Pensieve header: The full sl_2 invariant using the Drinfel'd double. Continues 2018-05/ybax.nb, Talks/StonyBrook-1805/ybax.nb, Projects/SL2Portfolio/Logoi.nb.

Profiling

In[•]:=
```
Once[
  SetDirectory["C:\\drorbn\\AcademicPensieve\\Projects\\SL2Invariant"];
  << KnotTheory`;
  << "../Profile/Profile.m";
 ];
Once@PopupWindow[Button["Show Profile Monitor"],
  Dynamic[PrintProfile[], UpdateInterval → 3, TrackedSymbols → {}]]
```

External Utilities

In[•]:=
```
HL[ℰ_] := Style[ℰ, Background → Yellow];
```

Program

Program

Internal Utilities

Program

Canonical Form:

Program

In[•]:=
```
CF[sd_SeriesData] := MapAt[CF, sd, 3];
CF[ℰ_] := PP_CF@ExpandDenominator@ExpandNumerator@Together[
      Expand[ℰ] //. e^x_ e^y_ :↦ e^(x+y) /. e^x_ :↦ e^CF[x]];
```

Program

The Kronecker δ:

Program

In[•]:=
```
Kδ /: Kδ_i_,j_ := If[i === j, 1, 0];
```

Program

Equality, multiplication, and degree-adjustment of perturbed Gaussians; $\mathbb{E}[L, Q, P]$ stands for $e^{L+Q}P$:

Program

In[•]:=
```
𝔼 /: 𝔼[L1_, Q1_, P1_] ≡ 𝔼[L2_, Q2_, P2_] :=
  CF[L1 == L2] ∧ CF[Q1 == Q2] ∧ CF[Normal[P1 - P2] == 0];
𝔼 /: 𝔼[L1_, Q1_, P1_] 𝔼[L2_, Q2_, P2_] := 𝔼[L1 + L2, Q1 + Q2, P1 * P2];
𝔼[L_, Q_, P_]$k_ := 𝔼[L, Q, Series[Normal@P, {ϵ, 0, $k}]];
```

Program

```
In[•]:= 𝔼 /: 𝔼[Q1_, P1_] ≡ 𝔼[Q2_, P2_] := CF[Q1 == Q2] ∧ CF[Normal[P1 - P2] == 0];
        𝔼 /: 𝔼[Q1_, P1_] 𝔼[Q2_, P2_] := 𝔼[Q1 + Q2, P1 * P2];
        𝔼[Q_, P_]$k_ := 𝔼[Q, Series[Normal@P, {ϵ, 0, $k}]];
```

Program

Zip and Bind

Program

Variables and their duals:

Program

```
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};  (u_i_)* := (u*)i;
```

Program

Finite Zips:

Program

```
In[•]:= collect[sd_SeriesData, ζ_] := MapAt[collect[#, ζ] &, sd, 3];
        collect[ℰ_, ζ_] := PPCollect@Collect[ℰ, ζ];
        Zip{}[P_] := P; Zip{ζ_,ζs___}[P_] := PPZip[
          (collect[P // Zip{ζs}, ζ] /. f_. ζ^d_. :↦ ∂{ζ*,d} f) /. ζ* → 0]
```

Program

QZip implements the "Q-level zips" on $\mathbb{E}(L, Q, P) = P e^{L+Q}$. Such zips regard the L variables as scalars.

Program

```
In[•]:= QZipζs_List,simp_@𝔼[L_, Q_, P_] := PPQZip@Module[{ζ, z, zs, c, ys, ηs, qt, zrule, Q1, Q2},
            zs = Table[ζ*, {ζ, ζs}];
            c = Q /. Alternatives@@(ζs ⋃ zs) → 0;
            ys = Table[∂ζ (Q /. Alternatives@@zs → 0), {ζ, ζs}];
            ηs = Table[∂z (Q /. Alternatives@@ζs → 0), {z, zs}];
            qt = Inverse@Table[Kδz,ζ* - ∂z,ζ Q, {ζ, ζs}, {z, zs}];
            zrule = Thread[zs → qt.(zs + ys)];
            Q2 = (Q1 = c + ηs.zs /. zrule) /. Alternatives@@zs → 0;
            simp /@ 𝔼[L, Q2, Det[qt] e^-Q2 Zipζs[e^Q1 (P /. zrule)]] ];
        QZipζs_List := QZipζs,CF;
```

Program

Upper to lower and lower to Upper:

Program

```
In[•]:= U2l = {B_i_^p_. → e^-p ħ γ b_i, B^p_. → e^-p ħ γ b, T_i_^p_. → e^p ħ t_i, T^p_. → e^p ħ t, 𝒜_i_^p_. → e^p γ α_i, 𝒜^p_. → e^p γ α};
        l2U = {e^c_. b_i_+d_. :↦ B_i^-c/(ħ γ) e^d, e^c_. b+d_. :↦ B^-c/(ħ γ) 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/γ e^d, e^c_. α+d_. :↦ 𝒜^c/γ e^d,
            e^ℰ_ :↦ e^Expand@ℰ};
```

Program

LZip implements the "L-level zips" on $\mathbb{E}(L, Q, P) = P e^{L+Q}$. Such zips regard all of $P e^{Q}$ as a single"P". Here the z's are b and α and the ζ's are β and a.

Program

In[ ]:=
```
Zip_{vs_List,is_List}@𝔼[Q_, P_] :=
  PP_𝔼Zip@Module[{v, i, ξs, ξ, z, zs, c, ys, ηs, qt, zrule, Q1, Q2},
    ξs = Flatten@Table[v_i, {v, vs}, {i, is}];
    zs = Table[ξ*, {ξ, ξs}];
    c = Q /. Alternatives @@ (ξs ⋃ zs) → 0;
    ys = Table[∂_ξ (Q /. Alternatives @@ zs → 0), {ξ, ξs}];
    ηs = Table[∂_z (Q /. Alternatives @@ ξs → 0), {z, zs}];
    qt = Inverse@Table[Kδ_{z,ξ*} - ∂_{z,ξ} Q, {ξ, ξs}, {z, zs}];
    zrule = Thread[zs → qt.(zs + ys)];
    Q2 = (Q1 = c + ηs.zs /. U2l /. zrule) /. Alternatives @@ zs → 0;
    CF /@ 𝔼[Q2, Det[qt] e^-Q2 Zip_ξs[e^Q1 (P /. U2l /. zrule)]] //. l2U];
```

Program

In[ ]:=
```
LZip_{ξs_List,simp_}@𝔼[L_, Q_, P_] :=
  PP_LZip@Module[{ξ, z, zs, c, ys, ηs, lt, zrule, L1, L2, Q1, Q2},
    zs = Table[ξ*, {ξ, ξs}];
    c = L /. Alternatives @@ (ξs ⋃ zs) → 0;
    ys = Table[∂_ξ (L /. Alternatives @@ zs → 0), {ξ, ξs}];
    ηs = Table[∂_z (L /. Alternatives @@ ξs → 0), {z, zs}];
    lt = Inverse@Table[Kδ_{z,ξ*} - ∂_{z,ξ} L, {ξ, ξs}, {z, zs}];
    zrule = Thread[zs → lt.(zs + ys)];
    L2 = (L1 = c + ηs.zs /. zrule) /. Alternatives @@ zs → 0;
    Q2 = (Q1 = Q /. U2l /. zrule) /. Alternatives @@ zs → 0;
    simp /@ 𝔼[L2, Q2, Det[lt] e^-L2-Q2 Zip_ξs[e^L1+Q1 (P /. U2l /. zrule)]] //. l2U];
LZip_ξs_List := LZip_{ξs,CF};
```

Program

In[ ]:=
```
Bind_{}[L_, R_] := L R;
Bind_{is__}[L_𝔼, R_𝔼] := PP_Bind@Module[{n, temp, out},
    temp = Times[
      L /. Table[(v : b | B | t | T | a | x | y)_i → v_{n@i}, {i, {is}}],
      R /. Table[(v : β | τ | α | 𝔄 | ξ | η)_i → v_{n@i}, {i, {is}}]
     ];
    out = temp // LZip_{Flatten@Table[{β_{n@i},τ_{n@i},a_{n@i}},{i,{is}}]} // QZip_{Flatten@Table[{ξ_{n@i},y_{n@i}},{i,{is}}]};
    Echo[(out /. 𝔼[L1_, Q_, P_] :↦ 𝔼[L1 + Q, P]) ≡
      ((temp /. 𝔼[L1_, Q_, P_] :↦ 𝔼[L1 + Q, P]) // Zip_{{β,τ,a},n/@{is}} // Zip_{{ξ,y},n/@{is}})];
    out
   ];
B_{l_List}[L_, R_] := Bind_l[L, R]; B_{is___}[L_, R_] := Bind_{is}[L, R];
```

Program

“Define” code

Program

Define[lhs = rhs, ...] defines the lhs to be rhs, except that rhs is computed only once for each value of $k. Fancy Mathematica not for the faint of heart. Most readers should ignore.

Program

In[•]:=
```
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,$k_Integer, PP_Boot@$k@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_}
     }] ]]
```

Program

Booting Up

Program

In[•]:=
```
$k = 2; ℏ = γ = 1;
```

Program

In[•]:=
```
Define[am_{i,j→k} = 𝔼[(α_i + α_j) a_k, (ⅇ^(-γ α_j) ξ_i + ξ_j) x_k, 1]_$k,
 bm_{i,j→k} = 𝔼[(β_i + β_j) b_k, (η_i + η_j) y_k, ⅇ^((ⅇ^(-ϵ β_i) - 1) η_j y_k)]_$k]
```

Program

In[•]:=
```
Define[R_{i,j} = 𝔼[ℏ a_j b_i, ℏ x_j y_i, ⅇ^(Sum[(1 - ⅇ^(γ ϵ ℏ))^k (ℏ y_i x_j)^k / (k (1 - ⅇ^(k γ ϵ ℏ))), {k, 2, $k+1}])]_$k,
 P_{i,j} = 𝔼[β_i α_j / ℏ, η_i ξ_j / ℏ,
   1 + If[$k == 0, 0, Normal@P_{i,j},$k-1[[3]] - (R_{1,2} ~ B_{1,2} ~ ((P_{1,j},0)$k (P_{i,2},$k-1)$k))[[3]]]]
```

Program

In[•]:=
```
Define[aS_i = 𝔼[-α_i a_j, -ξ_i x_i,
     Sum[Expand[ⅇ^(ξ_i x_i) (-ℏ γ ϵ)^k / (2^k k!) Nest[Expand[x_i^2 ∂_{x_i,2} #] &, ⅇ^(-ξ_i ⅇ^(ℏ ϵ a_i) x_i), k]], {k, 0, $k}]]_$k ~
   B_{i,j} ~ am_{i,j→i},
 aS̄_i = 𝔼[-a_i α_i, -x_i 𝒜_i ξ_i, 1 + If[$k == 0, 0,
     Normal@aS̄_{i},$k-1[[3]] - ((aS̄_{i},0)$k ~ B_i ~ aS_i ~ B_i ~ (aS̄_{i},$k-1)$k)[[3]]]]]
```

Program

In[•]:=
```
Define[bS_i = R_{i,1} ~ B_1 ~ aS_1 ~ B_1 ~ P_{i,1},
 bS̄_i = R_{i,1} ~ B_1 ~ aS̄_1 ~ B_1 ~ P_{i,1},
 aΔ_{i→j,k} = (R_{1,j} R_{2,k}) ~ B_{1,2} ~ bm_{1,2→3} ~ B_3 ~ P_{3,i},
 bΔ_{i→j,k} = (R_{j,1} R_{k,2}) ~ B_{1,2} ~ am_{1,2→3} ~ B_3 ~ P_{i,3}]
```

Program

```
In[•]:= Define[dm_{i,j→k} =
          (𝔼[β_i b_i + α_j a_j, η_i y_i + ξ_j x_j, 1] (aΔ_{i→1,2} ~ B_2 ~ aΔ_{2→2,3} ~ B_3 ~ a$S_3) (bΔ_{j→-1,-2} ~ B_{-2} ~ bΔ_{-2→-2,-3})) ~
           B_{-3,-2,-1,1,2,3,i,j} ~ (P_{-1,3} P_{-3,1} am_{2,j→k} bm_{i,-2→k}),
         dS_i = 𝔼[β_i b_1 + α_i a_2, η_i y_1 + ξ_i x_2, 1] ~ B_{1,2} ~ (b$S_1 aS_2) ~ B_{1,2} ~ dm_{2,1→i},
         dΔ_{i→j,k} = (bΔ_{i→3,1} aΔ_{i→2,4}) ~ B_{1,2,3,4} ~ (dm_{3,4→k} dm_{1,2→j})]
```

Program

```
In[•]:= Define[R$_{i,j} = Expand /@ R_{i,j} ~ B_j ~ dS_j,
         C_i = 𝔼[0, 0, B_i^{1/2} e^{-ħ ϵ a_i/2}]_{$k},
         C$_i = 𝔼[0, 0, B_i^{-1/2} e^{ħ ϵ a_i/2}]_{$k},
         Kink_i = (R_{1,3} C$_2) ~ B_{1,2} ~ dm_{1,2→1} ~ B_{1,3} ~ dm_{1,3→i},
         Kink$_i = (R$_{1,3} C_2) ~ B_{1,2} ~ dm_{1,2→1} ~ B_{1,3} ~ dm_{1,3→i}]
```

Program

Note. $t == \epsilon a - \gamma b$ and $b == -t/\gamma + \epsilon a/\gamma$.

Program

```
In[•]:= Define[b2t_i = 𝔼[α_i a_i - β_i t_i / γ, ξ_i x_i + η_i y_i, e^{ϵ β_i a_i/γ}]_{$k},
         t2b_i = 𝔼[α_i a_i - τ_i γ b_i, ξ_i x_i + η_i y_i, e^{ϵ τ_i a_i}]_{$k}]
```

Testing

```
In[•]:= BeginProfile[]
```

```
In[•]:= Block[{$k = 1}, {
          am → am_{i,j→k}, bm → bm_{i,j→k}, dm → dm_{i,j→k}, R → R_{i,j}, R$ → R$_{i,j}, P → P_{i,j},
          aS → aS_i, aS$ → a$S_i, bS → bS_i, bS$ → b$S_i, dS → dS_i, aΔ → aΔ_{i→j,k}, bΔ → bΔ_{i→j,k},
          dΔ → dΔ_{i→j,k}, C → C_i, C$ → C$_i, Kink → Kink_i, Kink$ → Kink$_i, b2t → b2t_i, t2b → t2b_i
         }] //
        Column
```

Check that on the generators this agrees with our conventions in the handout:

```
In[•]:= Timing@{{"[a,x]" → ((𝔼[0, 0, a_2 x_1] ~ B_{1,2} ~ am_{1,2→1}) ⟦3⟧ - (𝔼[0, 0, a_1 x_2] ~ B_{1,2} ~ am_{1,2→1}) ⟦3⟧),
            "[b,y]" → ((𝔼[0, 0, y_2 b_1] ~ B_{1,2} ~ bm_{1,2→1}) ⟦3⟧ - (𝔼[0, 0, y_1 b_2] ~ B_{1,2} ~ bm_{1,2→1}) ⟦3⟧)} /.
          z_{_1} → z,
         {"Δ[y]" → Last[𝔼[0, 0, y_1] ~ B_1 ~ bΔ_{1→1,2}],
          "Δ[b]" → Last[𝔼[0, 0, b_1] ~ B_1 ~ bΔ_{1→1,2}],
          "Δ[a]" → Last[𝔼[0, 0, a_1] ~ B_1 ~ aΔ_{1→1,2}],
          "Δ[x]" → Last[𝔼[0, 0, x_1] ~ B_1 ~ aΔ_{1→1,2}]},
         {
           "S(a)" → ((𝔼[0, 0, a_1] ~ B_1 ~ aS_1) ⟦3⟧),
           "S(x)" → ((𝔼[0, 0, x_1] ~ B_1 ~ aS_1) ⟦3⟧),
           "S(b)" → ((𝔼[0, 0, b_1] ~ B_1 ~ bS_1) ⟦3⟧),
           "S(y)" → ((𝔼[0, 0, y_1] ~ B_1 ~ bS_1) ⟦3⟧)
          } /. z_{_1} → z}
```

Hopf algebra axioms on both sides separately.

Associativity of am and bm:

```
In[•]:= Timing@Block[{$k = 3},
  HL /@ {(am[1,2→1] ~ B[1] ~ am[1,3→1]) ≡ (am[2,3→2] ~ B[2] ~ am[1,2→1]), (bm[1,2→1] ~ B[1] ~ bm[1,3→1]) ≡ (bm[2,3→2] ~ B[2] ~ bm[1,2→1])}
 ]
```

R and *P* are inverses:

```
In[•]:= Timing@Block[{$k = 3}, {R[i,j], P[i,k], HL[R[i,j] ~ B[i] ~ P[i,k] ≡ 𝔼[a[j] α[k], x[j] ξ[k], 1]]}]
```

as and $\overline{\text{aS}}$ are inverses, bs and $\overline{\text{bS}}$ are inverses:

```
In[•]:= Timing[HL /@ {aSbar[1] ~ B[1] ~ aS[1] ≡ 𝔼[a[1] α[1], x[1] ξ[1], 1], bSbar[1] ~ B[1] ~ bS[1] ≡ 𝔼[b[1] β[1], y[1] η[1], 1]}]
```

(co)-associativity on both sides

```
In[•]:= Timing[HL /@
  {(aΔ[1→1,2] ~ B[2] ~ aΔ[2→2,3]) ≡ (aΔ[1→1,3] ~ B[1] ~ aΔ[1→1,2]), (bΔ[1→1,2] ~ B[2] ~ bΔ[2→2,3]) ≡ (bΔ[1→1,3] ~ B[1] ~ bΔ[1→1,2]),
   (am[1,2→1] ~ B[1] ~ am[1,3→1]) ≡ (am[2,3→2] ~ B[2] ~ am[1,2→1]), (bm[1,2→1] ~ B[1] ~ bm[1,3→1]) ≡ (bm[2,3→2] ~ B[2] ~ bm[1,2→1])}]
```

Δ is an algebra morphism

```
In[•]:= Timing[HL /@ {am[1,2→1] ~ B[1] ~ aΔ[1→1,2] ≡ (aΔ[1→1,3] aΔ[2→2,4]) ~ B[1,2,3,4] ~ (am[3,4→2] am[1,2→1]),
    bm[1,2→1] ~ B[1] ~ bΔ[1→1,2] ≡ (bΔ[1→1,3] bΔ[2→2,4]) ~ B[1,2,3,4] ~ (bm[3,4→2] bm[1,2→1])}]
```

S is convolution inverse of id

```
In[•]:= Timing[HL[# ≡ 𝔼[0, 0, 1]] & /@ {
   (aΔ[1→1,2] ~ B[1] ~ aS[1]) ~ B[1,2] ~ am[1,2→1], (aΔ[1→1,2] ~ B[2] ~ aS[2]) ~ B[1,2] ~ am[1,2→1],
   (bΔ[1→1,2] ~ B[1] ~ bS[1]) ~ B[1,2] ~ bm[1,2→1], (bΔ[1→1,2] ~ B[2] ~ bS[2]) ~ B[1,2] ~ bm[1,2→1]}]
```

S is an algebra anti-(co)morphism

```
In[•]:= Timing[HL /@ {am[1,2→1] ~ B[1] ~ aS[1] ≡ (aS[1] aS[2]) ~ B[1,2] ~ am[2,1→1], bm[1,2→1] ~ B[1] ~ bS[1] ≡ (bS[1] bS[2]) ~ B[1,2] ~ bm[2,1→1],
    aS[1] ~ B[1] ~ aΔ[1→1,2] ≡ aΔ[1→2,1] ~ B[1,2] ~ (aS[1] aS[2]), bS[1] ~ B[1] ~ bΔ[1→1,2] ≡ bΔ[1→2,1] ~ B[1,2] ~ (bS[1] bS[2])}]
```

Pairing axioms

```
In[•]:= Timing[HL /@ {(bm[1,2→1] 𝔼[α[3] a[3], ξ[3] x[3], 1]) ~ B[1,3] ~ P[1,3] ≡
     (𝔼[β[1] b[1], η[1] y[1], 1] 𝔼[β[2] b[2], η[2] y[2], 1] aΔ[3→4,5]) ~ B[1,4] ~ P[1,4] ~ B[2,5] ~ P[2,5],
   (bΔ[1→1,2] 𝔼[α[3] a[3], ξ[3] x[3], 1] 𝔼[α[4] a[4], ξ[4] x[4], 1]) ~ B[1,3] ~ P[1,3] ~ B[2,4] ~ P[2,4] ≡
     (𝔼[β[1] b[1], η[1] y[1], 1] am[3,4→3]) ~ B[1,3] ~ P[1,3]}]

In[•]:= Timing[HL /@ {(bS[1] 𝔼[α[2] a[2], ξ[2] x[2], 1]) ~ B[1,2] ~ P[1,2] ≡ (𝔼[β[1] b[1], η[1] y[1], 1] aS[2]) ~ B[1,2] ~ P[1,2],
   (bSbar[1] 𝔼[α[2] a[2], ξ[2] x[2], 1]) ~ B[1,2] ~ P[1,2] ≡ (𝔼[β[1] b[1], η[1] y[1], 1] aSbar[2]) ~ B[1,2] ~ P[1,2]}]
```

Tests for the double.

Check the double formulas on the generators agree with SL2Portfolio.pdf:

```
In[•]:= Timing@{{
          "[a,y]" → ((𝔼[0, 0, y2 a1] ~B1,2~ dm1,2→1) ⟦3⟧ - (𝔼[0, 0, y1 a2] ~B1,2~ dm1,2→1) ⟦3⟧),
          "[b,x]"  → ((𝔼[0, 0, x2 b1] ~B1,2~ dm1,2→1) ⟦3⟧ - (𝔼[0, 0, x1 b2] ~B1,2~ dm1,2→1) ⟦3⟧),
          "xy-qyx" → ((𝔼[0, 0, x1 y2] ~B1,2~ dm1,2→1) ⟦3⟧ - (1 + ϵ) (𝔼[0, 0, y1 x2] ~B1,2~ dm1,2→1) ⟦3⟧)
        } /. {z_1 → z} // Expand // Factor,
       {
         "Δ(a)" → ((𝔼[0, 0, a1] ~B1~ dΔ1→1,2) ⟦3⟧),
         "Δ(x)" → ((𝔼[0, 0, x1] ~B1~ dΔ1→1,2) ⟦3⟧),
         "Δ(b)" → ((𝔼[0, 0, b1] ~B1~ dΔ1→1,2) ⟦3⟧),
         "Δ(y)" → ((𝔼[0, 0, y1] ~B1~ dΔ1→1,2) ⟦3⟧)
        } // Simplify,
       {
          "S(a)" → ((𝔼[0, 0, a1] ~B1~ dS1) ⟦3⟧),
          "S(x)" → ((𝔼[0, 0, x1] ~B1~ dS1) ⟦3⟧),
          "S(b)" → ((𝔼[0, 0, b1] ~B1~ dS1) ⟦3⟧),
          "S(y)" → ((𝔼[0, 0, y1] ~B1~ dS1) ⟦3⟧)
        } /. {z_1 → z} // Simplify
       }
```

(co)-associativity

```
In[•]:= Timing[HL /@
          {(dΔ1→1,2 ~B2~ dΔ2→2,3) ≡ (dΔ1→1,3 ~B1~ dΔ1→1,2), (dm1,2→1 ~B1~ dm1,3→1) ≡ (dm2,3→2 ~B2~ dm1,2->1)}]
```

Δ is an algebra morphism

```
In[•]:= Timing@HL[dm1,2→1 ~B1~ dΔ1→1,2 ≡ (dΔ1→1,3 dΔ2→2,4) ~B1,2,3,4~ (dm3,4→2 dm1,2→1)]
```

S is convolution inverse of id

```
In[•]:= Timing[
          HL[# ≡ 𝔼[0, 0, 1]] & /@ {(dΔ1→1,2 ~B1~ dS1) ~B1,2~ dm1,2→1, (dΔ1→1,2 ~B2~ dS2) ~B1,2~ dm1,2→1}]
```

S is a (co)-algebra anti-morphism

```
In[•]:= Timing[HL /@
          Expand /@ {dm1,2→1 ~B1~ dS1 ≡ (dS1 dS2) ~B1,2~ dm2,1→1, dS1 ~B1~ dΔ1→1,2 ≡ dΔ1→2,1 ~B1,2~ (dS1 dS2)}]
```

Quasi-triangular axiom 1:

```
In[•]:= Timing@HL[R1,2 ~B1~ dΔ1->1,3 ≡ (R1,4 R3,2) ~B2,4~ dm2,4→2]
```

Quasi-triangular axiom 2:

```
In[•]:= Timing@HL[((dΔ1->1,2 R3,4) ~B1,2,3,4~ (dm1,3→1 dm2,4→2)) ≡ ((dΔ1→2,1 R3,4) ~B1,2,3,4~ (dm3,1→1 dm4,2→2))]
```

The Drinfel'd element inverse property, $(u_1\,\overline{u}_2)\sim B_{1,2}\sim \mathrm{dm}_{1,2\to 1} \equiv \mathbb{E}[0,\,0,\,1]$:

```
In[•]:= Timing@
          HL[((R1,2 ~B1~ dS1 ~B1,2~ dm2,1→i) (R1,2 ~B2~ dS2 ~B2~ dS2 ~B1,2~ dm2,1→j)) ~Bi,j~ dmi,j→i ≡ 𝔼[0, 0, 1]]
```

The ribbon element v satisfies $v^2 = S(u)\,u$. The spinner C=uv^{-1}. It is convenient to compute $z = S(u)\,u^{-1}$ which is something easy.

```
In[•]:= Timing@Block[{$k = 3},
  (((R1,2 ~ B1 ~ dS1 ~ B1,2 ~ dm2,1→i) ~ Bi ~ dSi) (R1,2 ~ B2 ~ dS2 ~ B2 ~ dS2 ~ B1,2 ~ dm2,1→j)) ~ Bi,j ~ dmi,j→i]

In[•]:= Timing@Block[{$k = 2}, HL /@ {(Ci C̄j) ~ Bi,j ~ dmi,j→i ≡ 𝔼[0, 0, 1], (C̄i C̄j) ~ Bi,j ~ dmi,j→i ≡
    (((R1,2 ~ B1 ~ dS1 ~ B1,2 ~ dm2,1→i) ~ Bi ~ dSi) (R1,2 ~ B2 ~ dS2 ~ B2 ~ dS2 ~ B1,2 ~ dm2,1→j)) ~ Bi,j ~ dmi,j→i}]
```

Reidemeister 2:

```
In[•]:= Timing[HL[# ≡ 𝔼[0, 0, 1]] & /@
  {(R̄1,2 R3,4) ~ B1,2,3,4 ~ (dm1,3→1 dm2,4→2), (R1,2 R̄3,4) ~ B1,2,3,4 ~ (dm1,3→1 dm2,4→2)}]
```

Cyclic Reidemeister 2:

```
In[•]:= Timing@HL[(R1,4 R̄5,2 C̄3) ~ B2,4 ~ dm2,4→2 ~ B1,3 ~ dm1,3→1 ~ B1,5 ~ dm1,5→1 ≡ C̄1]
```

Reidemeister 3:

```
In[•]:= Timing@HL[((R1,2 R4,3 R5,6) ~ B1,4 ~ dm1,4→1 ~ B2,5 ~ dm2,5→2 ~ B3,6 ~ dm3,6→3) ≡
    ((R1,6 R2,3 R4,5) ~ B1,4 ~ dm1,4→1 ~ B2,5 ~ dm2,5→2 ~ B3,6 ~ dm3,6→3)]
```

Relations between the four kinks:

```
In[•]:= Timing[HL /@ {Kinki ≡ (R3,1 C2) ~ B1,2 ~ dm1,2→1 ~ B1,3 ~ dm1,3→i,
    K̄ink̄j ≡ (R̄3,1 C̄2) ~ B1,2 ~ dm1,2→1 ~ B1,3 ~ dm1,3→j, (Kinki K̄ink̄j) ~ Bi,j ~ dmi,j→1 ≡ 𝔼[0, 0, 1]}]
```

The Trefoil

```
In[•]:= Timing@Block[{$k = 1},
  Z = R1,5 R6,2 R3,7 C̄4 K̄ink̄8 K̄ink̄9 K̄ink̄10;
  Do[Z = Z ~ B1,r ~ dm1,r→1, {r, 2, 10}];
  {Simplify /@ Z, Simplify /@ (Z ~ B1 ~ b2t1 /. T1 → T)}]
```

Program

```
In[•]:= Define[kRi,j = Ri,j ~ Bi,j ~ (b2ti b2tj) /. ti|j → t,
  k̄R̄i,j = R̄i,j ~ Bi,j ~ (b2ti b2tj) /. ti|j → t,
  kmi,j→k = (t2bi t2bj) ~ Bi,j ~ dmi,j→k ~ Bk ~ b2tk /. {tk → t, Tk → T, τi|j → 0},
  kCi = Ci ~ Bi ~ b2ti /. Ti → T,
  k̄C̄i = C̄i ~ Bi ~ b2ti /. Ti → T,
  kKinki = Kinki ~ Bi ~ b2ti /. {ti → t, Ti → T},
  k̄K̄ink̄i = K̄ink̄i ~ Bi ~ b2ti /. {ti → t, Ti → T}]
```

Trefoil

```
In[•]:= Timing@Block[{$k = 1},
  Z = kR1,5 kR6,2 kR3,7 k̄C̄4 k̄K̄ink̄8 k̄K̄ink̄9 k̄K̄ink̄10;
  Do[Z = Z ~ B1,r ~ km1,r→1, {r, 2, 10}];
  Simplify /@ Z]
```

RVK, rot, Z from 2016-09/OneSmidgen.nb.

Program

```
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.";
```

Program

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

In[•]:=
```
RVK[Knot[3, 1]]
```

Program

In[•]:=
```
rot[_, 0] = 𝔼[0, 0, 1];
rot[i_, n_Integer] /; n > 0 :=
  rot[i, n] = Module[{j}, (rot[i, n - 1] kC_j) ~B_{i,j}~ km_{i,j→i}];
rot[i_, n_Integer] /; n < 0 := rot[i, n] = Module[{j}, (rot[i, n + 1] \overline{kC}_j) ~B_{i,j}~ km_{i,j→i}];
```

In[•]:=
```
rot[i, -3]
```

Program

In[•]:=
```
Z[K_] := Z[RVK@K];
Z[rvk_RVK] := Z[rvk] = Module[{todo, n, rots, ℱ, done, st, x, ℱ1, i, j, k, k1, k2, k3},
   {todo, rots} = List @@ rvk;
   AppendTo[rots, 0];
   n = Length[todo];
   ℱ = 𝔼[0, 0, 1];
   done = {0};
   st = Range[0, 2 n + 1];
   While[todo =!= {},
    {x} = MaximalBy[todo, Length[done ⋂ {#[[1]], #[[2]], #[[1]] - 1, #[[2]] - 1}] &, 1];
    Z$todo = todo; Z$x = x;
    {i, j} = List @@ x;
    ℱ1 = Switch[Head[x],
       Xp, m_{j,k→j}[R⁺_{i,j} (R⁻_{k3,k} nr_{k1} ul_{k2} // m_{k,k1→k} // m_{k,k2→k} // m_{k,k3→k})],
       Xm, m_{j,k→j}[R⁻_{i,j} (R⁺_{k,k3} nr_{k1} ul_{k2} // m_{k,k1→k} // m_{k,k2→k} // m_{k,k3→k})]
     ];
    ℱ1 = rot[k, rots[[i]]] ℱ1 // m_{k,i→i}; rots[[i]] = 0;
    ℱ1 = ℱ1 rot[k, rots[[i + 1]]] // m_{i,k→i}; rots[[i + 1]] = 0;
    ℱ1 = rot[k, rots[[j]]] ℱ1 // m_{k,j→j}; rots[[j]] = 0;
    ℱ1 = ℱ1 rot[k, rots[[j + 1]]] // m_{j,k→j}; rots[[j + 1]] = 0;
    ℱ *= ℱ1;
    If[MemberQ[done, i], ℱ = ℱ // m_{i,i+1→i}; st = st /. st[[i + 2]] → st[[i + 1]]];
    If[MemberQ[done, i - 1], ℱ = ℱ // m_{st[[i]],i→st[[i]]}; st = st /. st[[i + 1]] → st[[i]]];
    If[MemberQ[done, j], ℱ = ℱ // m_{j,j+1→j}; st = st /. st[[j + 2]] → st[[j + 1]]];
    If[MemberQ[done, j - 1], ℱ = ℱ // m_{st[[j]],j→st[[j]]}; st = st /. st[[j + 1]] → st[[j]]];
    done = done ⋃ {i - 1, i, j - 1, j};
    todo = DeleteCases[todo, x]
   ];
   ℱ /. {u_0 → u, c_0 → c, w_0 → w}
  ]
```

In[•]:=
```
EndProfile[];
```

Profile

In[•]:=
```
PrintProfile[]
```