TUM–T31–20/91
September 1991

TRACER

Version 1.1

A MATHEMATICA Package for γ-Algebra in Arbitrary Dimensions

MATTHIAS JAMIN and MARKUS E. LAUTENBACHER

Technische Universität München
Physik Department T30
8046 Garching, Germany

ABSTRACT

This paper describes the first MATHEMATICA implementation of γ-algebra in arbitrary space-time dimensions according to the 't Hooft-Veltman scheme. It is the only system based on a general purpose symbolic manipulation program that treats the γ_5-problem mathematically consistent in arbitrary dimensions. The TRACER package is capable of doing just purely algebraic manipulations as well as trace operations on strings of γ-algebra objects. In addition, it provides a set of utility functions for reordering, simplifying and improving the readability of the output. Optionally the output can be obtained in a form suitable to be fed into a TEX system for high quality text processing. As a whole, the TRACER package is intended as a computerized aid to a researcher working on higher order corrections in Relativistic Quantum Field Theories.

A short comparison of procedural versus rule-based programming approaches is given and the discussion is supplemented by a toy implementation of the γ-algebra in rule-based style. The paper describes in detail the usage of the TRACER package and for further illustration a correlation function of two weak currents is calculated with the aid of TRACER. Finally, data on the performance of TRACER on some common platforms are given.

COPYRIGHT AND DISCLAIMER OF WARRANTY

Contents

1 Introduction

"State of the art" radiative correction calculations in Relativistic Quantum Field Theories, notably the Standard Model (SM), nowadays involve at least one loop while for some processes 2-loop and even higher loop calculations are available. However, practically speaking as the number of Feynman diagrams to be evaluated increases tremendously with the number of loops, computerized aid by symbolic manipulation programs (SMP's) becomes almost indispensable. Simultaneously, from the technical viewpoint one has to get rid of intermediate singularities inherent to quantum corrections preferably by the method of dimensional regularization. Hence the calculations have to be performed in an arbitrary number of space-time dimensions. However, it is known that in space-time dimensions not equal to the physical dimension 4 the usual treatment of the γ-matrix γ_5 can lead to inconsistencies as e.g. using a naively anticommuting γ_5 is known not to reproduce the triangle anomaly. All in all the problem calls for a SMP capable of consistently doing γ-algebra in arbitrary space-time dimensions.

At the time the present work was started existing programs like REDUCE [1], SYMBOL-HIT [2] or FORM [3] were implemented consistently for dimensions equal to 4 only, or like SCHOONSCHIP [4] were written in assembly code and hence not directly extendible by the user. Therefore we decided to write our own fully consistent implementation of γ-algebra on top of a general purpose SMP. This has the advantage that built-in commands of the general purpose SMP like series expansion, factoring etc. can be used to further manipulate the results obtained from a trace calculation. As the working environment we favored MATHEMATICA [5] over other present generation SMP's like MAPLE [6] or MACSYMA [7] because it features rule-based programming, a coding technique especially suited for implementing mathematical knowledge.

In the present paper we describe the first MATHEMATICA implementation of γ-algebra in arbitrary space-time dimensions according to the 't Hooft-Veltman (HV) scheme [8, 9, 10]. To us it is the only SMP of its kind that treats the γ_5-problem mathematically consistent in arbitrary dimensions.[1)] The TRACER package allows for γ-algebraic manipulations as well as trace calculations. In addition several routines for tailoring the output are provided.

The paper is organized as follows. We start with a short review of procedural versus rule-based programming. Next we demonstrate how a γ-algebra system

[1)] In the final phase of this work we learned about two somehow related packages also written in MATHEMATICA. The HIP package from SLAC [11] is intended for tree level calculations only and furthermore limited to dimensions equal to 4. However, at the tree level the current version of HIP provides a rich set of operations like e.g. phase space integration for multi-particle final states. The code of the FEYNARTS - FEYNCALC package under development in Würzburg has to our knowledge not yet been distributed. According to the publications [12], [13] the package uses the naive anticommuting γ_5 in d dimensions.

could be programmed in principal. To this end we set up a toy γ-algebra system in rule-based style that is still easy to understand but already clearly demonstrates the key ideas underlying the full implementation. Following this general part all the routines provided in the TRACER package are described in detail and benchmarks for the performance of TRACER on different types of computers are given. The appendices cover issues like a collection of γ-algebra formulae relevant for the implementation of our package, a sample application program that uses TRACER, changes made to built-in MATHEMATICA functions and finally a package index that lists all user available functions together with their on-line documentation.

2 Procedural versus Rule-Based Programming

Since the key programming style of the TRACER package is rule-based and this style is not that often used in physics application programming, we think it is time well spent to give a short review on programming methodologies relevant to the implementation of the TRACER package.

The TRACER package was written in MATHEMATICA, version 1.2 [5], but should run as well under version 2.0 without any difficulty. MATHEMATICA is a relatively new programming system for both numerical and symbolic computation. It supports a variety of programming styles such as procedural, rule-based, functional or recursive but is especially well suited for rule-based programming. All these approaches to programming are more or less equivalent in their final result of implementing a certain algorithm on a computer. However, coding can be easier, more elegant and efficient in a particular style depending on the problem at hand.

In our short review of programming methodologies we will solely concentrate on procedural and rule-based styles. The first one is at present the most widely used in computer programming and the second is the one in which the overwhelming part of the TRACER package is written.

2.1 Procedural Programming

Procedural programming is the standard approach used in traditional computer languages like C, FORTRAN or PASCAL. The basic idea is to specify explicitly the complete sequence of steps that implement a particular algorithm. At any time the state of the program is given by the values of all variables that are currently in use. Logical flow control is done explicitly on the one hand by conditionals, e.g. `If` statements, that decide which branches of the code to execute and on the other hand by iteration, e.g. `For` or `While` statements, which execute a single piece of code several times. One of the advantages of procedural programming lies in the fact that because it is *the* traditional style of computer programming most of the existing hardware has been designed to meet its special needs best. In addition the vast amount of ready-to-use pseudo codes for all kinds of problems found in the literature is given in a procedural like style.

On the other hand, disadvantages are e.g. that every single step of the algorithm has to be specified explicitly and that flow control has to be done in an explicit manner which can lead to quite lengthy and complicated code. (See the example in section 2.2.) Furthermore, the need for many control structures and variables inherent to the procedural style may lead to programs that are not easy to extend. Finally, as we will see in the later examples the procedural implementation sometimes turns out to be quite different from the mathematically compact formulated solution of the problem at hand.

As a straightforward example for procedural programming we give an implementation of the factorial `fac[n]` from the MATHEMATICA book [5]:

```
fac[n_] := Block[ {i, out=1},
                  For[i=2, i<=n, i++, out = i out];
                  out ];
```

2.2 Rule-Based Programming

While the procedural style is inherent to most of the traditional computer languages the new MATHEMATICA system relies heavily on rule-based programming previously encountered e.g. in languages like LISP. This style gives the user the ability to perform calculations by setting up collections of specific transformation rules. These rules then mimic tables of exactly those mathematical relations that must be applied when proceeding from the initial problem to its solution. The idea underlying such a calculation is to transform any expression matching a particular pattern according to the transformation rule belonging to that pattern. In this approach writing programs actually means setting up special purpose transformation rules. This is summarized graphically in fig. 1 that shows the generic flow diagram according to which a program evolves in rule-based systems like MATHEMATICA:

The total input consists of the user supplied expression to be evaluated and the transformation rules known to MATHEMATICA at the time the expression is entered. It is important to notice that the transformation rules relevant for the calculation actually are the combination of those already built into MATHEMATICA and those supplied by the user in his code. Next, both the expression and the rules are fed into a loop that transforms the expression according to the full set of known rules as long as any pattern on the left hand side of the rules matches the current form of the expression. If there are no more matching patterns the calculation is regarded as finished with the result being the last transformed form of the initial expression.

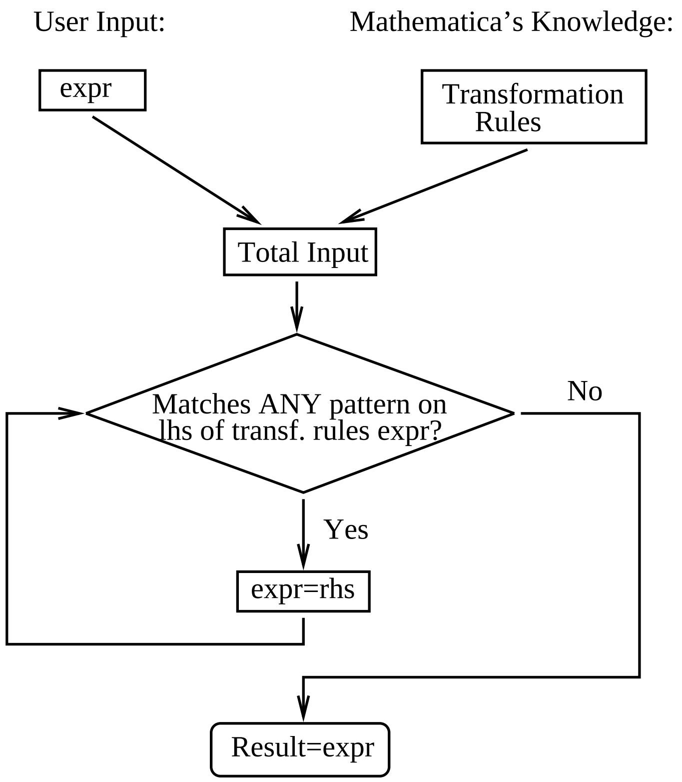

Figure 1: Flow chart for rule-based calculations in MATHEMATICA.

From the examples we are going to discuss later on it will become quite obvious that there is a striking similarity between transformation rules and pure mathematical relations. Therefore programs written in rule-based style naturally bear a close relation to the mathematical solution of the problem at hand. Another advantage of this approach is that because transformation rules are directed there is almost no need for explicit flow control via conditionals and iterations. This fact in turn leads to clearer and shorter code. Such a feature turns out to be of great importance for solutions to problems of ever growing abstraction and complexity such as the present symbolic implementation of γ-algebra in arbitrary dimensions. Finally, the rule-based style naturally allows for building up programs in stages since dealing with an increased range of valid input just amounts to adding appropriate new transformation rules to the already existing ones.

Of course rule-based programming also bears some drawbacks in it. A first disadvantage is that conventional hardware generally is not optimized for this coding technique to the same extent it is for procedural programming. In addition, since using rules rather than conditionals and iterations makes logical flow control more compact but at the same time less explicit, special care has to be taken in the design of large programs to avoid notorious errors like missing end conditions resulting in infinite transformation loops.

Before we draw our conclusions on the two programming styles just described, we supplement the preceding discussion of the various features inherent to both styles by another example from the MATHEMATICA book [5]: The implementation of the symbolic derivate for functions of generic type $f(x) = x^n$ and $f(x) = \log(x)$. This can be done as follows in procedural and rule-based style, respectively:

```
(* PROCEDURAL STYLE *)

diff[y_, x_] :=
   Block[ {n},
          If[ TrueQ[Length[y]==2 && y[[0]]==Power &&
                    y[[1]]==x], n=y[[2]]; n x^(n-1), If[ TrueQ[y==x],
              1,
                  If[TrueQ[Length[y]==1 &&
                         y[[0]]==Log && y[[1]]==x], 1/x ] ] ] ];


(* RULE-BASED STYLE *)

diff[x_^n_., x_] := n x^(n-1);
diff[Log[x_], x_] := 1/x;
```

These few lines of code clearly illustrate most of the points mentioned in the preceding discussion. It is quite striking how compact and close to the relevant mathematical relations for the derivative the rule-based implementation is. Using specific patterns like `x_^n_.` or `Log[x_]` to distinguish the different types of functions `diff[]` is able to derive, everything that is needed are two immediately understandable lines of code. In contrast, the procedurally coded version consists of only a single routine and hence has to take two non-specific arguments. The derivative routine therefore consists of a quite complicated structure of conditionals determining internally the class of the function passed as an argument. The resulting code is not immediately obvious and as a whole bears little similarity with the simple mathematical equations to be implemented.

Drawing a final conclusion, it is justified to say that rule-based programming is superior to the traditional procedural style for the implementation of complicated and abstract mathematical relations. Programming in this style then basically means to rewrite the desired mathematical relations in the syntax of the rule-based SMP. The number of rules to implement may be large but in order to get a reasonable performance only a few of them should be involved in any particular step of the total calculation. However, for classical applications like “number crunching” procedural programming is still believed to be the better choice because of the wide availability and long experience with highly optimized hardware and compilers for it.

3 A Toy Implementation of γ-Algebra

This section is intended for the reader who is not only interested in using the TRACER package for High Energy Physics calculations but also how the package actually has been implemented using the rule-based approach. To this end the section describes a toy implementation of the trace operation in γ-algebra. At this point we want to stress explicitly that this particular implementation is far from being complete and mathematically consistent, hence the word *toy*. The toy system is solely intended as a small piece of rule-based code that is easy to understand and uses methods closely related to those used in the actual TRACER package. Therefore, the toy system can act as a pedagogical tool to see how TRACER works in principle but it certainly is not anything beyond that.

In what follows we give several mathematical relations for the trace operation in the γ-algebra[2)] together with the corresponding rule-based MATHEMATICA formulation. We will denote the user defined trace operation for γ-matrices by the new MATHEMATICA function `GammaTrace[]`.

The toy package implements the following γ-algebra relations for the trace operation:

- The treatment of γ_5:

$$\{\gamma_\mu, \gamma_5\} = 0 \ . \tag{1}$$

RULE I

```
GammaTrace[a___,G5,b_,c___] := - GammaTrace[a,b,G5,c] /; b =!= G5;
```

Rule I prescribes that any γ-algebra object (the named pattern `b_` matches any single MATHEMATICA expression while `a___` and `c___` denote patterns matching zero, one or more expressions) anticommutes with γ_5 (like in

2)In the toy implementation we constrain ourselves to an anticommuting γ_5 in d dimensions. The actual TRACER package of course uses the γ_5 as defined in [9] which is regarded to be mathematically consistent for arbitrary dimensions.

TRACER denoted by G5) except γ_5 itself. As a consequence of this rule γ_5 is always commuted to the very right in a string of γ-algebra objects.

- The contraction of two identical Lorentz indices in a string of γ-algebra objects determined by the two relations

$$\gamma_\mu \gamma^\mu \;=\; d \cdot \mathbf{1} \qquad \text{and} \qquad \{\gamma_\mu, \gamma_\nu\} \;=\; 2\, g_{\mu\nu} \cdot \mathbf{1} \,, \tag{2}$$

is implemented as

RULE IIa, b

```
GammaTrace[a___,{m_},{m_},b___] := d GammaTrace[a,b];
GammaTrace[a___,{m_},b___,c_,{m_},e___] :=
   - GammaTrace[a,{m},b,{m},c,e] + 2 g[c,{m}] GammaTrace[a,{m},b,e];
```

Rule IIa implements the contraction of an index pair in d dimensions, whereas rule IIb commutes two identical Lorentz indices close to each other. A Lorentz index is denoted by a MATHEMATICA symbol in curly brackets, a pure MATHEMATICA symbol like p is used to represent $\not{p}$.

- The trace of the unit matrix is equal to 4

$$Tr(\mathbf{1}) = 4. \tag{3}$$

According to the above rule IIa the expression GammaTrace[{mu},{mu}] transforms to d GammaTrace[]. Hence the trace of the unit matrix will be denoted by GammaTrace[] and one has to add the rule

RULE III

```
GammaTrace[] = 4;
```

- The trace of γ_5 vanishes,

$$Tr(\gamma_5) \;=\; 0 \,, \tag{4}$$

which leads to

Rule IV

```
GammaTrace[G5] = 0;
```

To load them later on repeatedly into Mathematica all the above transformation rules should be saved in an ASCII file, say `toygalg.m`, using a standard text editor.

Below we enclose for further illustration the transcript of an actual Mathematica session using this particular toy implementation of γ-algebra:

```
In[1]:= <<toygalg.m

In[2]:= ?GammaTrace
GammaTrace
GammaTrace/: GammaTrace[a___,G5,b_,c___] := -GammaTrace[a,b,G5,c] /; b =!= G5
GammaTrace/: GammaTrace[a___,{m_},{m_},b___] := d GammaTrace[a,b]
GammaTrace/: GammaTrace[a___,{m_},b___,c_,{m_},e___] :=
>    -GammaTrace[a,{m},b,{m},c,e] + 2 g[c,{m}] GammaTrace[a,{m},b,e]
GammaTrace/: GammaTrace[] = 4
GammaTrace/: GammaTrace[G5] = 0

In[2]:= GammaTrace[p,G5,q,r,s]
Out[2]= -GammaTrace[p,q,r,s,G5]

In[3]:= GammaTrace[p,{mu},q,r,{mu}]
Out[3]= d GammaTrace[p,q,r] - 2 GammaTrace[p,{mu},r] g[q,{mu}] +
>    2 GammaTrace[p,{mu},q] g[r,{mu}]

In[4]:= GammaTrace[{mu},{mu}]
Out[4]= 4 d

In[5]:= GammaTrace[G5,{mu},{mu}]
Out[5]= 0
```

First, the package is loaded via the `Get[]` command which is aliased to `<<` by default. Second, one asks the SMP using `?GammaTrace` what it has been taught about γ-algebra objects by the user supplied rules in `toygalg.m`. Next, each of the implemented algebraic relations is tried once. Concerning the last point there are a few things worthwhile to be mentioned explicitly:

- Rules are applied repeatedly on expressions resulting from the previous transformation until patterns governing the application of those rules no longer match the expression under consideration. This is e.g. true for the commutation of γ_5 to the right (rule I) in example `In[2]` and for the contraction of indices with an object in between (rule IIb) in `In[3]`. Here the corresponding rules match three and two times, respectively.

- The complete transformation of a single initial expression can involve several rules in a successive manner. This is illustrated by `In[3]` to `In[5]` which first involve the contraction of indices (rules IIa,b) and then the traces of the unit matrix (rule III) and of γ_5 (rule IV), respectively.

To get a so to speak "dynamical" understanding of how the final result evolves through the repeated application of transformation rules, one can use the debugging aid `On[]` provided by the MATHEMATICA system. Running the above sample session with debugging mode switched on one can watch on-line the working of rule-based calculations along the lines of the general flow chart in fig. 1.

4 The TRACER Package for γ-Algebra

4.1 The Input Syntax for TRACER

The first step in the presentation of our package is devoted to its input syntax. The output syntax will be discussed in section 4.4. In order to facilitate the use of this package (at least for former REDUCE users) we partly followed the syntax of the REDUCE High Energy package which we found to be rather convenient. On the other hand, we also tried to keep the MATHEMATICA convention for the naming of functions, namely first upper case letters for each new word in a function name.

This caused us to supply two equivalent names for the trace operator: `GammaTrace[]` as one would expect in MATHEMATICA and `G[]` as in REDUCE. The complete syntax for this operator is `GammaTrace[linei, exp1, exp2, ..., expn]` or `G[linei, exp1, exp2, ..., expn]` where `linei` is a symbol specifying a certain γ-matrix line. The question of whether to calculate the trace or just perform γ-algebra manipulations on a certain γ-matrix line is controlled by setting an appropriate flag for the corresponding line index `linei`. Further details will be addressed in the next section. The `expi` can be one of the following expressions:

- a momentum or a sum of momenta `p`, `p-q` or `(p+k1-k2)` which corresponds to $\not{p}$, $\not{p}-\not{q}$ and $(\not{p}+\not{k}_1-\not{k}_2)$

- an open index of a gamma matrix which is denoted by a symbol in curly brackets {`alpha`} corresponding to γ_α

- a linear combination of the unit matrix in the gamma algebra `U` and the matrix γ_5, denoted by `G5`, with scalar coefficients

- a linear combination of a momentum `p` and a mass term `m U`

We should remark that TRACER keeps sums of momenta as they are and does not expand them in a sum of `G[]`'s, e.g. `G[l, p-q]` is *not* immediately converted into `G[l, p] - G[l, q]`. This expansion will however be performed in the scalar product once a trace has been calculated. Our conventions for the γ-algebra and in particular for γ_5 are given in appendix A.

The reason for using a different syntax for momenta and indices, unlike in REDUCE, is that in our package indices which appear in pairs will automatically be contracted. Needless to say that an index should not appear more than twice because this will certainly lead to trouble ! Therefore a regular input for a γ-matrix line may look like

```
G[ l1, p-m U, {mu}, U-a G5, {alpha}, p-q+M U, {mu}, U+b G5]
```

corresponding to

$$(\not{p} - m)\,\gamma_\mu(1 - a\,\gamma_5)\,\gamma_\alpha(\not{p} - \not{q} + M)\,\gamma^\mu(1 + b\,\gamma_5)\;. \tag{5}$$

We have included a special function `Sigma[]` for the sigma matrix $\sigma_{\mu\nu}$ defined as $\sigma_{\mu\nu} = \frac{i}{2}(\gamma_\mu\gamma_\nu - \gamma_\nu\gamma_\mu)$ which can be used in γ-matrix expressions together with the `G[]` operator. Internally all σ-matrices are immediately converted to γ-matrices according to their definition. The syntax for `Sigma[]` is equivalent to the syntax for `G[]`. For example `Sigma[l1, {mu},{nu}]` stands for $\sigma_{\mu\nu}$ and `Sigma[l1, q-p,{beta}]` corresponds to $(q-p)^\alpha\sigma_{\alpha\beta}$.

Different `G[]` and `Sigma[]` operators either on the same or on different γ-matrix lines can be concatenated using the MATHEMATICA built-in `NonCommutativeMultiply[]` which is aliased to `**` in infix notation. TRACER internally puts all `G[]`'s on the same line into a single `G[]` keeping the order of the γ-matrices while it leaves `G[]`'s on different lines as they are. After the input expression has been transformed into an expression with only one `G[]` operator for each γ-matrix line there exists a possibility to keep the `NonCommutativeMultiply[]`'s between the different γ-matrix lines or to have them replaced by the `Times[]` operator. This choice is toggled via the switch `RemoveNCM[on]` or `RemoveNCM[off]`. The default setting for this switch is `on`, i.e. `NonCommutativeMultiply[]` will be removed. Keeping `NonCommutativeMultiply[]` is only necessary if one wishes to multiply additional `G[]` operators after a certain γ-matrix expression has been calculated. For example for the default setting

```
G[l1, p1] ** G[l2, q] ** G[l1, {mu}, p2] ** G[l2, p2, {mu}]
```

would be transformed into

```
G[l1, p1, {mu}, p2] G[l2, q, p2, {mu}]
```

The scalar product in Minkowski-space is denoted by the function `S[]` similar to the dot product in REDUCE. In fact our package also allows usage of the MATHEMATICA `Dot[]` operator, which is aliased to "`.`", instead of the `S[]` function. `S[]` is symmetric and bilinear in its two arguments. Valid input might look like `S[p,q-p]` or `p.(q-p)` representing $p \cdot (q-p)$. Momenta with uncontracted Lorentz-indices and the metric tensor $g_{\mu\nu}$ are as in REDUCE expressed via the `S[]` function as well. For example `S[q,{alpha}]` stands for q_α and `S[{mu},{nu}]` corresponds to the metric tensor $g_{\mu\nu}$. The equivalent input for the two last expressions using the `Dot[]` operator would be `q.{alpha}` and `{mu}.{nu}`.

In the TRACER package the ε-tensor is denoted by the operator `Eps[]`. Its input syntax is `Eps[exp1, exp2, exp3, exp4]` where the `exp1-4` can be either a momentum, a sum of momenta or a Lorentz index. A valid input could be `Eps[ p, q-s, {mu}, {nu}]` corresponding to $p^\alpha (q-s)^\beta \varepsilon_{\alpha\beta\mu\nu}$. The ε-tensor is essentially a four dimensional object. The delicacies of treating the ε-tensor in $d \neq 4$ dimensions will be discussed in appendix A.

In table 1 we give a short summary of the correspondence between objects defined in the TRACER package and the physical notation for the γ-algebra. The operators `H` and `T` are relevant only in the HV scheme and will be explained in section 4.3.

4.2 User Available Functions

In this section we will present in alphabetical order some functions which are partly indispensable for a proper usage of the TRACER package or are included for the convenience of the user. Functions which are related to γ_5 and in particular to the HV scheme will be presented in the next section.

`AntiCommute[flag]` toggles depending on `flag` (`on/off`) the usage of an anticommuting matrix γ_5. The default after the startup of the TRACER package is a non-anticommuting γ_5. `AntiCommute[]` returns the currently used type of γ_5. The technicalities of a non-anticommuting γ_5 will be discussed in the next section and in appendix A.

`ContractEpsGamma[expr]` performs the contraction of index pairs stemming from an ε-tensor and γ-matrices in `expr`. The relevant formula is

$$\varepsilon_{\mu\nu\lambda\omega}\gamma^\omega \;=\; i\,\gamma_5 \left(\tilde{g}_{\mu\nu}\tilde{\gamma}_\lambda - \tilde{g}_{\mu\lambda}\tilde{\gamma}_\nu + \tilde{g}_{\nu\lambda}\tilde{\gamma}_\mu - \tilde{\gamma}_\mu\tilde{\gamma}_\nu\tilde{\gamma}_\lambda \right) , \qquad (6)$$

TRACER Object	Example	Physical Equivalent
`GammaTrace` ≡ `G`	`GammaTrace[l, {mu}]` ≡ `G[l,{mu}]`	γ_μ
	`G[l,G5]`	γ_5
	`G[l,p]`	$\not p$
	`G[l,U]`	1
`H`	`G[l,H@{mu}]`	$\hat{\gamma}_\mu$
	`G[l,H@p]`	$\hat{\not p}$
`T`	`G[l,T@{mu}]`	$\tilde{\gamma}_\mu$
	`G[l,T@p]`	$\tilde{\not p}$
`Sigma`	`Sigma[l, {mu},{nu}]`	$\sigma_{\mu\nu}$
`Eps`	`Eps[p,{mu},q,{nu}]`	$\varepsilon_{\alpha\mu\beta\nu}\, p^\alpha q^\beta$
`S` ≡ `.`	`S[p,q]` ≡ `p.q`	$p \cdot q$
	`S[p,{mu}]` ≡ `p.{mu}`	p_μ
	`S[{mu},{nu}]` ≡ `{mu}.{nu}`	$g_{\mu\nu}$
	`S[H@p,H@q]` ≡ `H@p.H@q`	$\hat{p} \cdot \hat{q}$
	`S[H@p,H@{mu}]` ≡ `H@p.H@{mu}`	$\hat{p}_\mu$
	`S[H@{mu},H@{nu}]` ≡ `H@{mu}.H@{nu}`	$\hat{g}_{\mu\nu}$
	`S[T@p,T@q]` ≡ `T@p.T@q`	$\tilde{p} \cdot \tilde{q}$
	`S[T@p,T@{mu}]` ≡ `T@p.T@{mu}`	$\tilde{p}_\mu$
	`S[T@{mu},T@{nu}]` ≡ `T@{mu}.T@{nu}`	$\tilde{g}_{\mu\nu}$
`Spur`	`Spur[l]; G[l,{mu},{nu}]`	$Tr[\gamma_\mu \gamma_\nu]$
`NonCommutativeMultiply` ≡ `**`	`G[l1,{mu},U-G5]**G[l2,{mu},U+G5]`	$\gamma_\mu (1-\gamma_5) \otimes \gamma^\mu (1+\gamma_5)$

Table 1: Correspondence between the objects defined in the TRACER *package and the physical notation for the γ-algebra. The choice of the line index associated with a γ-algebra object is of relevance only when different fermion lines are combined with the tensor product $\otimes$.*

where γ-algebra objects with the tilde symbol $\tilde{}$ attached are meant to be 4-dimensional. The notion of the tilde symbol will be introduced in full detail in the next section and also in appendix A.

`ListCommands[]` is a function to be called without argument. It returns a list of all user available functions defined in the TRACER package.

`NoSpur[l1, ..., ln]` removes the γ-matrix lines with line indices `l1, ..., ln` from the set of lines to be traced. If `NoSpur[]` is called without argument it returns a list of all the line indices which currently will be traced. Concerning a subtlety related to the trace calculation see the entry on `Spur[]`.

`OnShell[flag, list1, ..., listn]` imposes or removes the on-shell declarations specified in the `listi`, depending on `flag` (`on/off`). The `listi` can have the following form: `{p, m^2}`, `{p, s, 0}` or `{e, -1}` corresponding to the replacements $p^2 = m^2$, $p \cdot s = 0$ and $e^2 = -1$. Invoking `OnShell[]` without any arguments returns a list of the current on-shell declarations.

`SortLine[ expr, l, reflist]` sorts the product of γ-matrices on the line `l` in expression `expr` with reference to the ordering given in the list `reflist`. Note that all momenta and all γ-matrices appearing on line `l` also have to appear in the list `reflist`. However, the argument `reflist` is optional and if omitted the `SortLine[]` function returns the expression `expr` sorted into canonical order according to the MATHEMATICA built-in `Sort[]`. If `l` does not match any γ-matrix line in `expr` `SortLine[]` returns the expression unchanged. As an example we give the following application of the `SortLine[]` function:

```
In[1]:= SortLine[ G[l1,p,{m},{n},q] G[l2,p,q], l1, {{m},p,{n},q} ]

Out[1]= (2 p_{m} G[l1,{n},q] - G[l1,{m},p,{n},q]) G[l2,p,q]
```

`Spur[l1, ..., ln]` adds the γ-matrix lines with line indices `l1, ..., ln` to the set of lines to be traced. If `Spur[]` is called without argument it returns a list of all those line indices which currently will be traced. On all other γ-matrix lines only purely algebraic manipulations like e.g. automatic index contraction are performed. However, one has to be careful if several `G[]` operators on the same γ-line should be joined using `NonCommutativeMultiply`. Because MATHEMATICA starts to evaluate nested expressions from the lowest level upwards the first step in the evaluation of products of type `G[l,...]**G[l,...]` is the calculation of the various traces specified by `Spur[]`. Only after this step TRACER tries to concatenate whatever is left over. A simple example for this type of input error is:

```
In[1]:= Spur[l]; G[l, p]**G[l, q]

Out[1]= 0
```

What happens is that we calculate the trace of `G[l,p]` and `G[l,q]` separately. This obviously leads to a total result of 0 which however is wrong. To obtain the correct answer one has to leave all lines initially untraced and only after γ-string fusion substitute in the relevant lines by a line index which will be traced. Corrected in this way the above example would look like:

```
In[1]:= Spur[l]; G[l1, p]**G[l1, q] /. l1 -> l

Out[1]= 4 p.q
```

Now only the trace is calculated of the concatenated γ-matrix string `G[l,p,q]` giving the correct result $4p \cdot q$.

`ToDiracBasis[expr, llist]` decomposes the expression `expr` containing strings of γ-matrices into the basis spanned by 1, γ_5, γ_μ, $\gamma_\mu\gamma_5$ and $\sigma_{\mu\nu}$. The argument `llist` is optional and if given only the γ-strings on the lines listed in `llist={ l1, ..., ln}` are decomposed. If `llist` is omitted the decomposition is done for all γ-matrix strings occurring in `expr`. The above basis for the γ-matrices is however only valid in 4-dimensional space-time. Therefore the package checks whether the space-time dimension has been set to 4 with the command `VectorDimension[4]`. For any other dimension it exits with a warning message. The extension to d-dimensional space-time will be included in a future version of the TRACER package. Since usually the TRACER package immediately converts all σ-matrices into γ-matrices, this conversion has to be prevented through an internal application of the `Hold[]` command in the output of `ToDiracBasis[]`. For further processing of output stemming from `ToDiracBasis[]` a `Release[]` has to be applied. The following sequence from a MATHEMATICA session is intended as an example for the application of the `ToDiracBasis[]` command:

```
In[1]:= VectorDimension[4]

In[2]:= ToDiracBasis[ G[l, {mu}, {nu}] ]
Out[2]= -I Sigma[ l,{mu},{nu} ] + g_{mu nu} U[ l ]

In[3]:= % // Release // Expand
                                 G[ l, {mu}, {nu} ]   G[ l, {nu}, {mu} ]
Out[3]= g_{mu nu} U[ l ] + ------------------ - ------------------
                                         2                    2
```

`ToUG5[ expr, llist]` decomposes the expression `expr` containing strings of γ-matrices according to the formula

$$a\,\Gamma_1\Gamma_2 + b\,\Gamma_1\gamma_5\Gamma_2 \;=\; \frac{a+b}{2}\,\Gamma_1\Big(1+\gamma_5\Big)\Gamma_2 + \frac{a-b}{2}\,\Gamma_1\Big(1-\gamma_5\Big)\Gamma_2\;, \qquad (7)$$

where Γ_1 and Γ_2 are strings of γ-matrices on a specific line. Depending on `llist` the decomposition is performed on the lines listed in `llist={ l1, ..., ln}` or if omitted on all occurring lines. Note that both terms $\Gamma_1\Gamma_2$ and $\Gamma_1\gamma_5\Gamma_2$ on the left hand side of eq. (7) have to appear in `expr` in order make the decomposition of the γ-string $\Gamma_1\Gamma_2$ unique and `ToUG5[]` to take action. In all other cases the expression `expr` will be returned unchanged. We have added this function in the TRACER package for its apparent use in the calculation of Feynman diagrams containing left- or right-handed vertices. Using `ToUG5[]` the final result of a calculation can be transformed back into the left-right structure after the calculation.

`VectorDimension[expr]` sets the space-time dimension to the expression `expr`. The default symbol for the dimension is `d`. For instance `VectorDimension[4-2 eps]` can be used to set a suitable dimension allowing for a direct ε-expansion in dimensional regularization. Note that the command `VectorDimension[4]` internally declares γ_5 to be anticommuting. It is recommended not to use the symbol `D` for the dimension since this can mix up with the differential operator! (Try for instance: `VectorDimension[D]; ({mu}.{mu})[ Sin[x], x]`). Invoking `VectorDimension[]` without argument shows the currently used expression for the space-time dimension. As should have become clear, `AntiCommute[]` and `VectorDimension[]` can be used independently thereby leading e.g. to the possibility of setting `AntiCommute[on]` while using `VectorDimension[d]`. Since this can lead to mathematically inconsistent settings for the γ-algebra the user is advised to use these two switches with special care.

`Version[]` takes no argument and returns some information on the current version of the TRACER package.

4.3 Functions related to the 't Hooft-Veltman scheme

In order to properly implement the 't Hooft-Veltman scheme into the TRACER package it is necessary to supply some additional operators and functions. For the mathematical background of these definitions we refer the reader to appendix A and the references cited there.

The first function relevant in this context is the function `AntiCommute[on/off]` which toggles the use of an anticommuting γ_5. This function has already been introduced in the last section. To work with a non-anticommuting γ_5, as is the case for the HV scheme, one has to invoke `AntiCommute[off]`. This is the default on startup of the TRACER package.

The general philosophy behind the HV scheme is to introduce besides d-dimensional γ-algebra objects such as momenta, metric-tensors and γ-matrices their $(d-4)$-dimensional counterparts. With these two sets of objects a consistent γ-algebra can be constructed [8, 9]. This implies that in the HV scheme not only rules for the d-dimensional objects have to be supplied but for the $(d-4)$-dimensional and eventually 4-dimensional objects as well. This confronts us with the problem of how to distinguish the γ-algebra objects defined in d, $(d-4)$ and 4 dimensions, respectively.

In the TRACER package all objects which have been defined in the notation discussed above, e.g. `G[l,p]`, `G[l,{mu}]` or `{mu}.{nu}` corresponding to $\not{p}$, γ_μ and $g_{\mu\nu}$, are the d-dimensional ones assuming `VectorDimension[d]` has been set. Their $(d-4)$-dimensional and 4-dimensional counterparts are obtained by applying two projection-operators, namely `H[]` and `T[]`. `H[]` stands for *Hat* and refers to the notation of ref. [9] while `T[]` stands for *Tilde*, the notation used in ref. [14]. For example, the $(d-4)$-dimensional objects $\hat{\not{p}}$, $\hat{\gamma}_\mu$ and $\hat{g}_{\mu\nu}$ are obtained through the commands `G[l,H[p]]`, `G[l,H[{mu}]]` and `H[{mu}.{nu}]` respectively. One can also use the MATHEMATICA prefix notation for the operator `H[]`, e.g. `G[l,H@p]`, `G[l,H@{mu}]` and `H@{mu}.{nu}`. The corresponding 4-dimensional objects $\tilde{\not{p}}$, $\tilde{\gamma}_\mu$ and $\tilde{g}_{\mu\nu}$ are obtained in an analogous manner with only the `H[]` operator replaced by `T[]`.

We have to define rules for three different types of objects which are however related by $(a = \tilde{a} + \hat{a})$. Here a denotes one of the mentioned structures momenta, γ-matrices or metric-tensors. It is therefore possible to replace one of these objects in terms of the other two. This gives a more compact and thus easier to debug code and in addition improves the overall performance of the package because of fewer rules to test during the calculation. We have decided to replace all 4-dimensional *tilde*-objects in terms of the d-dimensional and $(d-4)$-dimensional ones. This is done internally, immediately after an input line has been submitted.

In order to be able to recover 4-dimensional objects at the end of a calculation we have included the function `ToHatTilde[expr]`. Applying `ToHatTilde[]` to an arbitrary γ-matrix expression decomposes this expression into 4- and $(d-4)$-dimensional parts, respectively. For example `ToHatTilde[G[l,p]]` leads to `G[l,p^] + G[l,(p)~]`. An internal `Hold[]` is applied to the *tilde*-parts to prohibit a direct decomposition back into the d- and $(d-4)$-dimensional parts. In the output parenthesis around symbols like in `G[l, (p)~]` should remind the user of the invisible `Hold[]`. To further process the output after the `ToHatTilde[]` command was used a `Release[]` on the expression in question has to be performed.

In physical applications *external* momenta usually can be taken to have only 4-dimensional components non-vanishing. It is thus useful to have a function at hand which removes the *hat*-parts of momenta at the end of a calculation after all momentum integrations have been performed. Note that $(d-4)$-dimensional momenta over which a loop integration is performed have to be kept fully because together with $1/(d-4)$ divergences they can lead to finite contributions. The function in question is called `RemoveHatMomenta[expr,p1,...,pn]` where `expr` is the expression to be simplified and `p1, ..., pn` are the momenta which's $(d-4)$-dimensional part is to be removed. However, in order to keep the output expression as simple as possible these momenta are printed without *tilde*. Hence one has to keep in mind which momenta are really 4-dimensional but in physical applications this should cause no difficulties.

4.4 The Output Syntax for TRACER

MATHEMATICA has built-in the command `Format[]` which allows to specify the output format of expressions as well as how to convert them to TEX code. The second feature can be invoked with the `TeXForm[expr]` command. In order to improve the readability of the output we have defined two special output formats for the scalar product the *hat*- and the *tilde*-parts[3)].

The two formats can be toggled with the command `OutputFormat[flag]` where `flag` can be either `texlike` or `subscript`. The default is `texlike`. Examples for the output form of the `S[]` function together with the corresponding input syntax are given in the following table.

Input:			`OutputFormat[flag]:`	
(explicit)	(shorthand)		`texlike`	`subscript`
`S[p, q]`	`p.q`	$\longrightarrow$	`p.q`	`p.q`
`S[p,{mu}]`	`p.{mu}`	$\longrightarrow$	`p_{mu}`	`p`
				`  mu`
`S[{mu},{nu}]`	`{mu}.{nu}`	$\longrightarrow$	`g_{mu nu}`	`g`
				`  mu nu`

Trying to mimic the physical notation for $(d-4)$- and 4-dimensional objects introduced in the previous section the relevant momenta or Lorentz indices simply have symbols `^` and `~` appended in the output. However, in some cases the projection property of the `H[]` and `T[]` operators is used to obtain a unique form of an expression. For instance the input `H@p.q` is returned as `p^.q^` and *not* as `p^.q` as one might expect. We have not supplied a special output format for the ε-tensor because the argument of `Eps[]` can be a mixture of indices and momenta, the later denoting momenta with which the ε-tensor is contracted. Since it might be confusing if e.g. momenta were printed like indices the output format of `Eps[]` is kept identical to the input format.

As already mentioned earlier, the TEX format of an expression is obtained with the command `TeXFormat[expr]` where `expr` can contain any regular TRACER function. In general the `TeXFormat[]` command works in exactly the same way as in MATHEMATICA. The only exception concerns the product of strings of γ-matrices on different lines. If `RemoveNCM[off]` is set, i.e. `NonCommutativeMultiply[]` will be kept to separate different lines, `TeXFormat[]` automatically converts `NonCommutativeMultiply[]`'s into `\otimes`. To get a line separator `\otimes` even in cases where the `NonCommutativeMultiply[]`'s have been removed by setting `RemoveNCM[on]`, the additional function `ToOtimes[]` has to be applied to the relevant expression first. This is illustrated in the following example:

3) In fact, there are additional output formats for internal reasons which can be found in the TRACER source code.

```
In[1]:= {mu}.{nu} G[l1, H@p, {alpha}] G[l2,q] // ToOtimes // TeXForm

Out[1]//TeXForm= g_{\mu \nu} \not\!\hat p \gamma_{\alpha} \otimes \not\! q
```

Because of the already mentioned question how to print the ϵ-tensor for a mixture of indices and momenta in the argument we have only provided a limited TEX form for the ϵ-tensor

```
In[1]:= Eps[ p, q, {mu}, {nu}] // TeXForm

Out[1]//TeXForm= \epsilon\left(p,q,\{ \mu \} ,\{ \nu \} \right)
```

4.5 Performance Of The TRACER Package

In this subsection we want to give some details on the performance of the TRACER package. To this end we ran two "benchmark" MATHEMATICA programs on a few widely available computers. The benchmark programs are thought to give a good description of TRACER's performance in everyday usage since e.g. the second one was part of an actual higher order QCD radiative correction calculation [15]. The CPU time consumed in each run was measured with the MATHEMATICA built-in command `Timing[]`.

As a first benchmark and one of many verification steps we tried our package on a trace calculation published in ref. [16]. The trace to evaluate is

$$\begin{aligned}
&Tr\,[\hat\gamma_{\alpha_1}\hat\gamma_{\alpha_2}\dots\hat\gamma_{\alpha_{10}}\,\hat\gamma^{\alpha_1}\hat\gamma^{\alpha_2}\dots\hat\gamma^{\alpha_{10}}] = \\
&\quad -31023169536 + 38971179008d - 21328977920d^2 + 6679521280d^3 - \\
&\quad 1320732160d^4 + 171464832d^5 - 14710080d^6 + 816960d^7 - 27840d^8 + \\
&\quad 520d^9 - 4d^{10}\ , \qquad (8)
\end{aligned}$$

using a γ-algebra with γ_5 non-anticommuting and arbitrary space-time dimension. To allow for direct comparison with the performance of other γ-algebra packages we actually calculated $Tr[\gamma_{\alpha_1}\gamma_{\alpha_2}\dots\gamma_{\alpha_{10}}\,\gamma^{\alpha_1}\gamma^{\alpha_2}\dots\gamma^{\alpha_{10}}]$ and replaced d by $d-4$ in the final result. This is possible because the trace in eq. (8) contains only γ-matrices of type $\hat\gamma$. As an independent cross check we repeated the trace calculation using the TRACER operator `H[]` to enter the $\hat\gamma_{\alpha_i}$-matrices directly. Both runs confirmed the result given by Veltman [16]. However, because contraction rules involving $\hat\gamma$-matrices are more complicated and generate more intermediate terms the direct evaluation using the `H[]` operator took notably longer than when using the trick. The first half of table 2 summarizes the time for the fast run required on various machines. If available for a particular computer type we also give the time requirement for FORM and REDUCE to perform the trace calculation making use of the described trick. A look at the CPU times shows that TRACER proofs to be at least even with FORM. However, in contrast to the TRACER package FORM can not be used interactively but only in batch mode and furthermore requires explicit declarations for every symbol, index, etc.. REDUCE falls off quite drastically against both TRACER and FORM. Unfortunately, we could compare it only on the PC against the other γ-algebra programs.

In higher order radiative correction calculations the quantity of interest, the value of a Feynman diagram $\mathcal{D}$, is typically obtained by contracting a Lorentz tensor $\mathcal{T}$ with a corresponding Feynman integral $\mathcal{I}$

$$\mathcal{D}^{\mu_1\mu_2\mu_3\cdots} = \mathcal{T}^{\mu_1\mu_2\mu_3\cdots\nu_1\nu_2\nu_3\cdots}\mathcal{I}_{\nu_1\nu_2\nu_3\cdots}\,, \tag{9}$$

where some of the tensor indices μ_i might be equal and therefore correspond to index contractions within the tensor.

Therefore we have chosen as our second benchmark a problem actually being part of a larger 2-loop calculation published in ref. [15]. Explicitly, we calculated terms of order $\mathcal{O}(1/\varepsilon^2)$ and $\mathcal{O}(1/\varepsilon)$ resulting from the contraction of the tensor structure with the regularized Feynman integral [4)] appearing in the Feynman diagram of fig. 2. These terms contribute to the anomalous dimension of the corresponding $\Delta S = 1$ electroweak decay process. The full list of contributing 2-loop diagrams and further details on the calculation can be found in refs. [15, 17].

Suppressing the QCD color structure one readily derives the tensor structure of the diagram in fig. 2 to be

$$\mathcal{T}^{\nu_1\nu_2\ldots\nu_6} = \gamma^\mu\,(1-\gamma_5)\,\gamma^{\nu_1}\gamma^{\nu_2}\gamma^{\nu_3}\gamma_\mu\,(1-\gamma_5)\,\gamma^{\nu_4}\gamma^{\nu_5}\otimes\gamma^{\nu_6} \tag{10}$$

where $\otimes$ delimits γ-matrices on different fermion lines. The tensor has eight Lorentz indices but one pair of indices is already contracted internally so actually only six free indices remain.

Neglecting trivial overall factors the corresponding 2-loop Feynman integral is given by

$$\begin{aligned}\mathcal{I}_{\nu_1\nu_2\ldots\nu_6} &= \int\!\frac{d^dk}{(2\pi)^d}\int\!\frac{d^dl}{(2\pi)^d}\;\frac{(k+p)_{\nu_1}(k+l+p)_{\nu_3}(l+p)_{\nu_4}}{l^2(k+p)^2(l+p)^2(l-q)^2(k+l+p)^2}\times\\ &\quad\Big\{(2l-q)_{\nu_6}\;g_{\nu_2\nu_5}-(l+q)_{\nu_5}\;g_{\nu_2\nu_6}-(l-2q)_{\nu_2}\;g_{\nu_5\nu_6}\Big\}\,. \end{aligned}\tag{11}$$

Again there are six free indices which have to be contracted with the corresponding indices of the tensor from eq. (10). Because of the large number of

4) The regularization of the divergent integrals was done in $d = 4 - 2\,\varepsilon$ dimensions using dimensional regularization. In this method of regularization the variable ε parameterizes the appearing ultraviolet divergences.

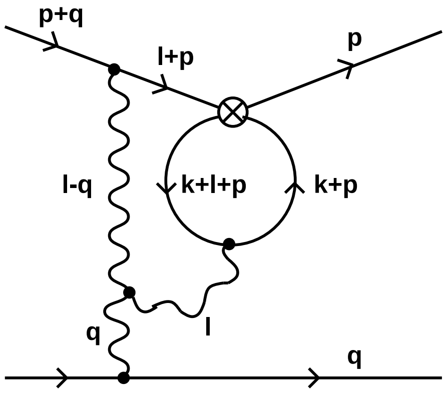

Figure 2: The 2-loop diagram calculated in one of the benchmarks. p and q denote external momenta and the wavy lines represent gluons. The four-quark vertex $\otimes$ stands for a quark operator insertion $\bar{Q}\gamma^\mu(1-\gamma_5)Q\ \bar{Q}\gamma_\mu(1-\gamma_5)Q$ where Q denotes the corresponding quark fields.

permutations one can get from six free indices and the complicated Lorentz structure stemming from the triple gluon vertex the explicit evaluation of the integral (11) is quite complicated and in the end yields 180 different terms. The individual terms consist of all possible products of the metric tensor $g_{\nu_i\nu_j}$ with external momenta p and q which have the same Lorentz structure as eq. (11). For obvious reasons the full result for the integral is not shown here.

In the light of such a vast number of terms encountered in practical calculations it becomes quite obvious that today's "state of the art" higher order radiative correction calculations indispensably call for assistance of the physicist by SMP's. The lack of such a tool in a modern symbolic computation environment like MATHEMATICA was one of the main reasons that lead to the development of the TRACER package.

After the MATHEMATICA session has been started and the TRACER package loaded the benchmark proceeded in the following steps:

1. Loading the tensor $\mathcal{T}$ and integral $\mathcal{I}$ from a MATHEMATICA source file using the `Get[]` command.

2. Contracting all pairs of identical Lorentz indices for all terms in the product $\mathcal{T}\cdot\mathcal{I}$ using an anticommuting γ_5 in $d=4-2\varepsilon$ dimensions.

3. Collecting terms of order $\mathcal{O}(1/\varepsilon^2)$ and $\mathcal{O}(1/\varepsilon)$.

4. Sorting matrices in the resulting γ-strings into a canonical order with the TRACER command `SortLine[]`.

5. If possible, combining different terms to form $(1\pm\gamma_5)$ structures using the TRACER command `ToUG5[]`.

6. Writing the result into two files as
   • plain ASCII text in MATHEMATICA's `OutputForm`. This file is intended to allow for an immediate look at the obtained results.
   • ASCII text consisting of TEX commands using MATHEMATICA's built-in and TRACER's newly defined `TeXForm` formats. This form of the output is suitable for further processing by a TEX system for improved readability.

It is worth mentioning that for a large number of terms in the final result the time spent on points 4. and 5. can become quite significant. This is of course of relevance from a practical point of view while from the pure algebraic point of view the calculation is already finished after the index contraction done in step 2.

The information gathered by the `Timing[]` command on different machines is collected in the second half of table 2.

Platform RAM [MB]	**PC 386/7** 4	**SS SLC** 16	**SS 1+** 24	**DS 3100** 16	**NS** 16	**SS 2** 16
Trace Calculation [sec]:						
TRACER	253	186	159	146	112	97
FORM	74	28	24	27	20	13
REDUCE	5584	1351	1140	–	–	657
Index Contraction [sec]	6825	1974	1730	1824	1261	975

Table 2: CPU time usage by TRACER *for trace calculation and index contraction "benchmarks" as described in the text. In the table SS, DS and NS stand for "Sun SPARCstation", "DEC station" and "NeXTstation", respectively. For the trace calculation we also give results for* FORM *and* REDUCE *if available.*

Comparing PC and workstation data one clearly sees that performance in large symbolic computation problems heavily depends on the amount of physical memory available to the program. For small problems which can be kept completely in the computer's RAM standard workstation performance is better only by factors 2 – 2.5. However, once for larger symbolic calculations virtual memory is involved the PC performance falls off drastically. Here obviously hard disk access when swapping intermediate results to disk becomes the bottle neck.

4.6 Getting TRACER

The TRACER package is available for distribution. It can be obtained by sending mail to the following address:

Matthias Jamin and Markus E. Lautenbacher
Technische Universität München
Physik Department T30
8046 Garching, Germany

The distribution can be done via electronic mail or via surface mail on floppy disks in MS-DOS format. Please enclose a properly *preformatted* $5\frac{1}{4}$" floppy disk for distribution on diskette.

5 Summary

We have given a short general discussion on the advantages of rule-based programming for the implementation of abstract algebraic relations. The discussion was supplemented by a toy implementation of a γ-algebra system showing the basic lines along such system can be realized in rule-based programming.

The TRACER package written in the general purpose SMP MATHEMATICA takes advantage of the rule-based programming approach to implement a fully consistent γ-algebra system in arbitrary space-time dimensions. It is intended as a powerful tool for the researchers who is working in the field of higher order corrections in Relativistic Quantum Field Theories. In this branch of physics the necessary amount of γ-algebraic manipulations becomes almost impractical to be done solely by hand and hence calls for computerized aid. Besides the implementation of the pure γ-algebra TRACER provides the user with a rich set of special purpose operators to tailor the output along his needs. The package can be used interactively as well as in batch mode, the latter mode being most suitable for calculations involving massive symbolic manipulations.

The TRACER system has been verified extensively against calculations done explicitly by hand. Furthermore, the package already provided valuable aid in several published or forthcoming works [15], [17] – [19] on radiative corrections to electroweak processes. The experience gathered during these calculations directly inspired improvements and extensions to the TRACER package. Ongoing research work on higher order quantum corrections will definitely continue to influence its future development.

Acknowledgment

We would like to thank W. Wetzel who generously provided us with a copy of his own MATHEMATICA implementation of γ-algebra for a 4-dimensional space-time. Having this as a reference point was quite useful during the early stages of our own implementation.

We are also grateful to A.J. Buras and P.H. Weisz for providing results on extensive trace calculations done by hand which we could use to test and debug the TRACER package as well as to G. Buchalla, S. Herrlich and J. Kalinowski for a plethora of suggestions on improving our package.

References

[1] A. C. HEARN, REDUCE *Version 3.3 – User's Manual*, The RAND Corporation Publication, 1987.

[2] P. KRAWCZYK and A. TURSKI, SYMBOL-HIT – *User's Manual*, *Warshaw University preprint* (1987).

[3] J. A. M. VERMASEREN, FORM *Version 1.0 – User's Manual and Tutorial*, CERN/NIKHEF-H Amsterdam Publication, 1989.

[4] H. STRUBBE, Review on Schoonship by M. Veltman, *Comp. Phys. Comm.* **8** (1974) 1.

[5] S. WOLFRAM, MATHEMATICA – *A System for Doing Mathematics by Computer*, Addison - Wesley Publishing Company Inc., 1988.

[6] B. W. CHAR *et al.*, MAPLE V – *Language Reference Manual*, Waterloo Maple Publishing, 1991.

[7] COMPUTER AIDED MATHEMATICS GROUP, MACSYMA – *Language Reference Manual*, Symbolics Inc. Publication, Burlington, MA, 1988.

[8] G. 'T HOOFT and M. VELTMAN, *Nucl. Phys.* **B44** (1972) 189.

[9] P. BREITENLOHNER and D. MAISON, *Comm. Math. Phys.* **52** (1977) 11.

[10] G. RUFA, *Annalen der Physik* **47** (1990) 6.

[11] A. HSIEH and E. YEHUDAI, HIP– *Symbolic High-Energy Physics Calculations*, *SLAC preprint,* **SLAC-PUB-5576 (T/E)**.

[12] J. KÜBLBECK, M. BÖHM, and A. DENNER, Feyn Arts, *Comp. Phys. Comm.* **60** (1990) 165.

[13] R. MERTIG, M. BÖHM, and A. DENNER, Feyn Calc, *Comp. Phys. Comm.* **64** (1991) 345.

[14] A. J. BURAS and P. H. WEISZ, *Nucl. Phys.* **B333** (1990) 66.

[15] A. J. BURAS, M. JAMIN, M. E. LAUTENBACHER, and P. H. WEISZ, *Technical University of Munich preprint* **TUM-T31-16/91** , to appear in Nucl. Phys. **B**.

[16] M. VELTMAN, *Nucl. Phys.* **B319** (1989) 253.

[17] A. J. BURAS, M. JAMIN, M. E. LAUTENBACHER, and P. H. WEISZ, *Technical University of Munich preprint, in preparation* .

[18] S. Herrlich and J. Kalinowski, *Technical University of Munich preprint* **TUM-T31-15/91**.

[19] S. Herrlich and J. Kalinowski, *Technical University of Munich preprint* **TUM-T31-17/91**.

[20] P. Pascual and R. Tarrach, *QCD: Renormalization for the Practitioner*, Springer Lecture Notes in Physics, Vol. 194, 1984.

Appendices

A Collection of Some γ-Algebra Relations

In this appendix we give a collection of formulae for γ-algebra in d dimensions in the 't Hooft-Veltman scheme. For a detailed mathematical treatment of the subject see refs. [8, 9, 16] and [10].

Let us denote by

$$g_{\mu\nu}, \qquad \hat{g}_{\mu\nu}, \qquad \tilde{g}_{\mu\nu} \qquad \text{and} \qquad \gamma_\mu, \qquad \hat{\gamma}_\mu, \qquad \tilde{\gamma}_\mu \tag{12}$$

the metric tensor and γ-matrix in d, $d-4$ and 4 dimensions, respectively.

The contraction rules for the metric tensor in d dimensions are

$$g_{\mu\lambda}\, g_\nu^{\ \lambda} \;=\; g_{\mu\nu} \;, \qquad g_\mu^{\ \mu} \;=\; d \;, \tag{13}$$

with $g_{\mu\nu}$ being symmetric in its indices. The anticommutator of two γ-matrices in d dimensions is defined as

$$\{\gamma_\mu, \gamma_\nu\} \;=\; 2\, g_{\mu\nu}\, \mathbf{1} \;, \tag{14}$$

where $\mathbf{1}$ denotes the d-dimensional unit matrix. Without loss of generality the trace of the unit matrix can be chosen to be[5)]

$$Tr\,(\mathbf{1}) \;=\; 4 \;. \tag{15}$$

The contraction rules for the metric tensor and anticommutator of two γ-matrices in $d-4$ and 4 dimensions are given in a completely analogous manner, namely

$$\hat{g}_{\mu\lambda}\, \hat{g}_\nu^{\ \lambda} \;=\; \hat{g}_{\mu\nu} \;, \qquad \hat{g}_\mu^{\ \mu} \;=\; d-4 \;, \qquad \{\hat{\gamma}_\mu, \hat{\gamma}_\nu\} \;=\; 2\, \hat{g}_{\mu\nu}\, \mathbf{1} \tag{16}$$

and

$$\tilde{g}_{\mu\lambda}\, \tilde{g}_\nu^{\ \lambda} \;=\; \tilde{g}_{\mu\nu} \;, \qquad \tilde{g}_\mu^{\ \mu} \;=\; 4 \;, \qquad \{\tilde{\gamma}_\mu, \tilde{\gamma}_\nu\} \;=\; 2\, \tilde{g}_{\mu\nu}\, \mathbf{1} \;. \tag{17}$$

Because of the projection property of the metric tensor in $d-4$ and 4 dimensions we have the following contraction rules for metric tensors stemming from different dimensions

$$\hat{g}_{\mu\lambda}\, g_\nu^{\ \lambda} \;=\; \hat{g}_{\mu\nu} \;, \qquad \tilde{g}_{\mu\lambda}\, g_\nu^{\ \lambda} \;=\; \tilde{g}_{\mu\nu} \;, \qquad \hat{g}_{\mu\lambda}\, \tilde{g}_\nu^{\ \lambda} \;=\; 0 \;. \tag{18}$$

Therefore γ-matrices from dimensions $d-4$ and 4 anticommute

$$\{\hat{\gamma}_\mu, \tilde{\gamma}_\nu\} \;=\; 0 \;. \tag{19}$$

[5)] A different choice would only correspond to a finite renormalization which is however irrelevant for physical quantities.

Using again the projection property of the metric tensor one derives for an arbitrary string of γ-matrices sandwiched between two matrices from 4 and $d-4$ dimensional spaces

$$\tilde{\gamma}_\mu \gamma_{\alpha_1} \cdot \ldots \cdot \gamma_{\alpha_m} \hat{\gamma}^\mu \;=\; \hat{\gamma}_\mu \gamma_{\alpha_1} \cdot \ldots \cdot \gamma_{\alpha_m} \tilde{\gamma}^\mu \;=\; 0\,. \tag{20}$$

This in turn implies the relations

$$\gamma_\mu \gamma_{\alpha_1} \cdot \ldots \cdot \gamma_{\alpha_m} \tilde{\gamma}^\mu \;=\; \tilde{\gamma}_\mu \gamma_{\alpha_1} \cdot \ldots \cdot \gamma_{\alpha_m} \gamma^\mu \;=\; \tilde{\gamma}_\mu \gamma_{\alpha_1} \cdot \ldots \cdot \gamma_{\alpha_m} \tilde{\gamma}^\mu\,, \tag{21}$$
$$\gamma_\mu \gamma_{\alpha_1} \cdot \ldots \cdot \gamma_{\alpha_m} \hat{\gamma}^\mu \;=\; \hat{\gamma}_\mu \gamma_{\alpha_1} \cdot \ldots \cdot \gamma_{\alpha_m} \gamma^\mu \;=\; \hat{\gamma}_\mu \gamma_{\alpha_1} \cdot \ldots \cdot \gamma_{\alpha_m} \hat{\gamma}^\mu\,. \tag{22}$$

Let us define the matrix γ_5 to be

$$\gamma_5 \;=\; \frac{i}{4!}\,\varepsilon_{\mu_1\mu_2\mu_3\mu_4}\,\gamma^{\mu_1}\gamma^{\mu_2}\gamma^{\mu_3}\gamma^{\mu_4}\,, \tag{23}$$

with $\varepsilon_{\mu_1\mu_2\mu_3\mu_4}$ being the totally antisymmetric tensor in 4 dimensions. Then the trace of γ_5 with four γ-matrices is given by

$$Tr\,(\gamma_5\gamma_\mu\gamma_\nu\gamma_\lambda\gamma_\omega) \;=\; 4\,i\,\varepsilon_{\mu\nu\lambda\omega}\,. \tag{24}$$

One can show that $\gamma_5^2 = 1$ and that γ_5 anticommutes with a 4-dimensional γ-matrix, whereas it commutes with a $(d-4)$-dimensional γ-matrix [9]:

$$\{\gamma_5, \tilde{\gamma}_\mu\} \;=\; 0 \qquad \text{and} \qquad [\gamma_5, \hat{\gamma}_\mu] \;=\; 0\,. \tag{25}$$

From equation (25) one readily derives

$$\{\gamma_5, \gamma_\mu\} \;=\; 2\,\hat{\gamma}_\mu\,\gamma_5\,, \tag{26}$$

the key identity in the HV scheme with a non-anticommuting γ_5.

The ε-tensor is a 4-dimensional object in the HV scheme. Therefore contracting a γ-algebra object with the ε-tensor projects onto 4 dimensions and we obtain the identity

$$\varepsilon_{\mu\nu\lambda\omega}\gamma^\omega \;=\; i\,\gamma_5\left(\tilde{g}_{\mu\nu}\tilde{\gamma}_\lambda - \tilde{g}_{\mu\lambda}\tilde{\gamma}_\nu + \tilde{g}_{\nu\lambda}\tilde{\gamma}_\mu - \tilde{\gamma}_\mu\tilde{\gamma}_\nu\tilde{\gamma}_\lambda\right). \tag{27}$$

The contraction of two ε-tensors yields

$$\varepsilon_{\mu_1\mu_2\mu_3\mu_4}\varepsilon_{\nu_1\nu_2\nu_3\nu_4} = -\sum_{\pi\in S_4} \mathrm{sign}\,\pi \prod_{i=1}^{4} \tilde{g}_{\mu_i\nu_{\pi(i)}}\,, \tag{28}$$

with S_4 denoting the permutation group of four objects.

B A Sample Calculation Using Tracer

As an example for a possible application of the Tracer package we calculate the correlation function of two weak currents to lowest order perturbation theory in the HV-scheme. The corresponding Feynman diagram is shown in fig. 3.

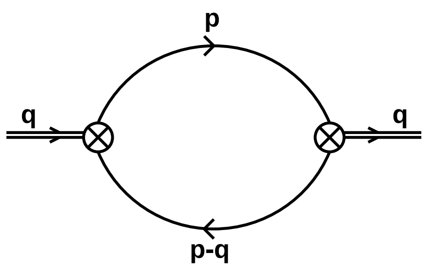

Figure 3: Correlation function of two weak currents to lowest order perturbation theory.

In momentum space the correlator is given by

$$\Pi_{\mu\nu}(q^2) \;=\; i\int d^dx \;\; e^{iqx}\,\langle 0\,|\,T\{j^W_\mu(x)\,j^W_\nu(0)\}|\,0\rangle\,, \tag{29}$$

with

$$j^W_\mu(x) \;\equiv\; \bar{q}_L(x)\,\gamma_\mu\,q_L(x) \;\equiv\; \bar{q}(x)\,\Gamma_\mu\,q(x) \tag{30}$$

and the γ-matrix structure $\Gamma_\mu = 1/4\,(1+\gamma_5)\,\gamma_\mu\,(1-\gamma_5)$. Note that in the HV-scheme γ_5 does not anticommute and hence $\Gamma_\mu \neq 1/2\,\gamma_\mu(1-\gamma_5)$ but rather $\Gamma_\mu = 1/2\,\tilde{\gamma}_\mu(1-\gamma_5)$. This relation can be easily shown using Tracer:

```
In[1]:= gammamu = 1/4 G[1, U+G5, {mu}, U-G5] // ToHatTilde // ToUG5

        G[ 1, ({mu}~), -G5 + U ]
Out[1]= ------------------------
                  2
```

Application of Wick's theorem and simple algebra leads to

$$\Pi_{\mu\nu}(q^2) \;=\; i\,N_c\,Tr\Big[\Gamma_\mu\gamma^\alpha\Gamma_\nu\gamma^\beta\Big]\,\nu^{2\varepsilon}\int\frac{d^dp}{(2\pi)^d}\;\frac{p_\alpha(p-q)_\beta}{p^2(p-q)^2}\,, \tag{31}$$

where for simplicity we have neglected the quark mass. N_c is the number of colors and ν is a renormalization scale in the MS-scheme. Here we do not want to go into the details of such a calculation. The interested reader is referred to the existing literature (for example the book of Pascual and Tarrach [20]). Evaluating the Feynman integral [20] we end up with the expression

$$\begin{aligned}\Pi_{\mu\nu}(q^2) \quad &= \quad -\frac{N_c}{32\,\pi^2}\left(\frac{\nu^2}{-q^2 e^{\gamma_E}}\right)^{\varepsilon} Tr\Big[\Gamma_\mu\gamma^\alpha\Gamma_\nu\gamma^\beta\Big]\cdot \\ &\cdot\Big\{\,g_{\alpha\beta}\,q^2+2\,(1-\varepsilon)\,q_\alpha q_\beta\,\Big\}\,\frac{\Gamma^2(2-\varepsilon)}{\Gamma(4-2\,\varepsilon)}\,\Gamma(\varepsilon-1)\,. \qquad (32)\end{aligned}$$

In this expression ν has been redefined to be a scale in the $\overline{MS}$-scheme, γ_E is the Euler constant and we have set the space-time dimension to $d=4-2\,\varepsilon$.

In our sample application we set $\nu^2 = -q^2$ for simplicity and disregard the common factor $-N_c/32\,\pi^2$. As an example it suffices to calculate only the coefficient of the Lorentz structure $g_{\mu\nu}$ in eq. (32) to order $\mathcal{O}(\varepsilon)$.

The corresponding MATHEMATICA program[6)] using functions defined in the TRACER package looks like:

```
(*********************************************************************)
(*          1-LOOP CORRELATION FUNCTION OF TWO WEAK CURRENTS         *)
(*                           Uses: Tracer.m                          *)
(*********************************************************************)

AntiCommute[off];          (* Use a non-anticommuting G5   *)
VectorDimension[4-2eps];   (* Set the space-time dimension *)

(* Define vertex structures Gamma_mu and Gamma_nu *)
Gammamu = 1/4 G[l,U+G5,{mu},U-G5];
Gammanu = 1/4 G[l,U+G5,{nu},U-G5];

Spur[ls];   (* Set the line index which will be traced later *)

(* Form the complete gamma structure of the diagram *)
gamma = Gammamu ** G[l,{alpha}] ** Gammanu ** G[l,{beta}] /. l->ls;

(* Calculate the trace by switching from line index l to ls;
   for further details see the entry on Spur[] in section 4.2 *)
gamma = gamma /. l -> ls;

(* Define the momentum structure resulting from the Feynman integral *)
qexpr = {alpha}.{beta} q.q + 2(1-eps) q.{alpha} q.{beta};

(* Define the product of the Gamma-functions *)
gammafun = Exp[eps EulerGamma] Gamma[2-eps]^2 Gamma[eps-1]/Gamma[4-2eps]

(* Calculate the product of the trace and the momentum structure *)
result1 = ExpandAll[gamma qexpr];

(* Remove the unphysical components of the external momentum *)
```

6)Of course, all the individual commands could be entered interactively as well.

```
result2 = RemoveHatMomenta[result1,q];

(* Laurent series of "result2 * gamma-functions" in terms of eps *)
result3 = Expand[ Normal[ Series[ result2 gammafun, {eps,0,1}] ] ];

(* Extract the coefficient of g_{mu nu} *)
result4 = Coefficient[ result3, {mu}.{nu}]
```

The code already contains numerous comments so we think giving just a brief outline of the main steps involved is sufficient:

- First, define the environment settings and the various expressions which are used during the calculation.
- Next, contract the γ-matrix structure with the momentum structure resulting from the Feynman integral.
- Finally, tailor the output according to ones needs; in this case remove the unphysical part of the external momentum q, expand the result in powers of `eps` using the MATHEMATICA built-in `Series[]` and extract the coefficient of the structure $g_{\mu\nu}$.

The transcript of the complete session running the above program is given below. For simplicity the TRACER startup messages have been suppressed.

```
In[1]:= << Tracer.m

(* ... suppressed startup messages go here ... *)

In[2]:= << Example.m

Package uses a non anticommuting G5 in "d" dimensions.
Dimension set to "4 - 2 eps". Used type of G5 left unchanged.
The gamma matrix line(s) "ls" will be traced.

                                         2
        26 q.q   4 q.q   160 eps q.q   Pi  eps q.q
Out[2]= ------ + ----- + ----------- - -----------
          9      3 eps       27             9
```

The reader is encouraged to find more efficient ways to perform this calculation using the TRACER package. Consider for instance changing the order in which the different steps in the calculation are performed, e.g. first contracting γ-matrices and momentum structure and then evaluating the trace.

C Redefined MATHEMATICA Built-in Functions

When implementing the TRACER package we found it useful to partially alter two of MATHEMATICA's built-in functions, namely `Dot[]` and `NonCommutativeMultiply[]` which have standard aliases `.` and `**`, respectively. To save the user running TRACER from a surprising behavior of the MATHEMATICA system that is not not documented in the manual [5] we comment on each change shortly.

The first modification is minor and should hardly affect ordinary MATHEMATICA usage. To allow for convenient input of Lorentz scalar products, e.g. `p.q` alternatively to the more awkward `S[p,q]`, we modify the standard MATHEMATICA `Dot[]` operator. The only change made is

```
Dot[a_,b_] := S[a,b] /; ( !Length[a] > 1 || Head[a] === Plus ) &&
                        ( !Length[b] > 1 || Head[b] === Plus );
```

This causes `Dot[]` to be automatically rewritten to the TRACER specific `S`-product if its arguments meet certain criteria. While previously e.g. `{mu}.{nu}` yielded `mu nu` it now transforms to `S[{mu},{nu}]` which of course will appear as `g_{mu nu}` on the output.

Secondly, changes have been made to the `NonCommutativeMultiply[]` function of MATHEMATICA. Since this may well cause considerably different behavior of the `**` command than found in the MATHEMATICA manual we recommend after TRACER has been run to restart a completely new session for applications relying on a standard `**`. Roughly speaking, changes were necessary to automatically extract pure scalar factors from a non-commuting product of γ-algebra objects. For instance while with the usual `**` the scalar expression $4-2\varepsilon$ in

```
G[l1,{la}]**(4-2 eps) G[l2,{mu}]**G[l3,{nu}]
```

$$\equiv \gamma_\lambda \otimes (4-2\,\varepsilon)\,\gamma_\mu \otimes \gamma_\nu$$

would be kept inside the tensor product it is automatically extracted to

```
(4-2 eps) G[l1,{la}]**G[l2,{mu}]**G[l3,{nu}]
```

$$\equiv (4-2\,\varepsilon)\,\gamma_\lambda \otimes \gamma_\mu \otimes \gamma_\nu$$

after the modifications. For a full list of changes made in connection with `NonCommutativeMultiply[]` the reader is referred to the actual code of the package.

D Package Index

In this appendix we give an alphabetically sorted list of all user available functions in the TRACER package together with their corresponding on-line documentation.

`AntiCommute[flag]`: Depending on `flag (on/off)` toggle the usage of an (non-) anticommuting matrix γ_5.
`AntiCommute[]`: Show the currently used type of γ_5.

`ContractEpsGamma[expr]` contracts index pairs appearing in `expr` according to the formula

$$\varepsilon_{\mu\nu\lambda\omega}\gamma^{\omega} \;=\; i\,\gamma_5\left(\tilde{g}_{\mu\nu}\tilde{\gamma}_{\lambda} - \tilde{g}_{\mu\lambda}\tilde{\gamma}_{\nu} + \tilde{g}_{\nu\lambda}\tilde{\gamma}_{\mu} - \tilde{\gamma}_{\mu}\tilde{\gamma}_{\nu}\tilde{\gamma}_{\lambda}\right)\,,$$

where as usual the tilde symbol denotes 4-dimensional objects.

`Eps[arg1,arg2,arg3,arg4]` is the completely antisymmetric product of the four arguments `arg1-4`. `Eps[]` is multi-linear with respect to a linear combination with scalar coefficients. Arguments of `Eps[]` can be a mixture of Lorentz indices and momenta, e.g. `Eps[[p,mu,q,nu]` means $\epsilon_{\alpha\mu\beta\nu}p^{\alpha}q^{\beta}$ in a standard physics notation.

`G[l,exp1,...,expn]` is a REDUCE-like shorthand for `GammaTrace[l,exp1, ...,expn]`. Type `?GammaTrace` for more information.

`GammaTrace[l,exp1,...,expn]` calculates the trace of a string of gamma matrices in arbitrary dimensions. The space-time dimension can be set with `VectorDimension[]`. The expi can be: `Symbols` (e.g. `p, q`) which denote vectors, brackets (e.g. {`mu`}, {`nu`}) which denote indices or expressions like `p+m U+a G5` where `U` is the unit matrix, `G5` is γ_5 and `m` and `a` are scalar coefficients. Indices which occur in pairs are contracted automatically. Different strings are distinguished by the line index `l`.

`G5` is the matrix γ_5 in `d` dimensional Gamma algebra.

`H@Symbol` is the projection of the `d` dimensional object `Symbol` onto `(d-4)` dimensions.
Display format: `H@p` $\longrightarrow$ `p^`.

`ListCommands[]` shows a list of the commands defined by the TRACER package.

`NoSpur[l1, ..., ln]` removes the gamma lines `l1, ..., ln` from being traced.

`OnShell[flag, list1, ..., listn]` : Depending on `flag` (`on/off`), the on-shell declarations specified in the `listi` are enforced or removed.
Example: `OnShell[on, {p1,m^2}, {s1,p1,0}, {e,-1}]` declares the substitutions `S[p1,p1]=m`^{2}, `S[s1,p1]=0, S[e,e]=-1`.
`OnShell[]` : Show all currently active on-shell declarations.

`OutputFormat[flag]` : Depending on `flag` (`texlike/subscript`) toggle the output format of the scalar product `S[expr]` between TEX-input-like and subscripted style.
`OutputFormat[]`: Show the currently used output format for `S[expr]`.
For examples of the display formats see the help entry on `S`.

`RemoveHatMomenta[expr, p1, p2, ...]` sets all `d-4` dimensional momenta `p1^`, `p2^`, ... in `expr` equal to 0.

`RemoveNCM[flag]`: Depending on `flag` (`on/off`) toggle whether `NonCommutativeMultiply[]` should be removed or kept if all gamma lines are different.
`RemoveNCM[]`: Show current treatment of `NonCommutativeMultiply[]`.

`S[arg1,arg2]` is the scalar product of the two arguments. `S` is symmetric and bilinear with respect to a linear combination with scalar coefficients. Use of the `Dot[]` operator for the scalar product is possible.
Display formats:

Input:			`OutputFormat[flag]`:	
(explicit)	(shorthand)		`texlike`	`subscript`
`S[p, q]`	`p.q`	⟶	`p.q`	`p.q`
`S[p,{mu}]`	`p.{mu}`	⟶	`p_{mu}`	`p` `  mu`
`S[{mu},{nu}]`	`{mu}.{nu}`	⟶	`g_{mu nu}`	`g` `  mu nu`

`Sigma[ l,p,q ]` is a shorthand notation for `I/2(G[l,p,q]-G[l,q,p])`. It is decomposed immediately into the equivalent gamma matrix strings. In the output of `ToDiracBasis[]` this automatic decomposition is prevented by an internal application of the Hold[] command invisible to the user. Thus, to use any `Sigma[]` stemming from the output of `ToDiracBasis[]` in further calculations you first have to apply a `Release[]` on the output.

`SortLine[ expr, l, reflist ]` sorts the product of gamma matrices on line `l` in expression `expr` with respect to the reference list `reflist`. If `l` does not match exactly one gamma matrix line in `expr`, `SortLine[]` returns `expr` unchanged.
Example:
`SortLine[ G[l1,p,{m},{n},q] G[l2,p,q], l1, {{m},p,{n},q}]`
⟶ `(2 p_{m} G[l1,{n},q] - G[l1,{m},p,{n},q]) G[l2,p,q]`.

`Spur[l1, ..., ln]` adds the γ-matrix lines with line indices `l1, ..., ln` to the set of lines to be traced.
`Spur[]` shows the currently used gamma lines which should be traced.

`T@Symbol` is the projection of the `d` dimensional object `Symbol` onto 4 dimensions.
Display format: `T@p` $\longrightarrow$ `p~`.

`ToDiracBasis[expr]` decomposes an expression containing strings `G[...]` of gamma matrices into the basis spanned by 1, γ_5, γ_μ, $\gamma_\mu\gamma_5$ and $\frac{i}{2}[\gamma_\mu, \gamma_\nu]$ for all fermion lines appearing in `expr`. Since this basis is valid for space-time dimensions equal to 4 only, the routine tests whether `VectorDimension[4]` has been set and if not issues a warning and exits.
`ToDiracBasis[expr,l1,l2,...]` decomposes only fermion lines `l1`, `l2`, etc.. in `expr`.
In the output of `ToDiracBasis[]` the automatic decomposition of `Sigma[]` to gamma strings is prevented by an internal application of the Hold[] command invisible to the user. Thus, to use any `Sigma[]` stemming from the output of `ToDiracBasis[]` in further calculations you first have to apply a `Release[]` on the output."
Note that applying `ExpandAll[]` on expressions containing held `Sigma[]` terms leads to weird output. Use commands `ExpandNumerator[]` and `ExpandDenominator[]` successively instead.

`ToHatTilde[expr]` decomposes `d` dimensional gamma algebra objects according to the formula $\gamma_\mu = \tilde{\gamma}_\mu + \hat{\gamma}_\mu$ into 4 and `d-4` dimensional objects, respectively. 4-dimensional objects which's default redecomposition into `d` and `d-4` ones is delayed via the `Hold[T[]]` command are displayed with parentheses around them, e.g. `Hold[T@{mu}] --> ({mu}~)`. To go back to the `d` and `d-4` objects just do a `Release[]` on the expression containing 4 dimensional objects.
Note that applying `ExpandAll[]` on expressions containing terms like `Hold[T@mu]` leads to weird output. Use commands `ExpandNumerator[]` and `ExpandDenominator[]` successively instead.

`ToOtimes[expr]` transforms all products of the form `Times[G[...], G[...], ...]` to `Otimes[G[...], G[...], ...]` preserving the order of the `G[...]`'s. It is most usefully used as `expr // ToOtimes // TeXForm` acting as a filter to give output of `expr` in `TeXForm` with the `\otimes` symbol between `G[...]`'s stemming from different lines. Note that `ToOtimes[expr]` is needed in `G[...] G[...] // TeXForm` but not in `G[...]**G[...]  // TeXForm` since the built-in `TeXForm` of `**` is already `\otimes`.

ToUG5[expr] decomposes all matching fermion lines appearing in expr according to the formula

$$a\,\Gamma_1\Gamma_2 + b\,\Gamma_1\gamma_5\Gamma_2 = \frac{a+b}{2}\,\Gamma_1(1+\gamma_5)\Gamma_2 + \frac{a-b}{2}\,\Gamma_1(1-\gamma_5)\Gamma_2\;,$$

with $a \neq 0 \wedge b \neq 0$.
ToUG5[expr,{l1,l2,...}] decomposes only fermion lines l1, l2, ... in expr into $1 \pm \gamma_5$ terms.

U is the d dimensional unit matrix. For instance U is used in connection with mass terms as G[l, p+m U].

VectorDimension[expr] : Set the space-time dimension to expr. The default symbol for the dimension is Global`d. Do not use the symbol D for the dimension since this may mix up with the differential operator System`D.
VectorDimension[] : Show the currently used expression for the space-time dimension.

Version[] prints out some information on the current version of the TRACER package.