Generating Graphs in Form

Thoughts, Opinions, and More.

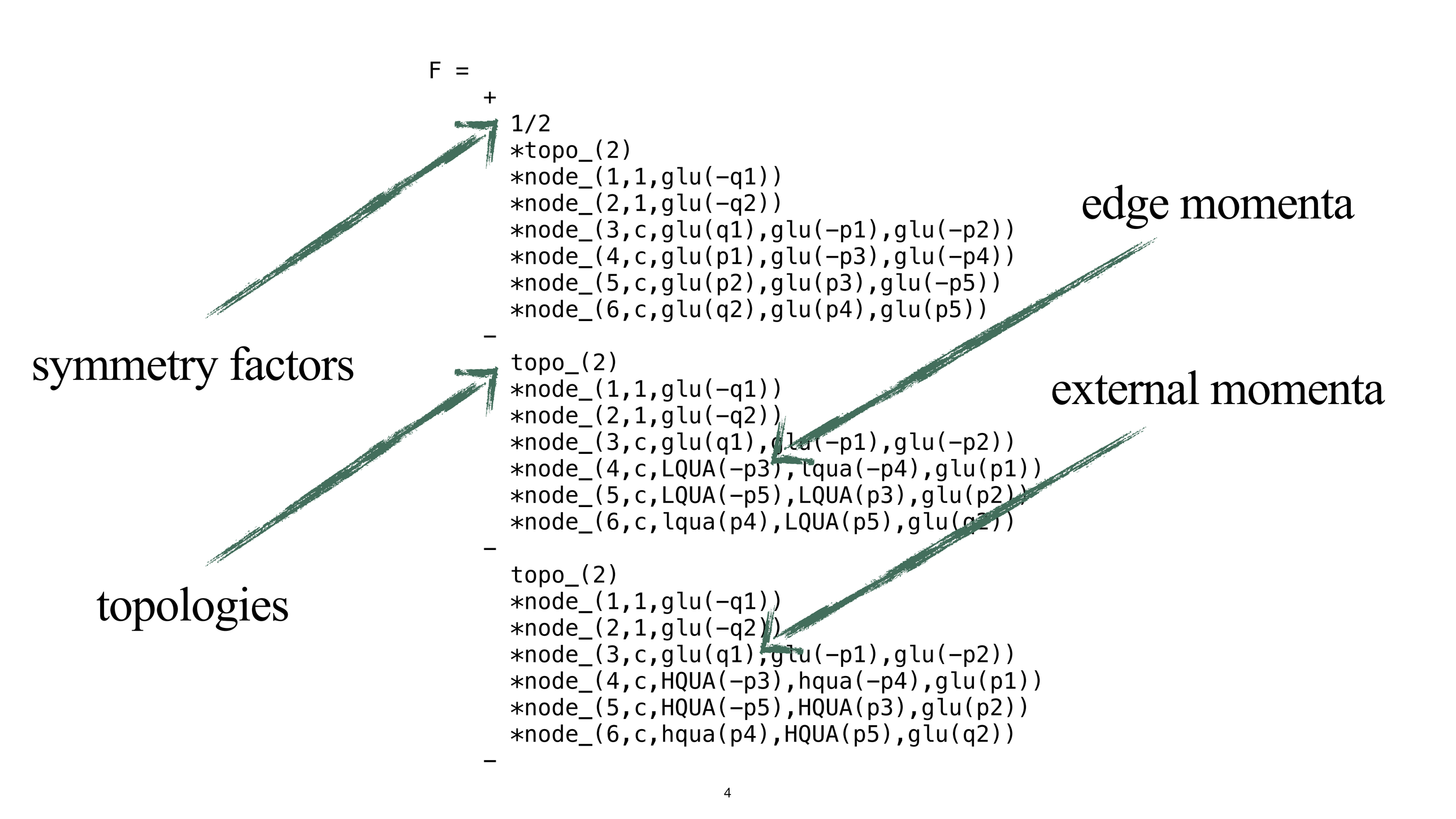
Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Marco Klann, 2026

Why should the generator be used?

What does the generator do?


F =
+
1/2
*topo_(2)
*node_(1,1,glu(-q1))
*node_(2,1,glu(-q2))
*node_(3,c,glu(q1),glu(-p1),glu(-p2))
*node_(4,c,glu(p1),glu(-p3),glu(-p4))
*node_(5,c,glu(p2),glu(p3),glu(-p5))
*node_(6,c,glu(q2),glu(p4),glu(p5))
-
topo_(2)
*node_(1,1,glu(-q1))
*node_(2,1,glu(-q2))
*node_(3,c,glu(q1),glu(-p1),glu(-p2))
*node_(4,c,LQUA(-p3),lqua(-p4),glu(p1))
*node_(5,c,LQUA(-p5),LQUA(p3),glu(p2))
*node_(6,c,lqua(p4),LQUA(p5),glu(q2))
-
topo_(2)
*node_(1,1,glu(-q1))
*node_(2,1,glu(-q2))
*node_(3,c,glu(q1),glu(-p1),glu(-p2))
*node_(4,c,HQUA(-p3),hqua(-p4),glu(p1))
*node_(5,c,HQUA(-p5),HQUA(p3),glu(p2))
*node_(6,c,hqua(p4),HQUA(p5),glu(q2))
-
symmetry factors
topologies
edge momenta
external momenta

What would I expect?

labels

```
*** TO QGRAF FORMAT

id node_(x1?,x2?,?a) = V(?a);
id V(x?) = 1;
$count = count_(V,1);

*** ASSIGN INDICES

Argument V;
  repeat id P?(+p?SETp[x]) = P(+p,+x);
  repeat id P?(-p?SETp[x]) = P(-p,+x);
  repeat id P?(+q?SETq[x]) = P(-q,
+x+`MAXEDGES');
EndArgument;
.sort
```

```
F =
      +
        1/2
        *topo_(2)
        *V(glu(-p1,1),glu(-p2,2),glu(-q1,6))
        *V(glu(-p3,3),glu(-p4,4),glu(p1,1))
        *V(glu(-p5,5),glu(p2,2),glu(p3,3))
        *V(glu(-q2,7),glu(p4,4),glu(p5,5))
      -
        topo_(2)
        *V(glu(-p1,1),glu(-p2,2),glu(-q1,6))
        *V(glu(-q2,7),lqua(p4,4),LQUA(p5,5))
        *V(glu(p1,1),lqua(-p4,4),LQUA(-p3,3))
        *V(glu(p2,2),LQUA(-p5,5),LQUA(p3,3))
      -
        topo_(2)
        *V(glu(-p1,1),glu(-p2,2),glu(-q1,6))
        *V(glu(-q2,7),hqua(p4,4),HQUA(p5,5))
        *V(glu(p1,1),hqua(-p4,4),HQUA(-p3,3))
        *V(glu(p2,2),HQUA(-p5,5),HQUA(p3,3))
```

```
*** PREPARE EQUATIONS

Multiply edges($count-1+`LOOP');
#do i = 1, `MAXEDGES'
  id edges(`i') = momenta(r1,...,r`i');
#enddo
id V(?a) = V(?a)*eq(?a);
Repeat id eq(?a1,P?(p?,x?),?a2) = eq(?a1,p,?a2);
Argument eq;
  #do i = 1, `MAXEDGES'
    id p`i' = r`i';
  #enddo
EndArgument;

*** PREPARE CHECK

Transform eq addargs(1,last);
id eq(?a) = eq(?a)*check(eq(?a));
ChainIn check;
Repeat id check(?a1,eq(p?),?a2) = check(?a1,p,?a2);

*** SOLVE EQUATIONS

Repeat;
  Transform eq addargs(1,last);
  id once eq(?a) = tmp(?a);
  SplitArg tmp;
  Repeat id tmp(?a1,p?,?a2) = tmp(?a1,f(p),?a2);
  Argument tmp;
    FactArg f;
    id f(p?) = f(1,p);
  EndArgument;
  id tmp(?a1,f(x?,r?SETr),?a2) = tmp(x,r,?a1,?a2);
  Transform tmp addargs(3,last);
  id tmp(x1?,p?,x2?) = tmp(p,-1/x1*x2);
  Argument tmp;
    id f(x?,p?) = x*p;
  EndArgument;
  id tmp(p1?,p2?) = replace_(p1,p2);
  id tmp(0) = 1;
EndRepeat;

*** LABEL MOMENTA

id momenta(?a) = tmp(?a)*momenta(?a);
SplitArg tmp;
Normalize (0) tmp;
Transform tmp dedup(1,last);
Repeat id tmp(?a1,p?{q},?a2) = tmp(?a1,?a2);
id tmp(p1?,...,p`LOOP'?) = replace_(<p1,k1>,...,<p`LOOP',k`LOOP'>);
```

loop momenta

```
F =
   -
     topo_(2)
     *V(glu(Q,7),glu(p4,4),glu(p5,5))
     *V(lqua(p2,2),LQUA(p3,3),glu(-p5,5))
     *V(LQUA(-p1,1),lqua(-p2,2),glu(-Q,6))
     *V(LQUA(-p3,3),LQUA(p1,1),glu(-p4,4))
     *mom(k1, - Q - k1,k2,k1 - k2, - Q - k1 + k2)
   -
     topo_(2)
     *V(glu(Q,7),glu(p4,4),glu(p5,5))
     *V(hqua(p2,2),HQUA(p3,3),glu(-p5,5))
     *V(HQUA(-p1,1),hqua(-p2,2),glu(-Q,6))
     *V(HQUA(-p3,3),HQUA(p1,1),glu(-p4,4))
     *mom(k1, - Q - k1,k2,k1 - k2, - Q - k1 + k2)
   -
     topo_(2)
     *V(glu(Q,7),glu(p4,4),glu(p5,5))
     *V(gho(p2,2),GHO(p3,3),glu(-p5,5))
     *V(GHO(-p1,1),gho(-p2,2),glu(-Q,6))
     *V(GHO(-p3,3),GHO(p1,1),glu(-p4,4))
     *mom(k1, - Q - k1,k2,k1 - k2, - Q - k1 + k2)
```

What should I not expect?

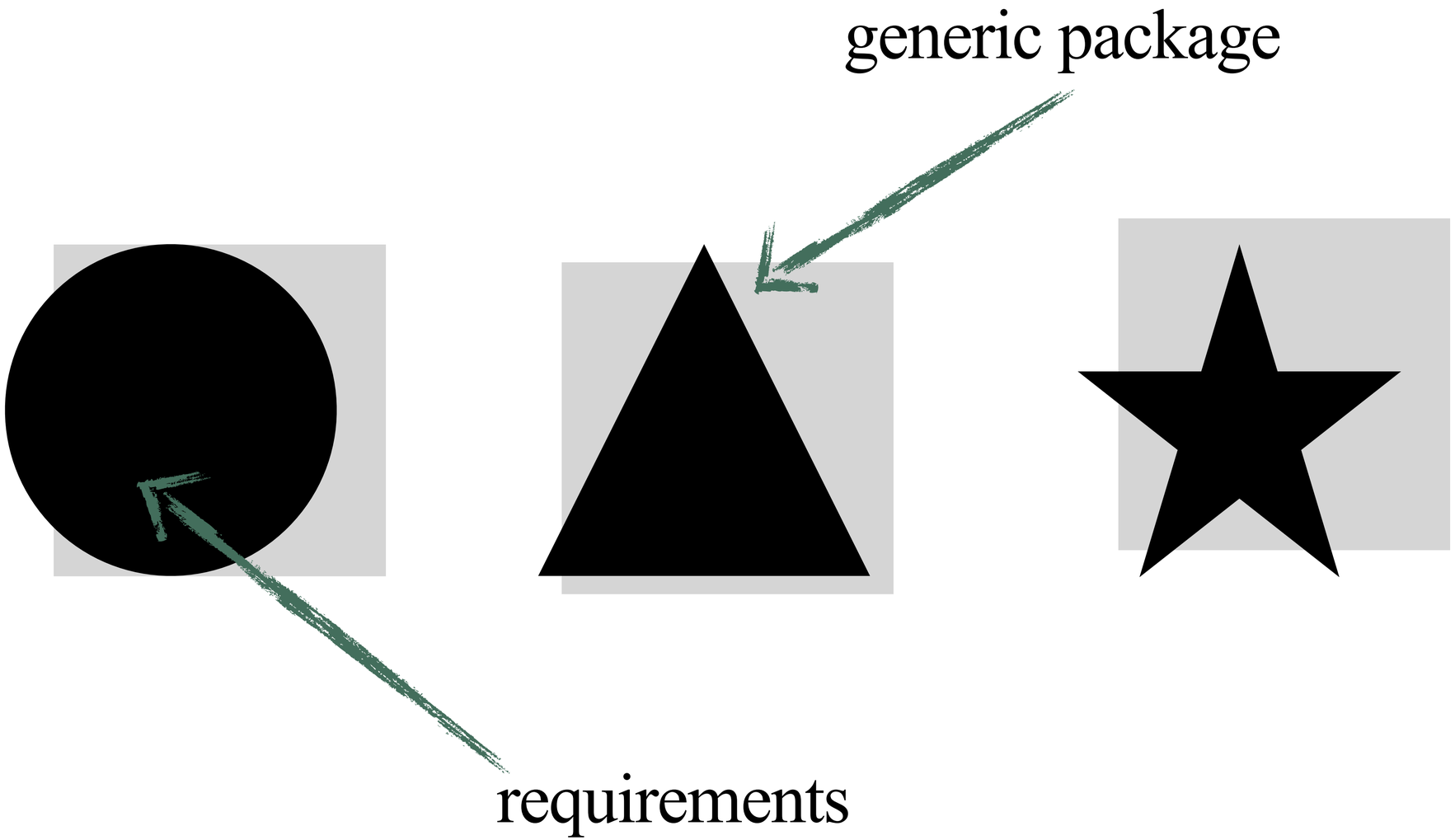
```
qvqv;f3x000011011;fhh011*nl*nh;CF*TF^2;amplitude(tr(Vqv(1,nu3),Dex(1,P),Vqv(1,nu2),Dlq(1,P+q3))*tr(Vqv(2,
qvqv;f3x110011110;fhh011*nh^2;CF*TF^2;amplitude(tr(Vqv(1,nu3),Dex(1,P),Vqv(1,nu2),Dlq(1,P+q3))*tr(Vqv(2,N
qvqv;f3x111100010;fhh020*nh;TF^3*d33*NR^-1;-1/2*amplitude(tr(Vqv(1,MU2),Dhq(1,q4),Vqv(1,mu10),Dhq(1,q2),V
qvqv;f3x000011011;fhh020*nh;TF^3*d33*NR^-1;-1/2*amplitude(tr(Vqv(1,MU2),Dhq(1,-q5),Vqv(1,mu9),Dhq(1,-q8),
qvqv;f3x111100010;fhh020*nh;CF^2*TF;-amplitude(tr(Vqv(1,MU2),Dhq(1,q2),Vqv(1,mu4),Dhq(1,P+q4),Vqv(1,nu6),
qvqv;f3x111111001;fhh020*nh;CF^2*TF;-amplitude(tr(Vqv(1,MU2),Dhq(1,-q3),Vqv(1,mu5),Dhq(1,-q1),Vqv(1,nu9),
qvqv;f3x110011110;fhh020*nh;CF^2*TF;-amplitude(tr(Vqv(1,MU2),Dhq(1,-q5),Vqv(1,mu9),Dhq(1,-q8),Vqv(1,mu6),
qvqv;f3x111100010;fhh020*nh;CA*CF*TF;-1/2*amplitude(tr(Vqv(1,MU2),Dhq(1,q2),Vqv(1,mu5),Dhq(1,q4),Vqv(1,mu
qvqv;f3x111111001;fhh020*nh;CA*CF*TF;1/2*amplitude(tr(Vqv(1,MU2),Dhq(1,-q3),Vqv(1,mu5),Dhq(1,-q1),Vqv(1,n
qvqv;                                                                                                 /qv(1,
qvqv;                                                                                                 qv(1,m
qvqv;                                                                                                 nu8),D
qvqv;                                                                                                 nu8),
qvqv;                                                                                                 qv(2,N
qvqv;                                                                                                 ,P+q8
qvqv;                                                                                                 ,nu10
qvqv;f3x000000000;fll110*nl;CA*CF*TF;-1/2*amplitude(tr(Vqv(1,MU2),Dlq(1,P-q1),Vqv(1,nu4),Dex(1,P),Vqv(1,n
qvqv;f3x000011011;fll110*nl*nh;CF*TF^2;amplitude(tr(Vqv(1,MU2),Dlq(1,P-q3),Vqv(1,nu4),Dex(1,P),Vqv(1,nu3)
qvqv;f3x000000000;fll110*nl^2;CF*TF^2;amplitude(tr(Vqv(1,MU2),Dlq(1,P-q3),Vqv(1,nu4),Dex(1,P),Vqv(1,nu3),
qvqv;f3x111100010;fll200*nh;TF^3*d33*NR^-1;-1/2*amplitude(tr(Vqv(1,MU2),Dex(1,P),Vqv(1,nu5),Dlq(1,P+q7),V
qvqv;f3x000011011;fll200*nh;TF^3*d33*NR^-1;-1/2*amplitude(tr(Vqv(1,MU2),Dlq(1,q4),Vqv(1,nu7),Dlq(1,q2),Vq
qvqv;f3x111100010;fll200*nh;CF^2*TF;-amplitude(tr(Vqv(1,MU2),Dlq(1,P+q6),Vqv(1,nu4),Dex(1,P),Vqv(1,nu5),D
qvqv;f3x000011011;fll200*nh;CF^2*TF;-amplitude(tr(Vqv(1,MU2),Dlq(1,P-q1),Vqv(1,nu5),Dlq(1,-q7),Vqv(1,nu10
qvqv;f3x110011110;fll200*nh;CF^2*TF;-amplitude(tr(Vqv(1,MU2),Dlq(1,P-q3),Vqv(1,nu4),Dex(1,P),Vqv(1,nu5),D
```

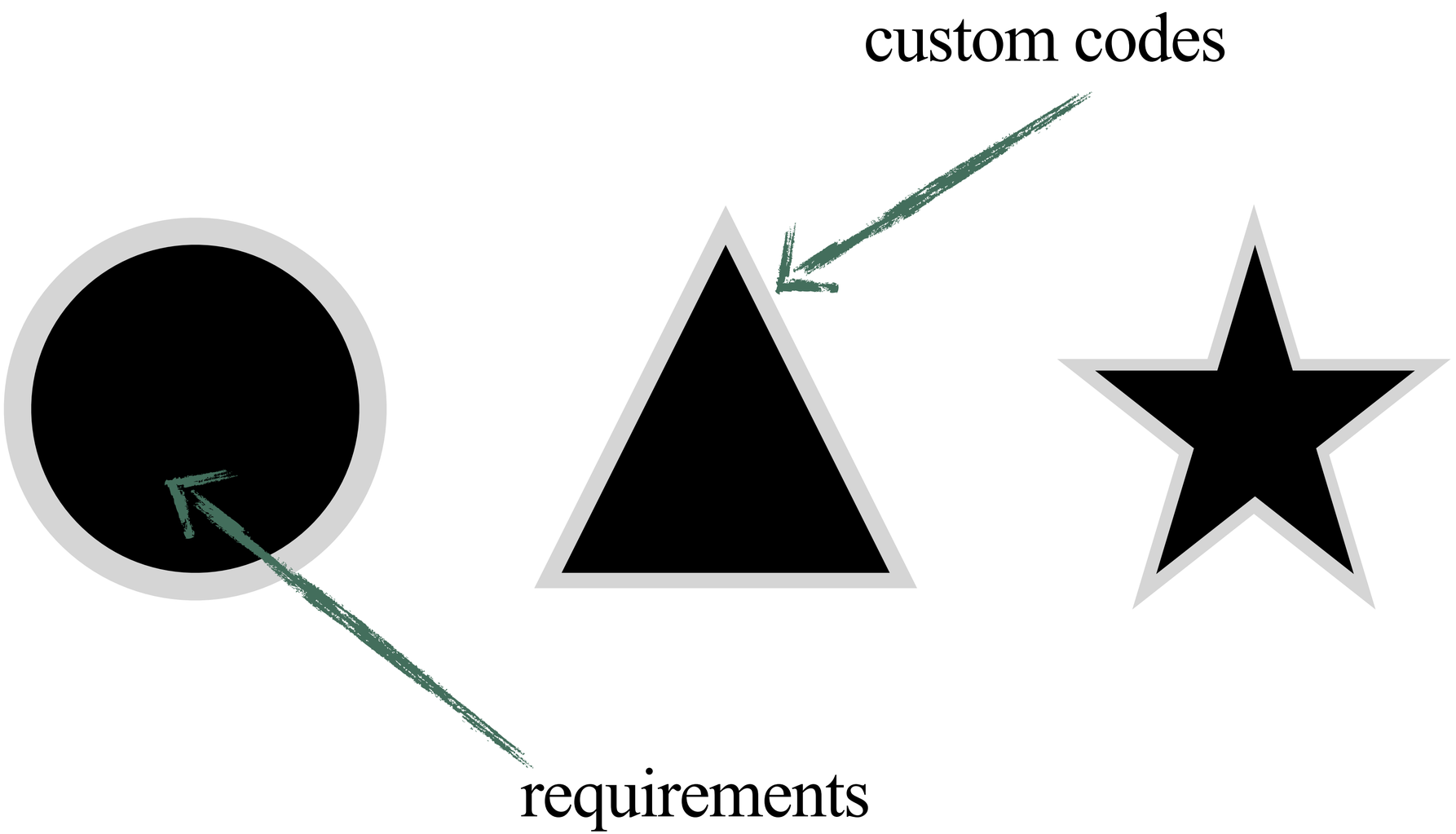
My preferred output.

More realistically…

```
F =
    +
      sym_(1/2)
      *topo_(2)
      *node_(1,1,glu(-Q))
      *node_(2,1,glu(Q))
      *node_(3,c,glu(Q),glu(-p1),glu(-p2))
      *node_(4,c,glu(p1),glu(-p3),glu(-p4))
      *node_(5,c,glu(p2),glu(p3),glu(-p5))
      *node_(6,c,glu(-Q),glu(p4),glu(p5))
      *mom_(k1, - Q - k1,k2,k1 - k2, - Q - k1 + k2)
    +
      sym(-1)
      *topo_(2)
      *node_(1,1,glu(-Q))
      *node_(2,1,glu(Q))
      *node_(3,c,glu(Q),glu(-p1),glu(-p2))
      *node_(4,c,LQUA(-p3),lqua(-p4),glu(p1))
      *node_(5,c,LQUA(-p5),LQUA(p3),glu(p2))
      *node_(6,c,lqua(p4),LQUA(p5),glu(-Q))
      *mom_(k1, - Q - k1,k2,k1 - k2, - Q - k1 + k2)
```

Generation ends, processing begins.


generic package
requirements

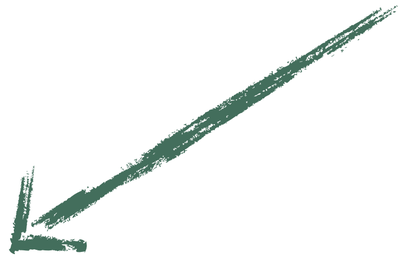

custom codes
requirements

”*FORM is hard to learn, there are few resources.*”
— someone somewhere

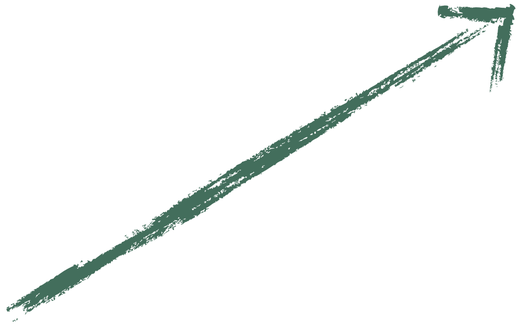
Functions for Forward Scattering

Ex. 1. Trivial Graphs

dangerous with masses



`NoTadpole_' + `NoSnail_'



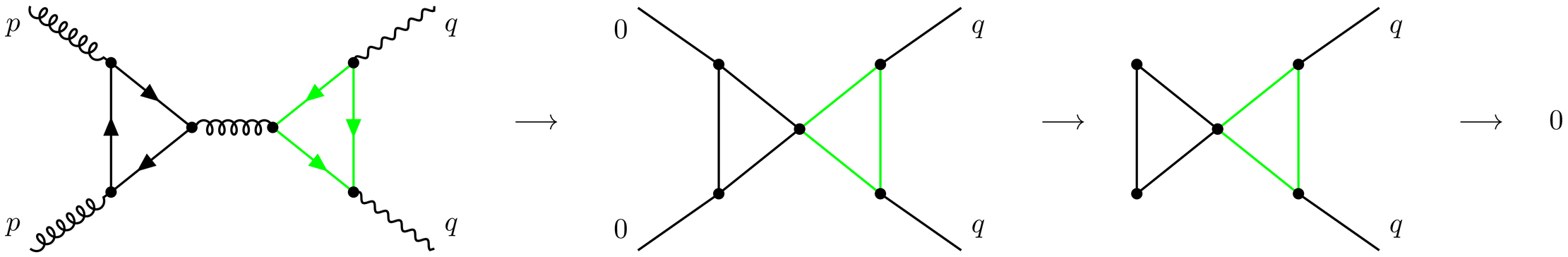
may not remove all integrals without a scale

Ex. 1. Trivial Graphs

scaleless integral

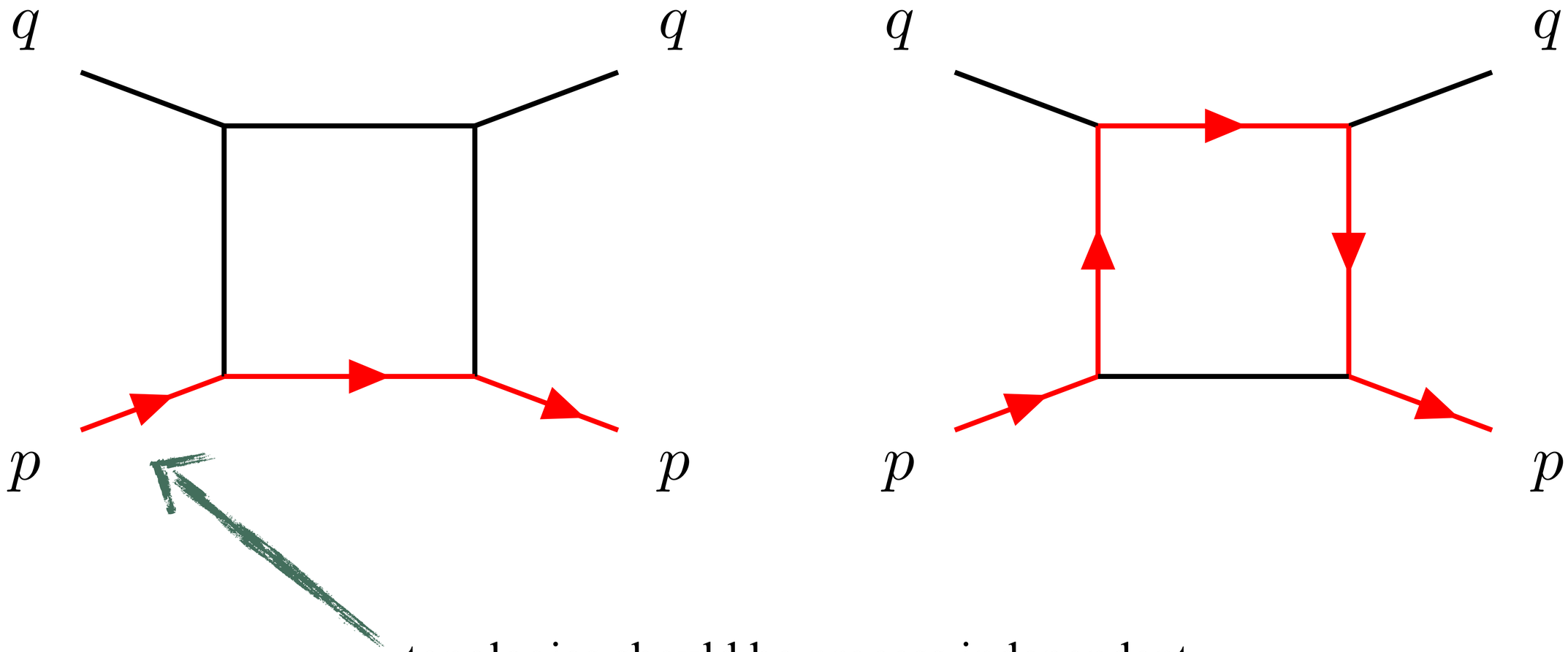
$$\int \mathrm{d}k_1 \ldots \mathrm{d}k_L \; f(k_1, \ldots, k_L; p_1, \ldots, p_N, m_1, \ldots m_E) = \int \mathrm{d}k_1 \ldots \mathrm{d}k_{L'} \; f(k_1, \ldots, k_{L'}) \times \int \mathrm{d}k_{L'+1} \ldots \mathrm{d}k_L \; f(k_{L'+1}, \ldots, k_L; p_1, \ldots, p_N, m_1, \ldots m_E)$$

one-vertex connectivity



(expansion at p = 0)

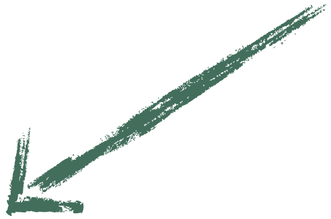
Ex. 2. Shortest Paths



topologies should be process independent

Ex. 2. Shortest Paths

labels independent topologies



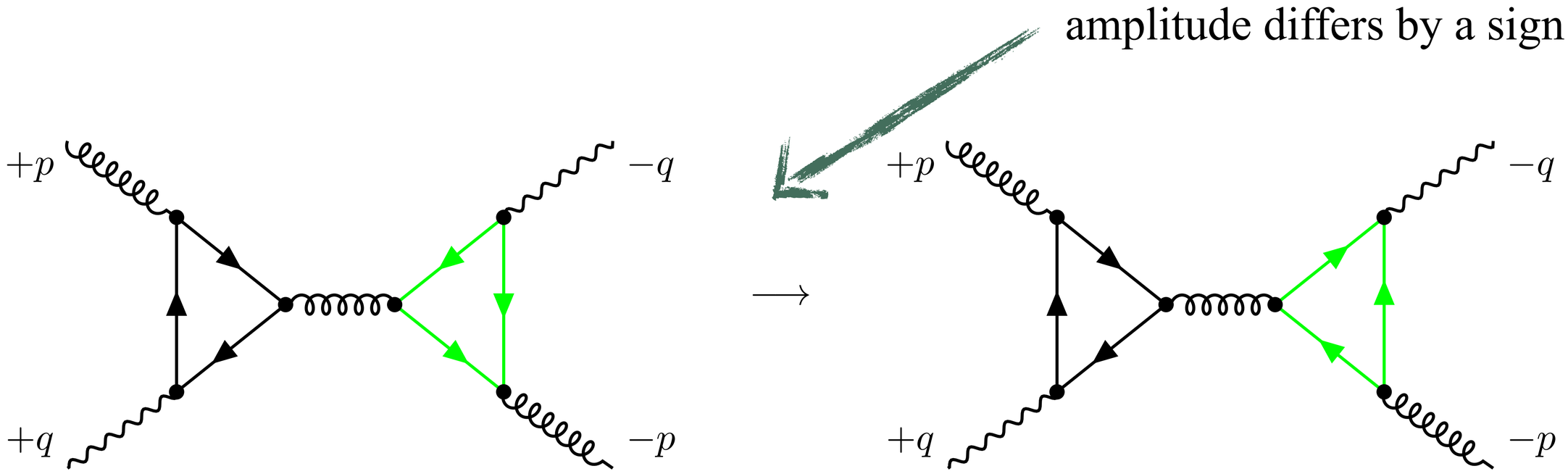
$$\text{topo\_(N)}$$

$$\text{May } \text{topo\_(N)} = \text{topo\_(M)} \text{ for } N \neq M \text{ ?}$$

Occurs with different processes in *same* code:

$$(g\gamma \rightarrow g\gamma) \ \wedge \ (g\phi \rightarrow g\phi)$$

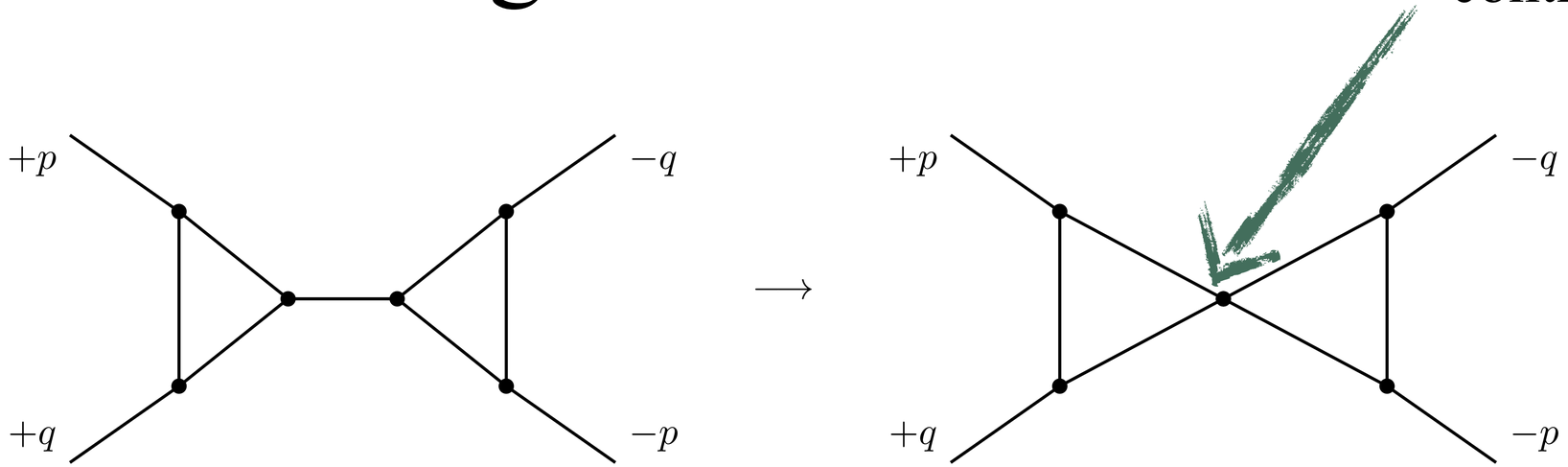
Ex. 3. Conjugation



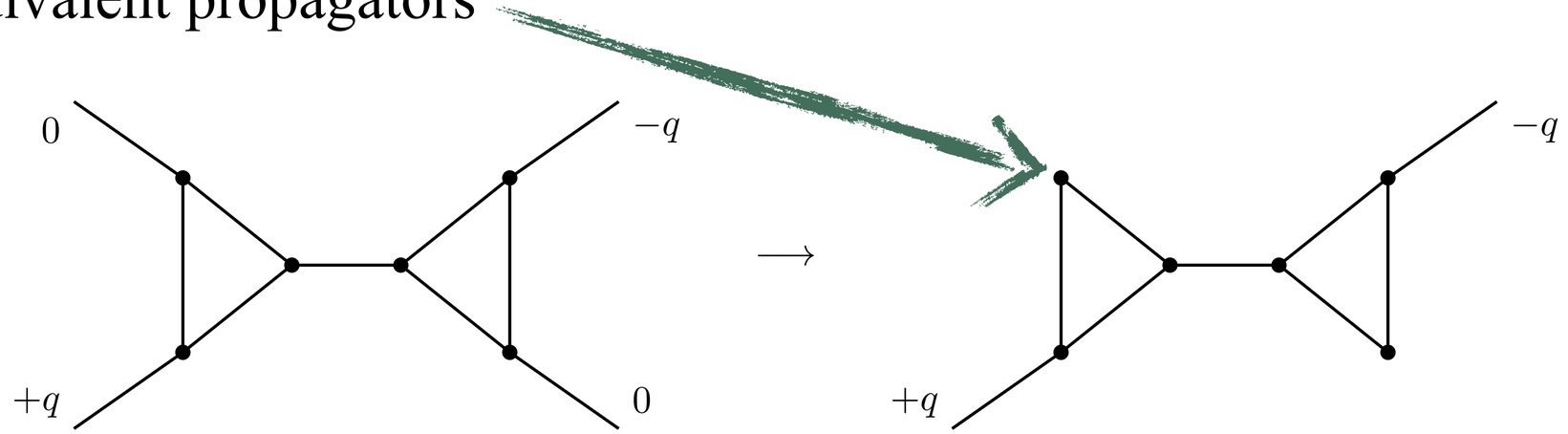
(A)symmetries are not determined by graph automorphisms

Ex. 4. Towards Integrals

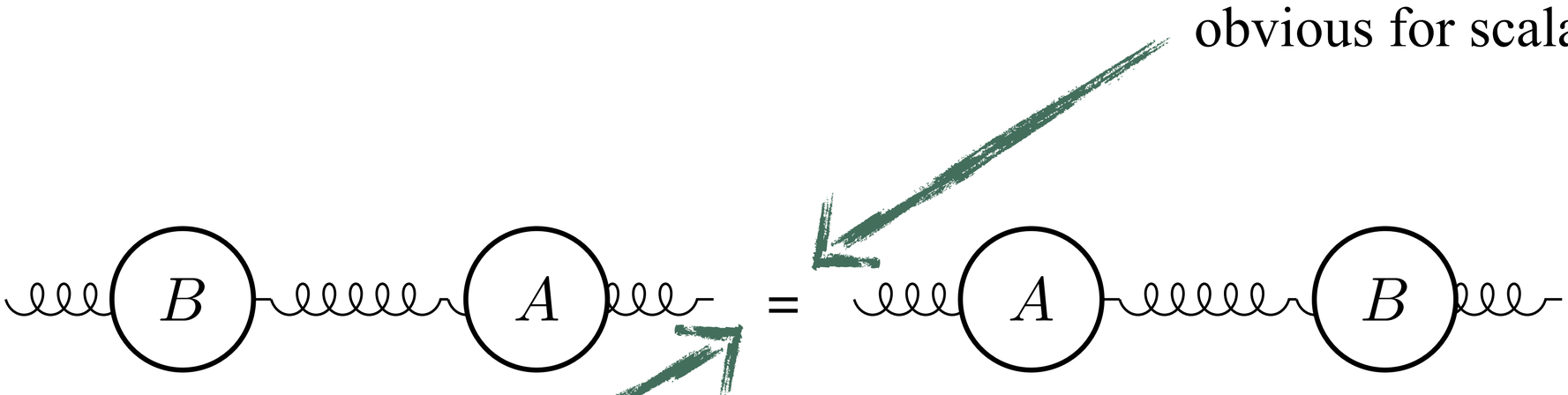
contract bridges



identify equivalent propagators



Ex. 5. Moving Self-Energies



Shuffling self-energies exposes graph automorphisms.

Extra

Ex. 6. Iterated Integrals

integration only as series expansion

$$F_{i_1,\ldots,i_n}(x) = \int_0^x \mathrm{d}t\ \omega_n(t)\ F_{i_1,\ldots,i_{n-1}}(t)$$

often poor convergence

Ex. 6. Iterated Integrals

very naive Mathematica implementation

```
In[1]:= w1 = 1 / x;

In[2]:= w2 = Series[1 / Sqrt[x (4 + x)], {x, 0, 400}] // Normal;

In[3]:= seriesInt[expr_, kernel_] := Normal@Series[Integrate[expr * kernel, x], {x, 0, 400}]

In[4]:= AbsoluteTiming[
          Int[] = 1;
          Int[1] = seriesInt[Int[], w1];
          Int[1, 2] = seriesInt[Int[1], w2];
          Int[1, 2, 2] = seriesInt[Int[1, 2], w2];
         ] // First

Out[4]= 26.509
```

quite slow (*but I am not good at Mathematica*)

Ex. 6. Iterated Integrals

$$\int \mathrm{d}x\, [\log(x)]^m\, x^s$$

integrals of this type are well-known

```
id ifmatch->SKIP pow(x,-1)*pow(log(x),n?) = 1/(n+1)*pow(log(x),n+1);
id ifmatch->SKIP pow(log(x),n1?)*pow(x,n2?) = (pow(x,n2+1)/(n2+1)
      *sum_(n,0,n1,sign_(n)*fac_(n1)/fac_(n1-n)*pow(log(x),n1-n)/(n2+1)^n)
  );
id ifmatch->SKIP pow(x,-1) = pow(log(x),1);
id ifmatch->SKIP pow(x,n?!{,-1}) = 1/(n+1)*pow(x,n+1);
Label SKIP;
```

Ex. 6. Iterated Integrals

74 integrals (up to weight 5)

```
Repeat;
  if(match(F(?a)));
    id F(?a,x?) = F(?a)*W(x);
    Multiply replace_(F,TBLF);
    Multiply replace_(TBLF,F);
    id F() = 1;
  endif;
EndRepeat;
ChainIn W;
id W(?a) = F(?a);
```

```
WTime =       3.37 sec    Generated terms =          1
       EXPANSION          Terms in output =          1
                          Bytes used      =         84
```

tabulate computed integrated integrals

use the differential equation

$$\mathrm{d}F_{i_1,\ldots,i_n}(x) = \omega_{i_n}(x)\, F_{i_1,\ldots,i_{n-1}}(x)$$