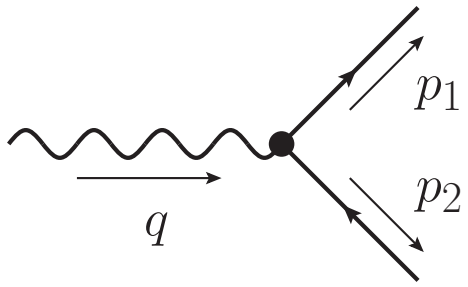
Simple FORM example

Let us consider the QED process, when the virtual photon decays into electron-positron pair:



Corresponding amplitude equals to the following:

$$\mathcal{M} = \epsilon_\mu^\lambda(q)\, \bar{u}(p_1)^{h_1}\, \gamma^\mu\, v(p_2)^{h_2}. \tag{1}$$

Suppose we need to calculate unpolarized the cross section of this process. Thus we need to take (1) squared, average it over photon helicity ($\lambda = \pm 1$) and sum over fermion ($h_1,\ h_2 = \pm 1/2$) helicities (assume we are in the center-of-mass frame).

Applying the completeness relation for the photon polarization vectors

$$\sum_\lambda \epsilon_\mu^\lambda(q)\epsilon_\nu^{*\lambda}(q) = -g_{\mu\nu} \tag{2}$$

and for the fermion spinors

$$\sum_{h_1} u^{h_1}(p_1)\bar{u}^{h_1}(p_1) = \not{p}_1 + m, \tag{3}$$

$$\sum_{h_2} v^{h_2}(p_2)\bar{v}^{h_2}(p_2) = \not{p}_2 - m, \tag{4}$$

we get the following expression:

$$\frac{1}{2}\sum_\lambda \sum_{h_1,h_2} |\mathcal{M}|^2 = -\frac{1}{2} g_{\mu\nu}\, \mathrm{Tr}[(\not{p}_1 + m)\gamma^\mu(\not{p}_2 - m)\gamma^\nu] \tag{5}$$

Let us calculate $-g_{\mu\nu}\,\mathrm{Tr}[(\not{p}_1 + m)\gamma^\mu(\not{p}_2 - m)\gamma^\nu]$ using FORM. The simplest FORM code for this problem is given below. It uses the gamma-matrices that are built into FORM.

```
#-
Dimension 4;
I mu, nu;
V q;
autodeclare vector p;
symbol m;

nwrite statistics;

local Amp2 = - d_(mu,nu)*(g_(1,p1)+m)*
g_(1,mu)*(g_(1,p2)-m)*g_(1,nu);

trace4,1;

contract;

id p1.p2 = q.q/2-m^2;

print;

.end
```

Each FORM program begins from the variables declaration. The variables have different types: indices `I`, vectors `V`, tensors `ntensor` autodeclaring vectors `autodeclare vector` or indices `autodeclare index`, symbols `symbol` and even more. The dimension of the working space is 4 by definition, but can be changed by typing `Dimension` ⟨*necessary number*⟩ before the allocations of variables.

Function `nwrite statistics` is used to exclude the usually unnecessary statistical information from the output.

As you see, the main function `Amp2` follows after variable allocation and is expressed in terms of the metric (**Euclidian** - FORM does not have a built in Minkowski metic) tensor `d_(mu,nu)` that depends on two indices `mu` and `nu`, gamma-matrices `g_(1,mu)`, `g_(1,nu)` (depends on index mu)

and `g_(1,p1)`, `g_(1,p2)` (dependence on some vector p depicts $\not{p}$). The first number in brackets of gamma-matrices means the number of set of gamma-matrices that you can operate (e.g taking trace) independently.

```
10   local Amp2 = - d_(mu,nu)*(g_(1,p1)+m)*
11   g_(1,mu)*(g_(1,p2)-m)*g_(1,nu);         *the main expression.
```

As natural in mathematical calculations, after writing the general expression, its simplifications follow. In our case they include taking trace over the one set of gamma-matrices, contraction over dummy indices (usually one can skip this string - FORM do this automatically), some particular substitution (`p1.p2` means the scalar product $p_1 \cdot p_2$) and printing the result to the terminal. Be aware, that simplifications run in the same order as you write them!

The code ends by `.end`

```
13   trace4,1;
14
15   contract;
16
17   id p1.p2 = q.q/2-m^2;
18
19   print;
20
21   .end
```

Also there exists another way to solve this problem, without using the built-in gamma-matrices, defining their algebra by hands.

Let us declare some tensor `g` that will be our gamma-matrices. then, we

```
5    ntensor g;
```

rewrite the main function in terms of g:

```
12   local Amp2 = - d_(mu,nu)*(g(p1)+m)*g(mu)*(g(p2)-m)*g(nu);  *the main expression.
```

After taking a trace we can merge all products of `g` in the following way

```
14   * ---- Merging tensors. ?a means everything inside the brackets. ----
15
16   repeat;
17   id g(?a)*g(?b)=g(?a,?b);
18   endrepeat;
```

and define necessary relations of the Dirac algebra

```
20   * Dirac algebra -------------------------
21
22   id g(i?) = 0;
23   id g(i?,i?) = 16;
24   id g(i1?,i2?,i3?)=0;
25
26   id g(i1?,i2?,i3?,i4?) = d_(i1,i2)*g(i3,i4)
27   -d_(i1,i3)*g(i2,i4)
28   +d_(i1,i4)*g(i2,i3);
29
30   id g(i1?,i2?) = 4*d_(i1,i2);
```

Here `i?` means "arbitrary index" (similarly, if p is the vector, `p?` means "arbitrary vector" etc.). `i` should be declared in the beginning of the code, in our case `autodeclare index i;`.

Both of the approaches give the same result

```
FORM 4.2 (Nov  2 2019) 64-bits
    #-

   Amp2 =
      8*m^2 + 4*q.q;

  0.00 sec out of 0.00 sec
```