Pensieve header: The Engine, with Zip3 encapsulation.

Canonical Forms:

```
CCF[ℰ_] := PP_CCF@Factor[ℰ]; (*Coefficient Canonical Form *)
CF[ℰ_] := PP_CF@Module[
    {vs = Cases[ℰ, (y | a | x | η | β | τ | ξ)_, ∞] ⋃ {y, a, x, η, β, τ, ξ}},
    Total[(CCF[#⟦2⟧] (Times @@ vs^#⟦1⟧)) & /@ CoefficientRules[ℰ, vs]]
   ];
CF[ℰ_𝔼] := CF /@ ℰ;
CF[ℰ_List] := CF /@ ℰ;
CF[𝔼_sp___[ℰs___]] := CF /@ 𝔼_sp[ℰs];
```

Variables and their duals:

```
In[∘]:= {t*, b*, y*, a*, x*, z*, τ*, β*, η*, α*, ξ*, ζ*} = {τ, β, η, α, ξ, ζ, t, b, y, a, x, z};
(vs_List)* := (v ↦ v*) /@ vs;
(u_i_)* := (u*)_i;
𝓕[u_i_] := 𝓕[u_i] = ToExpression["𝓕" <> ToString[u]]_i
```

Weights:

```
Clear[Wt];
Evaluate[Wt /@ {y, b, t, a, x, η, β, τ, α, ξ}] = {1, 0, 0, 2, 1, 1, 2, 2, 0, 1};
Wt[u_i_] := Wt[u];
```

The maximal weight \$n, i.e. the *n* of gl(*n*). Initially and for a long while this will not be tested beyond \$n == 2.

```
In[∘]:= $n = 2;
```

Upper to lower and lower to Upper:

```
U2l[ℰ_] := ℰ /. {B_i_^p_. :→ ⅇ^(-p ℏ b_i), B^p_. :→ ⅇ^(-p ℏ b), T_i_^p_. :→ ⅇ^(p ℏ t_i), T^p_. :→ ⅇ^(p ℏ t), 𝒜_i_^p_. :→ ⅇ^(p α_i), 𝒜^p_. :→ ⅇ^(p α)};
l2U[ℰ_] := ℰ //. {ⅇ^(c_. b_i_ + d_.) :→ B_i^(-c/ℏ) ⅇ^d, ⅇ^(c_. b + d_.) :→ B^(-c/ℏ) ⅇ^d, ⅇ^(c_. t_i_ + d_.) :→ T_i^(c/ℏ) ⅇ^d, ⅇ^(c_. t + d_.) :→ T^(c/ℏ) ⅇ^d,
    ⅇ^(c_. α_i_ + d_.) :→ 𝒜_i^c ⅇ^d, ⅇ^(c_. α + d_.) :→ 𝒜^c ⅇ^d, ⅇ^𝒳_ :→ ⅇ^(Expand@𝒳)};
l2U[r_Rule] := Module[{U = r⟦1⟧ /. {b → B, t → T, α → 𝒜}}, U → l2U[U2l[U] /. r]];
AlsoUpper[rs_List] := rs ⋃ (l2U /@ rs);
```

Derivatives in the presence of exponentiated variables:

```
In[∘]:= D_b[f_] := ∂_b f - ℏ B ∂_B f; D_b_i_[f_] := ∂_b_i f - ℏ B_i ∂_B_i f;
D_t[f_] := ∂_t f + ℏ T ∂_T f; D_t_i_[f_] := ∂_t_i f + ℏ T_i ∂_T_i f;
D_α[f_] := ∂_α f + 𝒜 ∂_𝒜 f; D_α_i_[f_] := ∂_α_i f + 𝒜_i ∂_𝒜_i f;
D_v_[f_] := ∂_v f;
```

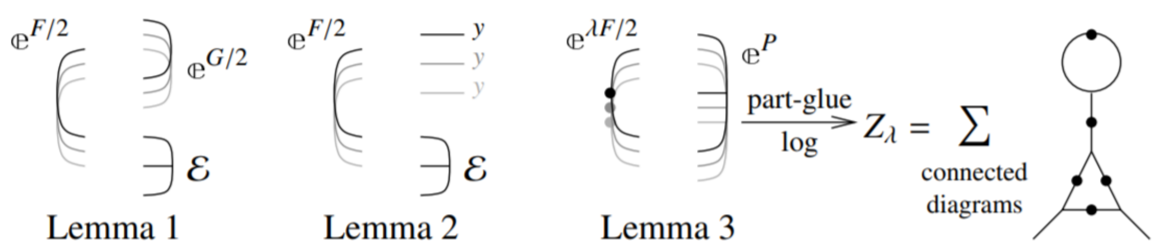
𝔼 operations:

```
ℰ_𝔼[$] := Length[ℰ] - 1; 𝔼_[ℰs___][$] := 𝔼[ℰs][$];
ℰ_𝔼[k_Integer] := ℰ⟦k + 1⟧; 𝔼_[ℰs___][k_Integer] := {ℰs}⟦k + 1⟧;
𝔼 /: ℰ1_𝔼 ≡ ℰ2_𝔼 := Inner[CF@#1 == CF@#2 &, ℰ1, ℰ2, And];
𝔼d1_→r1_[ℰ1s___] ≡ 𝔼d2_→r2_[ℰ2s___] ^:= (d1 == d2) ∧ (r1 == r2) ∧ (𝔼[ℰ1s] ≡ 𝔼[ℰ2s]);
𝔼 /: ℰ1_𝔼 * ℰ2_𝔼 := 𝔼 @@ Table[CF[ℰ1[kk] + ℰ2[kk]], {kk, 0, Min[ℰ1[$], ℰ2[$]]}];
𝔼d1_→r1_[ℰ1s___] 𝔼d2_→r2_[ℰ2s___] ^:= 𝔼(d1⋃d2)→(r1⋃r2) @@ (𝔼[ℰ1s] 𝔼[ℰ2s]);
```

```
𝔼d1_→r1_[ℰ1s___] // 𝔼d2_→r2_[ℰ2s___] := Module[{is = r1 ⋂ d2, lvs},
  lvs = Flatten@Table[{y$@i, b$@i, t$@i, a$@i, x$@i}, {i, is}];
  𝔼(d1⋃Complement[d2,is])→(r2⋃Complement[r1,is]) @@ (Zip_{lvs⋃lvs*}[{(𝓕 /@ lvs*).(𝓕 /@ lvs), Times[
      𝔼[ℰ1s] /. Table[(v : b | B | t | T | a | x | y)_i → v$@i, {i, is}],
      𝔼[ℰ2s] /. Table[(v : β | τ | α | 𝒜 | ξ | η)_i → v$@i, {i, is}]
     ]}])
 ]
```

```
Λ2𝔼d_→r_[Λ_] := Module[{k}, 𝔼d→r @@ l2U@Table[SeriesCoefficient[Λ, {ϵ, 0, k}], {k, 0, $k}]];
```

Zipping! Lemmas 2 and 3 are combined, yet they must be applied first to the middle weight variables and then to the heavy and light variables.



Comment. Zip3 of the outer variables must occur after all other operations are completed, because we must allow for gluings of the weight n variables in perturbations with the weight 0 variables in the coefficients of Q.

```
Zip_vs_[{𝓕_, ℰ_}] :=
  {𝓕, ℰ} // Zip1_vs (*//Zip2_Select[vs, (0<Wt[#]<$n)&]*) // Zip2_Select[vs, (Wt[#]==0∨Wt[#]==$n)&] //
    EZip3_Select[vs, (0<Wt[#]<$n)&] // Zip3_Select[vs, (Wt[#]==0∨Wt[#]==$n)&] // Last;
```

Getting rid of the quadratic.

Lemma 1. With convergences left to the reader,

$$\left\langle F:\mathcal{E}\,e^{\frac{1}{2}\sum_{i,j\in B}G_{ij}z_iz_j}\right\rangle_B=\det(1-GF)^{-1/2}\left\langle F(1-GF)^{-1}:\mathcal{E}\right\rangle_B$$

```
In[∘]:=
Zip1{} = Identity;
Zip1_vs_@{𝓕_, 𝔼[Q_, P___]} := PP_Zip1@Module[{𝓘, F, G, u, v},
   𝓘 = IdentityMatrix@Length@vs;
   F = Table[If[Wt[u] + Wt[v] == $n, ∂_{𝓕[u],𝓕[v]} 𝓕, 0], {u, vs}, {v, vs}];
   G = Table[If[Wt[u] + Wt[v] == $n, ∂_{u,v} Q, 0], {u, vs}, {v, vs}];
   {CF[(𝓕 /@ vs).(F.Inverse[𝓘 - G.F]).(𝓕 /@ vs) / 2],
    𝔼[CF[Q - PowerExpand@Log[Det[𝓘 - G.F]] / 2 - vs.G.vs / 2], P]}
  ]
```

Getting rid of linear terms.

Lemma 2. $\left\langle F:\mathcal{E}\,e^{\sum_{i\in B}y_iz_i}\right\rangle_B=e^{\frac{1}{2}\sum_{i,j\in B}F_{ij}y_iy_j}\left\langle F:\ \mathcal{E}|_{z_B\to z_B+Fy_B}\right\rangle_B.$

In[∘]:=
```
Zip2{} = Identity;
Zip2vs_@{𝓕_, 𝔼[Q_, P___]} := PPZip2@Module[{F, Y, u, v},
   F = Table[If[Wt[u] + Wt[v] == $n, CF[∂F[u],F[v] 𝓕], 0], {u, vs}, {v, vs}];
   Y = Table[∂v Q, {v, vs}] /. AlsoUpper@Table[v → 0, {v, vs}];
   CF /@ ({𝓕, 𝔼[Q - Y.vs + Y.F.Y / 2, P]} /. AlsoUpper@Thread[vs → vs + F.Y])
  ]
```

Dealing with Feynman diagrams.

Lemma 3. With an extra variable λ, $Z_\lambda := \log[\lambda F : \mathbb{e}^P]_B$ satisfies and is determined by the following PDE / IVP:

$$Z_0 = P \quad \text{and} \quad \partial_\lambda Z_\lambda = \frac{1}{2} \sum_{i,j \in B} F_{ij} \left(\partial_{z_i} \partial_{z_j} Z_\lambda + (\partial_{z_i} Z_\lambda)(\partial_{z_j} Z_\lambda) \right).$$

Note that the power m of λ is at most $k - 1 + \frac{2k+2}{2} = 2k$. We write $Z_\lambda = \sum Z[m]\,\lambda^m$.

```
Zip3vs_@{𝓕_, ℰ_𝔼} := PPZip3@Module[{F, u, v, Z, $k, kk, jj, $m = 0, m, n},
   $k = Length[ℰ] - 1;
   Do[Z[0, kk] = ℰ⟦kk + 1⟧, {kk, 0, $k}];
   F[u_, v_] := F[u, v] = CF@If[Wt[u] + Wt[v] == $n, ∂F[u],F[v] 𝓕, 0];
   Z[m_, kk_, u_] := Z[m, kk, u] = Du[Z[m, kk]];
   Z[m_, kk_, u_, v_] := Z[m, kk, u, v] = Dv[Z[m, kk, u]];
   For[m = 0, m ≤ 2 $m, ++m, For[kk = 0, kk ≤ $k, ++kk,
     Z[m + 1, kk] = CF@Sum[
        If[F[u, v] === 0, 0, F[u, v] / (2 (m + 1))
          (Z[m, kk, u, v] + Sum[Z[n, jj, u] * Z[m - n, kk - jj, v], {n, 0, m}, {jj, 0, kk}])],
        {u, vs}, {v, vs}];
     If[Z[m + 1, kk] =!= 0, $m = m + 1]
    ]];
   CF /@ ({
       𝓕 - Sum[F[u, v] * F[u] * F[v] / 2, {u, vs}, {v, vs}],
       𝔼 @@ Table[Sum[Z[m, kk], {m, 0, $m}], {kk, 0, $k}]
      } /. AlsoUpper@Table[v → 0, {v, vs}])
  ]
```

Encapsulation.

```
EZip3vs_@{ℱ_, ℰ_𝔼} := PPEZip3@Module[
  {nℰ, nℱ, rc, ps, rr = {(*release rules*)}, Fvs},
  rc = 0; nℰ = Total[
     CoefficientRules[#, vs] /. (ps_ → c_) :→ (AppendTo[rr, cℰ[++rc] → c]; cℰ[rc] (Times @@ vs^ps))
    ] & /@ ℰ;
  rc = 0; Fvs = F /@ vs;
  nℱ = Total[CoefficientRules[ℱ, Fvs] /.
     (ps_ → c_) :→ (AppendTo[rr, cℱ[++rc] → c]; cℱ[rc] (Times @@ Fvs^ps))];
  CF[Expand[{nℱ, nℰ} // Zip3vs] /. rr]
 ]
```