fermion line involving $\$AL = \mu$;) the same effect as the Breitenlohner-Maison-'t Hooft-Veltman scheme.

```
In[2206]:= Uncontract[ScalarProduct[p, q], q, Pair → All]
Out[2206]= p^μ$8099(1) q^μ$8099(1)
```

See also: Tr, $BreitMaison, $West.

$LimitTo4

Description

$LimitTo4 is a global variable with default setting True. If set to False, no limit Dimension → 4 is performed after tensor integral decomposition.

```
In[2207]:= $AL = .;
Out[2207]= γ5
```

See also: PaVe, PaVeReduce, OneLoop.

$LorentzIndices

Description

$LorentzIndices is a global variable. If set to True the dimension of LorentzIndex is displayed as an index.

```
In[2208]:= $Covariant

In[2209]:= False
Out[2209]= GA[μ]

In[2210]:= γ^μ
Out[2210]= $Covariant = True;

In[2211]:= GA[μ]
Out[2211]= γμ

In[2212]:= $Covariant = False;
Out[2212]= $FCS

In[2213]:= {CDr, FAD, FC, FCE, FCI, FDr, FI, FV, FVD, GA, GA5, GGV, GP,

             GS, GSD, LC, LCD, MT, MTD, QGV, QO, SD, SOD, SP, SPC, SPD, SPL, PMV}
```

$MIntegrate

Description

$MIntegrate is a global list of integrations done by Mathematica inside OPEIntDelta.

```
In[2214]:= $FCT = True;
Out[2214]= GA[μ] . GA[ν]
```

See also: OPEIntDelta.

$MU

Description

$MU is the head for dummy indices which may be introduced by Chisholm (and evtl. Contract and DiracReduce).

```
In[2215]:= γ^μ γ^ν
Out[2215]= $FCT = False;
```

See also: Chisholm.

$MemoryAvailable

Description

$MemoryAvailable is a global variable which is set to an integer *n*, where *n* is the available amount of main memory in MB. The default is 128. It should be increased if possible. The higher $MemoryAvailable can be, the more intermediate steps do not have to be repeated by FeynCalc.

```
In[2216]:= GA[μ] . GA[ν]
Out[2216]= γ^μ.γ^ν
```

$NonComm

Description

$NonComm contains a list of all non-commutative heads present.

```
In[2217]:= $FeynCalcVersion
Out[2217]= 5.0.0beta1
```

See also: NonCommQ, NonCommutative.

$PairBrackets

Description

$PairBrackets determines whether brackets are drawn around scalar products in the notebook interface.

```
In[2218]:= $FortranContinuationCharacter
Out[2218]= &

In[2219]:= γ5
Out[2219]= γ5

In[2220]:= γ5
```

```
Out[2220]= $Larin

In[2221]:= False
Out[2221]= $LimitTo4

In[2222]:= True
Out[2222]= $LorentzIndices = True;
```

$SpinorMinimal

Description

$SpinorMinimal is a global switch for an additional simplification attempt in DiracSimplify for more than one Spinor-line. The default is False, since otherwise it costs too much time.

```
In[2223]:= MetricTensor[α, β, Dimension → n]DiracMatrix[α, Dimension → n]
Out[2223]= γ^α_n MetricTensor(α, β, Dimension → n)
```

See also: DiracSimplify.

$VeryVerbose

Description

$VeryVerbose is a global variable with default setting 0. If set to 1, 2, ..., less and more intermediate comments and informations are displayed during calculations.

```
In[2224]:= %//StandardForm
Out[2224]= DiracGamma[LorentzIndex[α, n], n] MetricTensor[α, β, Dimension → n]

In[2225]:= MTD[α, β] FVD[p, β]GAD[μ]//FCI

In[2226]:= γ^μ_D g^αβ p^β_D

In[2227]:= %//StandardForm
```

See also: $V0.

$West

Description

If $West is set to True (which is the default), traces involving more than 4 Dirac matrices and DiracGamma[LorentzIndex[μ, D], D] Pair[LorentzIndex[α, D], LorentzIndex[β, D]] Pair[LorentzIndex[β, D], Momentum[p, D]] are calculated recursively

according to formula (A.5) from Comp. Phys. Comm 77 (1993) 286-298, which is based on the Breitenlohner Maison $*LorentzIndices* = False;-scheme.
See examples in Tr.

Evaluation time and memory usage

```
In[2228]:= $MIntegrate
Out[2228]= {}

In[2229]:= $MU
Out[2229]= $MU

In[2230]:= $MemoryAvailable
Out[2230]= 256

In[2231]:= $NonComm
Out[2231]= {ChiralityProjector, DiracGamma, DiracGammaT, DiracMatrix, DiracSigma, DiracSlash, Spinor, GA,
             GS, LeftPartialD, LeftRightPartialD2, LeftRightPartialD, PartialD, QuantumField, RightParti
             SpinorUBar, SpinorU, SpinorVBar, SpinorV, SUNT, FieldStrength, QuarkGluonVertex, QuarkPropag
             C, Twist2QuarkOperator, OPESum, CovariantD, 𝒰, HighEnergyPhysics`Phi`Objects`Private`plus,
             χ_, χ₊, UFMinus, UFPlus, UGamma, UMatrix, USmall, Twist2AlienOperator, Twist2CounterOperator}
```

Functions (experimental)

DoPolarizationSums

Description

EXPERIMENTAL
DoPolarizationSums[exp] sums over vector polarizations for expressions with a factor of the form
Pair[LorentzIndex[rho1_], Momentum[Polarization[p_, -I]]]*
Pair[LorentzIndex[rho2_], Momentum[Polarization[p_, I]]].
See also: Polarization, Uncontract.

Examples

In the following (and indeed everywhere else within FeynCalc), notice that Lorentz indices written as super or subscripts are not to be taken as such. Instead, by convention, when two indices are contracted one is always lower and the other upper.
FeynCalc uses the following normalization of Polarization vectors:

```
In[2232]:= $PairBrackets
```

```
Out[2232]= False

In[2233]:= SP[p, q]SP[r, s]
Out[2233]= p · q r · s
```

DoPolarizationSums is chosen in such a way as to be consistent with this normalization and with $*PairBrackets* = TrueTrue=SP[*p, q*]SP[*r, s*]:

We can choose e.g. the following polarization vectors, labeled with a subscript, $\{(p \cdot q)\,(r \cdot s)$,$*PairBrackets* = False,False,$*SpinorMinimal*$\} = \{(i,0,0,0), (0,i,0,0), (0,0,i,0), (0,0,0,i)\}$.

```
In[2234]:= False

In[2235]:= $VeryVerbose

In[2236]:= 0
Out[2236]= $VeryVerbose = 3;

In[2237]:= $VeryVerbose = 0;

In[2238]:= γ5
```

341.105MemoryInUse[]=24424832:

```
In[2239]:= MaxMemoryUsed[]
Out[2239]= 25668048
```

$$\text{TimeUsed[]}182.8= \frac{\text{Pair[LorentzIndex[}\mu\text{], Momentum[Polarization[p4, }-i\text{]]]}}{\text{Pair[LorentzIndex[}\mu\text{], Momentum[Polarization[p4, }i\text{]]]}}:$$

```
In[2240]:= ε*μ(p4) εμ(p4)
Out[2240]= %//Contract

In[2241]:= -1
Out[2241]= ∑

In[2242]:= εμ("p"4) (εν) ("p"4)
Out[2242]= -gμ ν

In[2243]:= e1
Out[2243]= e2

In[2244]:= e3
Out[2244]= e4

In[2245]:= Clear[ee, tt]
Out[2245]= ev[i_] := ReplacePart[{0, 0, 0, 0}, I, i];

In[2246]:= {ev[1], ev[2], ev[3], ev[4]}
```

FeynmanDoIntegrals ***unfinished***

Description

EXPERIMENTAL
FeynmanDoIntegrals[exp, moms, vars] attempts to evaluate integrals over Feynman parameters vars in an expression exp as produced e.g. with FeynmanReduce. The variables given must be atomic quantities (AtomQ). If vars is omitted all variables in the integrals will be integrated. If vars is given, only the variables in vars will be integrated. moms is a list of all external momenta. The integrals in exp must be given in the form Integratedx[x, low, up].int, where x is the integration variable, low and up are the integration limits and int the integrand. The interval [low,up] is assumed to include integration bounds put by possible DeltaFunctions and moreover it is assumed that up ≥ 0 and that up > low. The two options FCIntegrate and FCNIntegrate determine which integration will be applied to integrals judged respectively analytically and numerically doable. This judging is a very rough one. Using (TimedIntegrate[##, Integrate→Integrate]&) or (TimedIntegrate[##, Integrate→NIntegrate]&) as the setting of one or both allows for finer judging. Those that are judged neither analytically nor numerically doable are left unevaluated, but can of course be attempted evaluated by a simple sustitution. Beside the explicit options of FeynmanDoIntegrals options of the integration functions specified (FCIntegrate and FCNIntegrate) may be given and are passed on to these.

See also: FeynmanReduce, FeynmanParametrize1.

```
                 i  0  0  0
In[2247]:= ( 0  i  0  0 )
           ( 0  0  i  0 )
                 0  0  0  i
Out[2247]= e_i^μ
```

Examples

```
In[2248]:= ee[i_][mu_] := ev[i][[μ]];
```

FeynmanParametrize1 ***unfinished***

Description

EXPERIMENTAL
FeynmanParametrize1[exp,k,Method->Denominator] introduces Feynman parameters for all one-loop integrals in exp (k = integration momentum) using formula (11.A.1) from "The Quantum Theory of Fields" vol. 1 by Steven Weinberg. FeynmanParametrize1[exp,k,Method->Exp] introduces Feynman parameters for all one-loop integrals in exp (k = integration momentum) using 1/(A-I eps) = I Integrate[Exp[-I x (A-I eps)],{x,0,Infinity}, Assumptions->{Arg[A]==0,Arg[eps]==0}]. In this case, when the option Integrate is set to True, odd factors of k-tensors are dropped and even factors are replaced according to Itzykson&Zuber (8-117).

```
In[2249]:= tt[i_, j_, mu_, nu_] := ee[i][μ]ee[j][ν]
Out[2249]= e_i^μ
```

Examples

```
In[2250]:= e_j^ν
```

FeynmanReduce ***unfinished***

Description

EXPERIMENTAL
FeynmanReduce[exp,params] takes a Feynman parameterized expression exp (as e.g. generated with FeynmanParametrize1) and a list of Feynman parameters as input and attempts to simplify the expression. If no parameters are given, Integratedx variables in the expression will be used. Currently, reduction of exponentials is implemented. This will work on terms of the form E^p1[a,b,c,...]*p2[a,b,c,...], where p1 and p2 are fractions of polynomials in the Feynman parameters a,b,c,... If the option Expand is set to True, FeynmanReduce will attempt to bring the expression exp into a sum of such terms and operate on the terms one by one.

```
In[2251]:= ∑
Out[2251]= e_i^μ e_{j,ν}
```

Examples

```
In[2252]:= -g_μ^ν
```

Other objects that are considered to be ok or under active development:

```
In[2253]:= Table[∑_{i=1}^{4} tt[i, i, μ, ν], {μ, 1, 4}, {ν, 1, 4}]

Out[2253]= ⎛-1   0   0   0 ⎞
           ⎜ 0  -1   0   0 ⎟
           ⎜ 0   0  -1   0 ⎟
           ⎝ 0   0   0  -1 ⎠
```

Objects that are under consideration for being discarded:

```
In[2254]:= ∑
Out[2254]= e_i^μ e_{j,ν}

In[2255]:= -g_i^j

Out[2255]= Table[∑_{μ=1}^{4} tt[i, j, μ, μ], {i, 1, 4}, {j, 1, 4}]

           ⎛-1   0   0   0 ⎞
In[2256]:= ⎜ 0  -1   0   0 ⎟
           ⎜ 0   0  -1   0 ⎟
           ⎝ 0   0   0  -1 ⎠
```

Bibliography

[1] J. A. Vermaseren, arXiv:math-ph/0010025.

[2] T. Hahn and M. Perez-Victoria, Comput. Phys. Commun. **118** (1999) 153 [arXiv:hep-ph/9807565].

[3] J. Küblbeck, M. Böhm and A. Denner, Comp. Phys. Comm. **60** (1990) 165.
"FeynArts web site".

[4] R. D. Drinkard and N. K. Sulinski, "Macsyma: A Program For Computer Algebraic Manipulation (Demonstrations And Analysis)," NUSC-6401 "SPIRES entry".

[5] E. Yehudai and A. C. K. Hsieh, *HIP* — Symbolic High-Energy Physics Calculations,
SLAC-PUB- **5576** July 1991.

[6] A. Pukhov *et al.*, arXiv:hep-ph/9908288.

[7] G. Altarelli and G. Parisi, Nucl. Phys. B **126** (1977) 298.

[8] R. Mertig and R. Scharf, "TARCER: A mathematica program for the reduction of two-loop propagator integrals," Comput. Phys. Commun. **111** (1998) 265 "arXiv:hep-ph/9801383".

[9] O. V. Tarasov, "Generalized recurrence relations for two-loop propagator integrals with arbitrary masses," Nucl. Phys. B **502** (1997) 455 "arXiv:hep-ph/9703319".

[10] J. Gasser and H. Leutwyler, "Chiral Perturbation Theory: Expansions In The Mass Of The Strange Quark," Nucl. Phys. B **250** (1985) 465.

[11] G. 't Hooft and M. J. Veltman, "Scalar One Loop Integrals," Nucl. Phys. B **153** (1979) 365.

[12] A. Denner, "Techniques for the Calculation of Electroweak Radiative Corrections at the One-Loop Level and Results for W-Physics at LEP200", to appear in *Fortschritte der Physik 1992, 41* and references therein.

[13] R. Mertig, M. Böhm, and A. Denner, Comp. Phys. Comm. **64** (1991) 345.

[14] Comp. Phys. Comm **77** (1993) 286-298.

[15] G. J. van Oldenborgh, "FF – a package to evaluate one-loop Feynman diagrams", Comp. Phys. Comm. **66** (1991) 1.
Z. Phys. C **46** (1990) 425,
"Scanned version from KEK".

[16] P. Breitenlohner and D. Maison, “Dimensional Renormalization And The Action Principle,” Commun. Math. Phys. **52** (1977) 11.

[17] C. P. Martin and D. Sanchez-Ruiz, “Action principles, restoration of BRS symmetry and the renormalization group equation for chiral non-Abelian gauge theories in dimensional renormalization with a non-anticommuting gamma(5),” Nucl. Phys. B **572** (2000) 387 [arXiv:hep-th/9905076].

[18] P.Cvitanovic, Phys. Rev. D **14** (1976) 1536.

[19] A.P. Kryukov and A.Ya. Rodionov, Comp. Phys. Comm. **48** (1988) 327.

[20] Matthias Jamin and E. Lautenbacher, "TRACER, A Mathematica Package for γ-Algebra in Arbitrary Dimensions", preprint Technische Universität München, TUM-T31-20/91.

[21] S. Wolfram, "MACSYMA Tools for Feynman Diagram Calculations", *Proceedings of the 1979 MACSYMA Users Conference.*

[22] Frederik Orellana, "PHI, a Mathematica package for doing loop calculations in Chiral Perturbation Theory", *in preparation*, "PHI web site".

[23] G. Ecker, CERN-TH-6660-92 *To be published in the proceedings of 4th Hellenic School on Elementary Particle Physics, Corfu, Greece, 2-20 Sep 1992 and Lectures given at Cargese Summer School on Quantitative Particle Physics, Cargese, 20 Jul-1 Aug 1992*.

[24] T. Hahn, Comput. Phys. Commun. **140** (2001) 418 [arXiv:hep-ph/0012260]. "FeynArts web site".

[25] T. Muta, World Sci. Lect. Notes Phys. **57** (1998) 1.