The Engine

Canonical

```
CCF[ℰ_] := PP_CCF@Factor[ℰ];
(*Coefficient Canonical Form *)
LogReduce[ℰ_] :=
  ℰ /. c_ * Log[a_] :> Log@Factor[a^c] //.
   Log[a_] + Log[b_] :> Log@Factor[a b];
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 /@ MapAt[LogReduce, ℰ, 1];
CF[U_𝕎] := CF /@ MapAt[LogReduce, U, 1];
CF[ℰ_List] := CF /@ ℰ;
CF[𝔼_sp___[ℰs___]] := CF /@ 𝔼_sp[ℰs];
CF[𝕎_sp___[Us___]] := CF /@ 𝕎_sp[Us];
```

Variables and their duals:

```
{t*, b*, y*, a*, x*, z*, τ*, β*, η*, α*, ξ*, ξ*} =
  {τ, β, η, α, ξ, ξ, 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.

```
$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:

```
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;
```

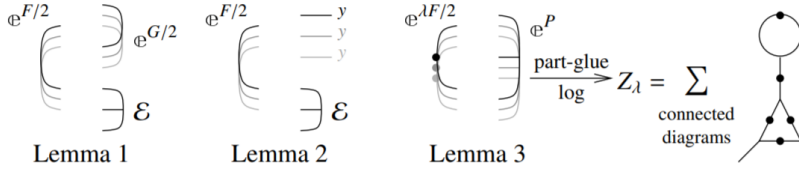
𝔼

```
ℰ_𝔼[$] := Length[ℰ] - 1; 𝔼_[ℰs___][$] := 𝔼[ℰs][$];
ℰ_𝔼[k_Integer] := ℰ[[k + 1]];
𝔼_[ℰs___][k_Integer] := {ℰs}[[k + 1]];
𝔼 /: ℰ1_𝔼 ≡ ℰ2_𝔼 :=
  Inner[CF@#1 == CF@#2 &, ℰ1, ℰ2, And];
𝔼_d1_→r1_[ℰ1s___] ≡ 𝔼_d2_→r2_[ℰ2s___] ^:=
  (d1 == d2) ∧ (r1 == r2) ∧ (𝔼[ℰ1s] ≡ 𝔼[ℰ2s]);
𝔼 /: ℰ1_𝔼 * ℰ2_𝔼 :=
  𝔼 @@ Table[CF[ℰ1[kk] + ℰ2[kk]],
    {kk, 0, Min[ℰ1[$], ℰ2[$]]}];
𝔼_d1_→r1_[ℰ1s___] 𝔼_d2_→r2_[ℰ2s___] ^:=
  𝔼_(d1∪d2)→(r1∪r2) @@ (𝔼[ℰ1s] × 𝔼[ℰ2s]);
```

```
𝔼_d1_→r1_[ℰ1s___] // 𝔼_d2_→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}]
      ]}])
 ]
```

```
Λ2𝔼_d_→r_[Λ_] :=
  Module[{k},
   𝔼_d→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,

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

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

```
Zip2_{} = Identity;
Zip2_vs_@{ℱ_, 𝔼[Q_, P___]} :=
 PP_Zip2@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:

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

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

```
Zip3_vs_@{ℱ_, ℰ_, 𝔼} :=
 PP_Zip3@
  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] = D_u[Z[m, kk]];
   Z[m_, kk_, u_, v_] :=
    Z[m, kk, u, v] = D_v[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.

```
EZip3_vs_@{F_, E_E} := PP_EZip3@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} // Zip3_vs] /. rr]
 ]
```

Exponentials and logarithms as in Exp.nb

```
Exp_m_[U : U_is_→{i1_}[__]] :=
 Module[{λ, μ, k, n, F, f, i, j, lhs, rhs, U1, MI
   (*multi-index*), mis, mi, yax},
  MI /: Coefficient[E_, MI[p_, n_, q_]] :=
   Coefficient[Coefficient[Coefficient[E, y_i, p],
     a_i, n], x_i, q];
  yax /: yax^MI[p_,n_,q_] := y_i^p a_i^n x_i^q;
  U1 = U /. (v : (y | b | t | a | x | B | T | R))_i1 → v_i;
  F = E_is→{i}[];
  Do[AppendTo[F, 0]; Do[
    mis = Flatten@Table[MI[p, n, q],
        {p, 0, Min[k + 1, 2 k + 2 - 2 n]},
        {q, 0, Min[k + 1, 2 k + 2 - 2 n - p]}];
    F[[-1]] += Sum[f_mi[λ] yax^mi, {mi, mis}];
    lhs =
     (∂_μ U2l@Last[F (F /. {λ → μ, i → j}) //
              m_i,j→i]) /. μ → 0 /. f_[0] → 0 /.
      Table[f_mi'[0] → Coefficient[U1[[k + 1]], mi],
       {mi, mis}];
    rhs = ∂_λ U2l@Last[F];
    F =
     l2U[
      F /.
       First@
        DSolve[Table[Coefficient[lhs - rhs, mi] == 0 ∧
           f_mi[0] == 0, {mi, mis}],
         Table[f_mi, {mi, mis}], λ]],
    {n, k + 1, 0, -1}], {k, 0, Length[U1] - 1}];
  CF@l2U[F /. {λ → 1, i → i1}] ]
```

Logarithms

Task. Define $\mathrm{Exp}_m[U : \mathbb{U}_{\{\_\}\to\{i\_\}}[\_\_]]$ to compute $e^{\mathbb{D}(U)}$ to order $\epsilon^{\mathrm{Length@}\{U\}-1}$ using the $m_{i,i\to i}$ multiplication, where U is an ϵ-dependent sub-balanced near-docile element, giving the answer in $\mathbb{E}$-form.

Example: $\mathrm{Exp}_{\mathrm{dm},1}[\mathbb{U}_{\{\}\to\{2\}}[b_2\, a_2 + y_2\, x_2,\, 0]]$ is the exponential of the arrow on strand 2, computed to degree 1.

Task. Define $\mathrm{Log}_m[\mathcal{E} : \mathbb{E}_{\{\_\}\to\{i\_\}}[\_\_]]$ to compute $\mathrm{Log@}\mathbb{D}[e^{\mathcal{E}}]$ to order $\epsilon^{\mathrm{Length@}\{U\}-1}$ using the $m_{i,i\to i}$ multiplication, where $\mathcal{E}$ is an ϵ-dependent sub-balanced docile element, giving the answer in $\mathbb{U}$-form.

```
Log_m_[ℰ:𝔼_is_→{i_}[__]] :=
 Module[{e, k, n, G, c, g, eqn, Sanify, MI
   (*multi-index*), mis, mi, yax, p, q},
  G = 𝕌_is→{i}[c1 a_i + c2 x_i y_i];
  eqn = U2l[Last[Exp_m[G]] - ℰ[[1]]];
  {eqn, G} =
   CF /@
    ({eqn, G} /.
      First@Solve[Coefficient[eqn, a_i] == 0, c1]);
  Sanify[{{v_ → s_}}] :=
   v → PowerExpand[Normal[s] /. c_ → 0];
  G =
   CF[
    G /. Sanify@Solve[Coefficient[eqn, x_i y_i] == 0,
       c2]];
  G[[1]] += c0 + c1 x_i + c2 y_i;
  eqn = U2l[Last[Exp_m[G]] - ℰ[[1]]];
  {eqn, G} =
   CF /@
    ({eqn, G} /.
      First@Solve[Coefficient[eqn, x_i] == 0 ∧
          Coefficient[eqn, y_i] == 0, {c1, c2}]);
  G = G /. First@Solve[eqn == 0, c0];
  MI /: Coefficient[e_, MI[p_, n_, q_]] :=
   Coefficient[Coefficient[Coefficient[e, y_i, p],
     a_i, n], x_i, q];
  yax /: yax^MI[p_,n_,q_] := y_i^p a_i^n x_i^q;
  Do[
   mis = Flatten@Table[MI[p, n, q], {n, 0, k + 1},
       {p, 0, Min[k + 1, 2 k + 2 - 2 n]},
       {q, 0, Min[k + 1, 2 k + 2 - 2 n - p]}];
   AppendTo[G, Sum[g_mi yax^mi, {mi, mis}]];
   eqn = U2l[Last[Exp_m[G]] - ℰ[[k + 1]]];
   G =
    CF[
     G /.
      First@Solve[Table[Coefficient[eqn, mi] == 0,
         {mi, mis}], Table[g_mi, {mi, mis}]]],
   {k, Length[ℰ] - 1}];
  CF[l2U@G]
 ]
```

The Objects

"Define"

```
SetAttributes[Define, HoldAll];
Define[def_, defs__] :=
  (Define[def]; Define[defs];);
Define[op_is__ = ℰ_] :=
 Module[{SD, ii, jj, kk, isp, nis, nisp, sis},
  Block[{i, j, k},
   ReleaseHold[Hold[
      SD[op_nisp,$k_Integer,
       PP_Boot@Block[{i, j, k}, op_isp,$k = ℰ; op_nis,$k]];
      SD[op_isp, op_{is},$k]; SD[op_sis__, op_{sis}];
     ] /. {SD → SetDelayed,
      isp → {is} /. {i → i_, j → j_, k → k_},
      nis → {is} /. {i → ii, j → jj, k → kk},
      nisp → {is} /. {i → ii_, j → jj_, k → kk_}
     }] ]]
```

Symmetric Algebra Objects

```
sm_i_,j_→k_ :=
  Λ2𝔼_{i,j}→{k}[b_k (β_i + β_j) + t_k (τ_i + τ_j) + a_k (α_i + α_j) +
    y_k (η_i + η_j) + x_k (ξ_i + ξ_j)];
sΔ_i_→j_,k_ :=
  Λ2𝔼_{i}→{j,k}[β_i (b_j + b_k) + τ_i (t_j + t_k) + α_i (a_j + a_k) +
    η_i (y_j + y_k) + ξ_i (x_j + x_k)];
sS_i_ := Λ2𝔼_{i}→{i}[-β_i b_i - τ_i t_i - α_i a_i - η_i y_i - ξ_i x_i];
sη_i_ := Λ2𝔼_{}→{i}[0];
sϵ_i_ := Λ2𝔼_{i}→{}[0];
```

```
sσ_i_→j_ := Λ2𝔼_{i}→{j}[β_i b_j + τ_i t_j + α_i a_j + η_i y_j + ξ_i x_j];
sϒ_i_→j_,k_,l_,m_ :=
  Λ2𝔼_{i}→{j,k,l,m}[β_i b_k + τ_i t_k + α_i a_l + η_i y_j + ξ_i x_m];
```

The CU Definitions

$$c\Lambda = \left(\eta_i + \frac{e^{-\alpha_i - \epsilon\,\beta_i}\,\eta_j}{1 + \epsilon\,\eta_j\,\xi_i}\right) y_k + \left(\beta_i + \beta_j + \frac{\text{Log}[1 + \epsilon\,\eta_j\,\xi_i]}{\epsilon}\right) b_k + (\alpha_i + \alpha_j + \text{Log}[1 + \epsilon\,\eta_j\,\xi_i])\,a_k + \left(\frac{e^{-\alpha_j - \epsilon\,\beta_j}\,\xi_i}{1 + \epsilon\,\eta_j\,\xi_i} + \xi_j\right) x_k;$$

```
Define[
 cm_i,j→k =
  Λ2𝔼_{i,j}→{k}[(η_i + e^(-α_i - ϵ β_i) η_j / (1 + ϵ η_j ξ_i)) y_k +
   (β_i + β_j + Log[1 + ϵ η_j ξ_i] / ϵ) b_k +
   (α_i + α_j + Log[1 + ϵ η_j ξ_i]) a_k +
   (e^(-α_j - ϵ β_j) ξ_i / (1 + ϵ η_j ξ_i) + ξ_j) x_k]]
```

```
Define[cσ_{i→j} = sσ_{i,j} /. τ_i → 0, cϵ_i = sϵ_i, cη_i = sη_i,
  cΔ_{i→j,k} = sΔ_{i→j,k},
  cS_i = sS_i // sϒ_{i→1,2,3,4} // cm_{4,3→i} // cm_{i,2→i} // cm_{i,1→i}];
```

Booting Up QU

```
Define[aσ_{i→j} = Λ2𝔼_{{i}→{j}}[a_j α_i + x_j ξ_i],
 bσ_{i→j} = Λ2𝔼_{{i}→{j}}[b_j β_i + y_j η_i]]
```

```
Define[
 am_{i,j→k} = Λ2𝔼_{{i,j}→{k}}[(α_i + α_j) a_k + (𝒜_j^{-1} ξ_i + ξ_j) x_k],
 bm_{i,j→k} = Λ2𝔼_{{i,j}→{k}}[(β_i + β_j) b_k + (η_i + e^{-ϵ β_i} η_j) y_k]]
```

```
Define[
 R_{i,j} = Module[{k},
   Λ2𝔼_{{}→{i,j}}[ħ a_j b_i + Sum[((1 - e^{ϵ ħ})^k (ħ y_i x_j)^k)/(k (1 - e^{k ϵ ħ})), {k, 1, $k+1}]]]]
```

Three types of inverses appear below!
$\overline{R}$ is the inverse of R in the algebra $\mathbb{B}\otimes\mathbb{A}$.
P is the inverse of R as a quadratic form, like how an element of $V^*\otimes V^*$ can be the inverse of an element of $V\otimes V$.
$\overline{aS}$ is the inverse of aS as an operator form, like how an element of $V^*\otimes V$ can be the inverse of another element of $V^*\otimes V$.

```
Define[
 R̄_{i,j} = If[$k == 0, 𝔼_{{}→{i,j}}[-ħ a_j b_i - ħ x_j y_i / B_i],
   Append[R̄_{{i,j},$k-1},
    -Last[
      PadRight[R̄_{{i,j},0}, $k + 1] R_{1,2}
          PadRight[R̄_{{3,4},$k-1}, $k + 1] //
        (bm_{i,1→i} am_{j,2→j}) // (bm_{i,3→i} am_{j,4→j})]]]
]
```

```
Define[
 P_{i,j} = If[$k == 0, 𝔼_{{i,j}→{}}[β_i α_j / ħ + η_i ξ_j / ħ],
   Append[P_{{i,j},$k-1},
    -Last[
      R_{1,2} // (PadRight[P_{{1,j},0}, $k + 1] *
          PadRight[P_{{i,2},$k-1}, $k + 1])]]]]
```

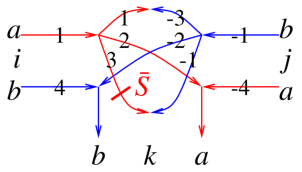
```
Define[aS_i = (aσ_{i→2} R̄_{1,i}) // P_{1,2}]
```

```
Define[āS_i = If[$k == 0, 𝔼_{{i}→{i}}[-a_i α_i - x_i 𝒜_i ξ_i],
   Append[āS_{{i},$k-1},
    -Last[PadRight[āS_{{i},0}, $k + 1] // aS_i //
       PadRight[āS_{{i},$k-1}, $k + 1]]]
 ]]
```

Booting Up QU

```
Define[bS_i = bσ_{i→1} R_{i,2} // aS_2 // P_{1,2},
 b̄S_i = bσ_{i→1} R_{i,2} // āS_2 // P_{1,2},
 aΔ_{i→j,k} = (R_{1,j} R_{2,k}) // bm_{1,2→3} // P_{3,i},
 bΔ_{i→j,k} = (R_{j,1} R_{k,2}) // am_{1,2→3} // P_{i,3}]
```

The Drinfel'd double:

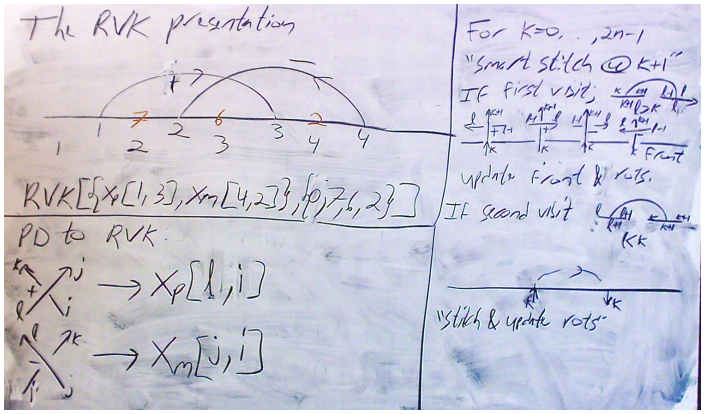



```
Define[
 dm_{i,j→k} =
  ((sϒ_{i→4,4,1,1} // aΔ_{1→1,2} // aΔ_{2→2,3} // āS_3)
     (sϒ_{j→-1,-1,-4,-4} // bΔ_{-1→-1,-2} // bΔ_{-2→-2,-3})) //
   (P_{-1,3} P_{-3,1} am_{2,-4→k} bm_{4,-2→k})]
```

```
Define[dσ_{i→j} = aσ_{i→j} bσ_{i→j},
 dϵ_i = sϵ_i, dη_i = sη_i,
 dS_i = sϒ_{i→1,1,2,2} // (b̄S_1 aS_2) // dm_{2,1→i},
 d̄S_i = sϒ_{i→1,1,2,2} // (bS_1 āS_2) // dm_{2,1→i},
 dΔ_{i→j,k} = (bΔ_{i→3,1} aΔ_{i→2,4}) // (dm_{3,4→k} dm_{1,2→j})]
```

```
Define[C_i = Λ2𝔼_{{}→{i}}[-ħ/2 (b_i + ϵ a_i)],
 C̄_i = Λ2𝔼_{{}→{i}}[ħ/2 (b_i + ϵ a_i)],
 Kink_i = (R_{1,3} C̄_2) // dm_{1,2→1} // dm_{1,3→i},
 K̄ink_i = (R̄_{1,3} C_2) // dm_{1,2→1} // dm_{1,3→i}]
```

RVK and Z




```
RVK::usage =
  "RVK[xs, rots] represents a Rotational
    Virtual Knot with a list of n Xp/Xm
    crossings xs and a length 2n list of
    rotation numbers rots. Crossing sites
    are indexed 1 through 2n, and rots[[k]]
    is the rotation between site k-1 and
    site k. RVK is also a casting operator
    converting to the RVK presentation from
    other knot presentations.";
```

```
RVK[pd_PD] :=
  PP_RVK@Module[{n, xs, x, rots, front = {0}, k},
    n = Length@pd; rots = Table[0, {2 n}];
    xs = Cases[pd,
      x_X :> {Xp[x[[4]], x[[1]]]  PositiveQ@x
              Xm[x[[2]], x[[1]]]        True   ];
    For[k = 0, k < 2 n, ++k,
     If[k == 0 ∨ FreeQ[front, -k],
      front = Flatten@Replace[front, k → (xs /. {
              Xp[k + 1, l_] | Xm[l_, k + 1] :>
               {l, k + 1, 1 - l},
              Xp[l_, k + 1] | Xm[k + 1, l_] :>
               (++rots[[l]];
                {1 - l, k + 1, l}),
              _Xp | _Xm :> {}
             }), {1}],
      Cases[front, k | -k] /.
        {k, -k} :> --rots[[k + 1]];
     ]];
    RVK[xs, rots]];
RVK[K_] := RVK[PD[K]];
```

```
rot[i_, 0] := dη_i;
rot[i_, n_] := rot[i, n, $k];
rot[i_, n_, k_] := Module[{j},
   rot[i, n, k] =
    If[n > 0, rot[i, n - 1] kC_j, rot[i, n + 1] kC̄_j] //
     km_{i,j→i}];
```

```
Width[pd_PD] :=
 Max[
  Length /@ FoldList[Complement[#1 ⋃ #2, #1 ⋂ #2] &,
    {}, List @@ List @@@ pd]]
```

```
ThinPosition[K_] := Module[{todo, done, pd, c},
   todo = List @@ PD@K; done = {}; pd = PD[];
   While[todo =!= {},
    AppendTo[pd,
     c = RandomChoice@MaximalBy[todo,
         Length[done ⋂ List @@ #] &]];
    todo = DeleteCases[todo, c];
    done = done ⋃ List @@ c];
   pd ];
ThinPosition[K_, n_] :=
  First@MinimalBy[Table[ThinPosition[K], n],
    Width];
```

```
Z[K_] :=
  Z[RVK@EchoFunction[Width]@ThinPosition[K, 100]];
Z[rvk_RVK] :=
 Monitor[ PP"Z"@Module[{ξ, done, st, c, χ, i, j, k},
    ξ = 1; done = {}; st = Range[2 Length[rvk[[1]]]];
    $M = {};
    Do[AppendTo[$M, c];
     {i, j} = List @@ c;
     χ =
      (c /. {_Xp :> kR_{i,j} k̄Kink̄_0, _Xm :> k̄R̄_{i,j} kKink_0}) //
       km_{j,0→j};
     Do[χ = (rot[0, rvk[[2, k]]] χ) // km_{0,k→k},
      {k, {i, j}}];
     ξ *= χ;
     Do[
      If[MemberQ[done, k + 1], ξ = ξ // km_{k,k+1→k};
       st = st /. k + 1 → k];
      If[MemberQ[done, k - 1],
       ξ = ξ // km_{st[[k-1]],k→st[[k-1]]};
       st = st /. k → st[[k - 1]],
      {k, {i, j}}];
     done = done ⋃ {i, j},
     {c, rvk[[1]]}
    ];
    CF /@ (ξ /. {x_1 → x, y_1 → y, a_1 → a})
   ], {Length@$M, $M}]
```