Pensieve header: By Roland, June 19, 2020.

Introducing the q-Casimir w to all orders and expressing and computing the invariant in terms of it.

Pensieve header: The “Speedy” engine.

In[ ]:=
```
Once[<< KnotTheory`];
```

```
Loading KnotTheory` version of September 6, 2014, 13:37:37.2841.
Read more at http://katlas.org/wiki/KnotTheory.
```

In[ ]:=
```
PP_ = Identity; $k = 0; γ = 1; ℏ = 1;
```

The “Speedy” Engine

Internal Utilities

Canonical Form:

In[ ]:=
```
CCF[ℰ_] := ExpandDenominator@ExpandNumerator@Together[
    Expand[ℰ] //. 𝕖^x_ 𝕖^y_ :→ 𝕖^(x+y) /. 𝕖^x_ :→ 𝕖^CCF[x]];
CF[ℰ_List] := CF /@ ℰ;
CF[sd_SeriesData] := MapAt[CF, sd, 3];
CF[ℰ_] := Module[
   {vs = Cases[ℰ, (y | b | t | a | w | x | η | β | τ | α | ω | ξ)_, ∞] ⋃
      {y, b, t, a, w, x, η, β, τ, α, ω, ξ}},
   Total[CoefficientRules[Expand[ℰ], vs] /. (ps_ → c_) :→ CCF[c] (Times @@ vs^ps)]
  ];
CF[ℰ_𝔼] := CF /@ ℰ; CF[𝔼_sp___[ℰs___]] := CF /@ 𝔼_sp[ℰs];
```

The Kronecker δ:

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

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

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}]];
```

Zip and Bind

Variables and their duals:

In[∘]:=
```
{t*, b*, y*, a*, w*, x*, z*} = {τ, β, η, α, ω, ξ, ζ};
{τ*, β*, η*, α*, ω*, ξ*, ζ*} = {t, b, y, a, w, x, z}; (u_i_)* := (u*)_i;
```

Upper to lower and lower to Upper:

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), Ω_i_^p_. :> e^(p ω_i), Ω^p_. :> e^(p ω)};
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^(c_. ω_i_+d_.) :> Ω_i^c e^d, e^(c_. ω+d_.) :> Ω^c e^d,
   e^ℰ_ :> e^Expand@ℰ};
```

Derivatives in the presence of exponentiated variables:

In[∘]:=
```
D_b[f_] := ∂_b f - ħ γ B ∂_B f; D_b_i_[f_] := ∂_b_i f - ħ γ 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 + γ A ∂_A f; D_α_i_[f_] := ∂_α_i f + γ A_i ∂_A_i f;
D_ω[f_] := ∂_ω f + Ω ∂_Ω f; D_ω_i_[f_] := ∂_ω_i f + Ω_i ∂_Ω_i f;
D_v_[f_] := ∂_v f; D_{v_,0}[f_] := f; D_{}[f_] := f; D_{v_,n_Integer}[f_] := D_v[D_{v,n-1}[f]];
D_{l_List,ls___}[f_] := D_{ls}[D_l[f]];
```

Finite Zips:

In[∘]:=
```
collect[sd_SeriesData, ζ_] := MapAt[collect[#, ζ] &, sd, 3];
collect[ℰ_, ζ_] := Collect[ℰ, ζ];
Zip_{}[P_] := P;
Zip_ζs_[Ps_List] := Zip_ζs /@ Ps;
Zip_{ζ_,ζs___}[P_] :=
  (collect[P // Zip_{ζs}, ζ] /. f_. ζ^d_. :> (D_{ζ*,d}[f])) /. ζ* → 0 /.
   ((ζ* /. {b → B, t → T, α → A, ω → Ω}) → 1)
```

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

$$\left\langle P(z_i,\zeta^j)e^{c+\eta^i z_i+y_j\zeta^j+q^i_j z_i\zeta^j}\right\rangle = |\tilde{q}|\left\langle P(z_i,\zeta^j)e^{c+\eta^i z_i}\Big|_{z_i\to\tilde{q}^k_i(z_k+y_k)}\right\rangle$$
$$= |\tilde{q}|e^{c+\eta^i\tilde{q}^k_i y_k}\left\langle P\left(\tilde{q}^k_i(z_k+y_k),\zeta^j+\eta^i\tilde{q}^j_i\right)\right\rangle.$$

In[∘]:=
```
QZip_ζs_List@𝔼[L_, Q_, P_] := Module[{ζ, z, zs, c, ys, ηs, qt, zrule, ζrule, out},
   zs = Table[ζ*, {ζ, ζs}];
   c = CF[Q /. Alternatives @@ (ζs ⋃ zs) → 0];
   ys = CF@Table[∂_ζ (Q /. Alternatives @@ zs → 0), {ζ, ζs}];
   ηs = CF@Table[∂_z (Q /. Alternatives @@ ζs → 0), {z, zs}];
   qt = CF@Inverse@Table[Kδ_{z,ζ*} - ∂_{z,ζ}Q, {ζ, ζs}, {z, zs}];
   zrule = Thread[zs → CF[qt.(zs + ys)]];
   ζrule = Thread[ζs → ζs + ηs.qt];
   CF /@ 𝔼[L, c + ηs.qt.ys, Det[qt] Zip_ζs[P /. (zrule ⋃ ζrule)]] ];
```

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

In[∘]:=
```
LZip_ζs_List@𝔼[L_, Q_, P_] :=
  Module[{ζ, z, zs, Zs, c, ys, ηs, lt, zrule, Zrule, ζrule, Q1, EEQ, EQ},
   (*Print["LZip"];*)
   zs = Table[ζ*, {ζ, ζs}];
   Zs = zs /. {b → B, t → T, α → 𝒜, ω → Ω};
   c = L /. Alternatives @@ (ζs ⋃ zs) → 0 /. Alternatives @@ Zs → 1;
   ys = Table[∂_ζ (L /. Alternatives @@ zs → 0), {ζ, ζs}];
   ηs = Table[∂_z (L /. Alternatives @@ ζs → 0), {z, zs}];
   lt = Inverse@Table[Kδ_{z,ζ*} - ∂_{z,ζ}L, {ζ, ζs}, {z, zs}];
   zrule = Thread[zs → lt.(zs + ys)];
   Zrule = Join[zrule, zrule /.
      r_Rule :→ ((U = r[[1]] /. {b → B, t → T, α → 𝒜, ω → Ω}) → (U /. U2l /. r //. l2U))];
   ζrule = Thread[ζs → ζs + ηs.lt];
   Q1 = Q /. (Zrule ⋃ ζrule);
   EEQ[ps___] := EEQ[ps] =
     (CF[e^-Q1 D_Thread[{zs,{ps}}][e^Q1]] /. {Alternatives @@ zs → 0, Alternatives @@ Zs → 1});
   CF@𝔼[c + ηs.lt.ys, Q1 /. {Alternatives @@ zs → 0, Alternatives @@ Zs → 1},
     Det[lt] (Zip_ζs[(EQ @@ zs) (P /. (Zrule ⋃ ζrule))] /.
         Derivative[ps___][EQ][___] :→ EEQ[ps] /. _EQ → 1) ] ];
```

In[∘]:=
```
TZip_ζs_List@𝔼[L_, Q_, P_] := Module[{ζ, z, zs, Zs, c, ys, ηs,
    lt, zrule, Zrule, ζrule, Q1, EEQ, EQ, Lnew = L, Qnew = Q, Pnew = P},
   zrule = Table[ζ* → Coefficient[L, ζ], {ζ, ζs}];
   (*Print["Tzip"];*)
   ζrule = Table[ζ → 0, {ζ, ζs}];
   Lnew = L /. U2l /. zrule /. ζrule;
   Qnew = Q /. U2l /. zrule /. ζrule; (**)
   Pnew = P /. U2l /. zrule /. ζrule;
   CF@(𝔼[Lnew, Qnew, Pnew] //. l2U)
  ];
```

In[∘]:=
```
B{}[L_, R_] := L R;
B{is__}[L_𝔼, R_𝔼] := Module[{n},
   Times[
     L /. Table[(ν : b | B | t | T | a | w | x | y)_i → v_n@i, {i, {is}}],
     R /. Table[(ν : β | τ | α | 𝒜 | ω | Ω | ξ | η)_i → v_n@i, {i, {is}}]
    ] // TZip_Join@@Table[{τ_n@i},{i,{is}}] // LZip_Join@@Table[{w_n@i,β_n@i,a_n@i},{i,{is}}] //
   QZip_Join@@Table[{ξ_n@i,y_n@i},{i,{is}}] ];(**)
B_is___[L_, R_] := B{is}[L, R];
```

𝔼 morphisms with domain and range.

In[∘]:=
```
B_is_List[𝔼_d1_→r1_[L1_, Q1_, P1_], 𝔼_d2_→r2_[L2_, Q2_, P2_]] :=
  𝔼_(d1⋃Complement[d2,is])→(r2⋃Complement[r1,is]) @@ B_is[𝔼[L1, Q1, P1], 𝔼[L2, Q2, P2]];
𝔼_d1_→r1_[L1_, Q1_, P1_] // 𝔼_d2_→r2_[L2_, Q2_, P2_] :=
  B_r1⋂d2[𝔼_d1→r1[L1, Q1, P1], 𝔼_d2→r2[L2, Q2, P2]];
𝔼_d1_→r1_[L1_, Q1_, P1_] ≡ 𝔼_d2_→r2_[L2_, Q2_, P2_] ^:=
  (d1 == d2) ∧ (r1 == r2) ∧ (𝔼[L1, Q1, P1] ≡ 𝔼[L2, Q2, P2]);
𝔼_d1_→r1_[L1_, Q1_, P1_] 𝔼_d2_→r2_[L2_, Q2_, P2_] ^:=
  𝔼_(d1⋃d2)→(r1⋃r2) @@ (𝔼[L1, Q1, P1] 𝔼[L2, Q2, P2]);
𝔼_dr_[L_, Q_, P_]_$k_ := 𝔼_dr @@ 𝔼[L, Q, P]_$k;
𝔼_[ℰ___][i_] := {ℰ}[[i]];
```

𝔼[Λ]

In[∘]:=
```
𝔼_dr_[Λ_] := CF@
  Module[{L, Λ0 = Limit[Λ, ϵ → 0]}, 𝔼_dr[L = Λ0 /. (η | y | ξ | x)_ → 0, Λ0 - L, ℯ^(Λ-Λ0)]_$k /. l2U]
```

“Define” Code

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.

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,$k_Integer, Block[{i, j, k}, op_isp,$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

In[∘]:=
```
sm_{i_,j_→k_} := 𝔼_{{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_} := 𝔼_{{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_} := 𝔼_{{i}→{i}}[-β_i b_i - τ_i t_i - α_i a_i - η_i y_i - ξ_i x_i];
sϵ_{i_} := 𝔼_{{}→{i}}[0];
sη_{i_} := 𝔼_{{i}→{}}[0];
```

In[∘]:=
```
sσ_{i_→j_} := 𝔼_{{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_} := 𝔼_{{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];
```

Booting Up QU

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

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

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 $V\otimes V$.

$\overline{\mathrm{aS}}$ 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 $V^*\otimes V$.

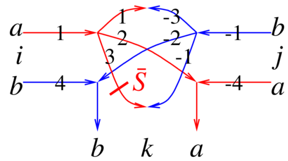
In[∘]:=
```
Define[R_{i,j} = 𝔼_{{}→{i,j}}[ℏ a_j b_i + Sum_{k=1}^{$k+1} ((1 - e^{γ ϵ ℏ})^k (ℏ y_i x_j)^k) / (k (1 - e^{k γ ϵ ℏ}))],
  R̄_{i,j} = CF@𝔼_{{}→{i,j}}[-ℏ a_j b_i, -ℏ x_j y_i / B_i, 1 + If[$k == 0, 0, (R̄_{{i,j},$k-1})_{$k}[3] -
       (((R̄_{{i,j},0})_{$k} R_{1,2} (R̄_{{3,4},$k-1})_{$k}) // (bm_{i,1→i} am_{j,2→j}) // (bm_{i,3→i} am_{j,4→j}))[3]]],
  P_{i,j} = 𝔼_{{i,j}→{}}[β_i α_j / ℏ, η_i ξ_j / ℏ, 1 + If[$k == 0, 0, (P_{{i,j},$k-1})_{$k}[3] -
      (R_{1,2} // ((P_{{1,j},0})_{$k} (P_{{i,2},$k-1})_{$k}))[3]]]]
```

In[∘]:=
```
Define[aS_i = (aσ_{i→2} R̄_{1,i}) // P_{1,2},
  aS̄_i = 𝔼_{{i}→{i}}[-a_i α_i, -x_i 𝒜_i ξ_i, 1 + If[$k == 0, 0, (aS̄_{{i},$k-1})_{$k}[3] -
       ((aS̄_{{i},0})_{$k} // aS_i // (aS̄_{{i},$k-1})_{$k})[3]]]]
```

(was $\mathrm{aS}_j = \overline{R}_{i,j} \sim B_i \sim P_{i,j}$).

In[∘]:=
```
Define[bS_i = bσ_{i→1} R_{i,2} // aS_2 // P_{1,2},
  bS̄_i = bσ_{i→1} R_{i,2} // aS̄_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}]
```

(was $\mathrm{bS}_i = R_{i,1} \sim B_1 \sim \mathrm{aS}_1 \sim B_1 \sim P_{i,1}$, $\overline{\mathrm{bS}}_i = R_{i,1} \sim B_1 \sim \overline{\mathrm{aS}}_1 \sim B_1 \sim P_{i,1}$).

The Drinfel'd double: 

In[∘]:=
```
Define[
  dm_{i,j→k} = ((sΥ_{i→4,4,1,1} // aΔ_{1→1,2} // aΔ_{2→2,3} // \overline{aS}_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})]
```

In[∘]:=
```
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} // (\overline{bS}_1 aS_2) // dm_{2,1→i},
  \overline{dS}_i = sΥ_{i→1,1,2,2} // (bS_1 \overline{aS}_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})]
```

In[∘]:=
```
Define[C_i = 𝔼_{{}→{i}}[0, 0, B_i^{1/2} e^{-ℏ ϵ a_i/2}]_{$k},
  \overline{C}_i = 𝔼_{{}→{i}}[0, 0, B_i^{-1/2} e^{ℏ ϵ a_i/2}]_{$k},
  Kink_i = (R_{1,3} \overline{C}_2) // dm_{1,2→1} // dm_{1,3→i},
  \overline{Kink}_i = (\overline{R}_{1,3} C_2) // dm_{1,2→1} // dm_{1,3→i}]
```

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

In[∘]:=
```
Define[b2t_i = 𝔼_{{i}→{i}}[α_i a_i + β_i (ϵ a_i + t_i) / γ + ξ_i x_i + η_i y_i],
  t2b_i = 𝔼_{{i}→{i}}[α_i a_i + τ_i (-ϵ a_i + γ b_i) + ξ_i x_i + η_i y_i]]
```

The t-Tensors

In[∘]:=
```
Define[tR_{i,j} = R_{i,j} // (b2t_i b2t_j),
  \overline{tR}_{i,j} = \overline{R}_{i,j} // (b2t_i b2t_j) ,
  tm_{i,j→k} = ((t2b_i t2b_j) // dm_{i,j→k} // b2t_k ),
  tC_i = (C_i // b2t_i ),
  \overline{tC}_i = (\overline{C}_i // b2t_i ),
  tKink_i = Kink_i // b2t_i ,
  \overline{tKink}_i = \overline{Kink}_i // b2t_i ,
  tΔ_{i→j,k} = t2b_i // dΔ_{i->j,k} // (b2t_j b2t_k),
  tS_i = t2b_i // dS_i // b2t_i]
```

Use the central variable $\mathrm{w} = \frac{1}{2} + \mathrm{a} + \frac{\mathrm{x\,y}}{1-\mathrm{T}}$

In[∘]:=

```
a2w_{i_} := 𝔼_{{i}→{i}}[τ_i t_i + α_i (-1/2 + w_i), (ⅇ^{-α_i} - 1) (y_i x_i)/(1 - T_i) + ξ_i x_i + η_i y_i, 1]
w2a_{i_} := 𝔼_{{i}→{i}}[τ_i t_i + (a_i + 1/2) ω_i, (1 - ⅇ^{-ω_i})/(1 - T_i) y_i x_i + ξ_i x_i + η_i y_i, 1]
```

Up to some notational annoyance the kink is exp(tw)

In[∘]:=

```
tKink_i // a2w_i
tKink-bar_i // a2w_i
```

Out[∘]= $\mathbb{E}_{\{\}\to\{i\}}\left[-\frac{t_i}{2} + t_i w_i,\ 0,\ \frac{1}{\sqrt{T_i}} + O[\epsilon]^1\right]$

Out[∘]= $\mathbb{E}_{\{\}\to\{i\}}\left[\frac{t_i}{2} - t_i w_i,\ 0,\ \sqrt{T_i} + O[\epsilon]^1\right]$

The R-matrix becomes complicated!:

In[∘]:=

```
tR_{i,j} // a2w_i // a2w_j
tR-bar_{i,j} // a2w_i // a2w_j
w2a_i // w2a_j // tm_{i,j→k} // a2w_k
```

Out[∘]= $\mathbb{E}_{\{\}\to\{i,j\}}\left[-\frac{t_i}{2} + t_i w_j,\ x_j y_i + \frac{(1 - T_i)\, x_j y_j}{-1 + T_j},\ 1 + O[\epsilon]^1\right]$

Out[∘]= $\mathbb{E}_{\{\}\to\{i,j\}}\left[\frac{t_i}{2} - t_i w_j,\ -\frac{x_j y_i}{T_i} + \frac{(-1 + T_i)\, x_j y_j}{-T_i + T_i T_j},\ 1 + O[\epsilon]^1\right]$

Out[∘]= $\mathbb{E}_{\{i,j\}\to\{k\}}\left[t_k \tau_i + t_k \tau_j + w_k \omega_i + w_k \omega_j,\ y_k \eta_i + y_k \eta_j + x_k \xi_i + (1 - T_k)\, \eta_j \xi_i + x_k \xi_j,\ 1 + O[\epsilon]^1\right]$

Passing to constant t and w will simplify things further:

In[∘]:=

```
(tR_{i,j} // a2w_i // a2w_j) //. {t_ → t, T_ → T, w_ → w} // Simplify
(tR-bar_{i,j} // a2w_i // a2w_j) //. {t_ → t, T_ → T, w_ → w} // Simplify
(w2a_i // w2a_j // tm_{i,j→k} // a2w_k) /. {τ_ → 0, ω_ → 0, t_ → t, T_ → T, w_ → w} // Simplify
```

Out[∘]= $\mathbb{E}_{\{\}\to\{i,j\}}\left[t\left(-\frac{1}{2} + w\right),\ x_j (y_i - y_j),\ 1 + O[\epsilon]^1\right]$

Out[∘]= $\mathbb{E}_{\{\}\to\{i,j\}}\left[\frac{1}{2}(t - 2 t w),\ -\frac{x_j (y_i - y_j)}{T},\ 1 + O[\epsilon]^1\right]$

Out[∘]= $\mathbb{E}_{\{i,j\}\to\{k\}}\left[0,\ y_k (\eta_i + \eta_j) - (-1 + T)\, \eta_j \xi_i + x_k (\xi_i + \xi_j),\ 1 + O[\epsilon]^1\right]$

So let's define the newly found building blocks independently: (recall T = exp(-t) so the annoying $-\frac{t}{2}$ in the L part is really Sqrt[T] in the P part.)

In[∘]:=
```
Define[
  wR_{i,j} = 𝔼_{{}→{i,j}}[t w, x_j (y_i - y_j), Sqrt[T]],
  \overline{wR}_{i,j} = 𝔼_{{}→{i,j}}[-t w, x_j (-y_i + y_j)/T, 1/Sqrt[T]],
  wC_i = 𝔼_{{}→{i}}[0, 0, Sqrt[T]],
  \overline{wC}_i = 𝔼_{{}→{i}}[0, 0, 1/Sqrt[T]],
  wm_{i,j→k} = 𝔼_{{i,j}→{k}}[0, y_k (η_i + η_j) - (-1 + T) η_j ξ_i + x_k (ξ_i + ξ_j), 1]
]
```

Almost matching Γ calculus

Checking Reidemeister 1: (it is satisfied up to an overall factor of $e^{+-w t}$)

In[∘]:=
```
wR_{1,2} \overline{wC}_3 // wm_{1,3→1} // wm_{1,2→1}
\overline{wR}_{1,2} wC_3 // wm_{1,3→1} // wm_{1,2→1}
\overline{wR}_{1,2} \overline{wC}_3 // wm_{2,3→2} // wm_{2,1→1}
wR_{1,2} wC_3 // wm_{2,3→2} // wm_{2,1→1}
```

Out[∘]= $\mathbb{E}_{\{\}\to\{1\}}[t\,w, 0, 1]$

Out[∘]= $\mathbb{E}_{\{\}\to\{1\}}[-t\,w, 0, 1]$

Out[∘]= $\mathbb{E}_{\{\}\to\{1\}}[-t\,w, 0, 1]$

Out[∘]= $\mathbb{E}_{\{\}\to\{1\}}[t\,w, 0, 1]$

Checking Reidemeister 2:

In[∘]:=
```
wR_{1,2} \overline{wR}_{3,4} // wm_{1,3->1} // wm_{2,4→2}
```

Out[∘]= $\mathbb{E}_{\{\}\to\{1,2\}}[0, 0, 1]$

Checking Reidemeister 3:

In[∘]:=
```
(wR_{1,2} wR_{4,3} wR_{5,6} // wm_{1,4→1} // wm_{2,5→2} // wm_{3,6→3}) ≡
  (wR_{2,3} wR_{1,6} wR_{4,5} // wm_{1,4→1} // wm_{2,5→2} // wm_{3,6→3})
```

Out[∘]= True

Trefoil knot

In[∘]:=
```
(wR_{5,1} wR_{2,6} wR_{7,3} wC_4 // wm_{1,2→1} // wm_{1,3→1} // wm_{1,4→1} // wm_{1,5→1} // wm_{1,6→1} // wm_{1,7→1})
```

Out[∘]= $\mathbb{E}_{\{\}\to\{1\}}\left[3\,t\,w, 0, \frac{T}{1 - T + T^2}\right]$

Let’s look at the product in Γ calculus style. Caution: variables γ and ϵ are in use, use g and e instead. Taking the opposite product almost gives Γ calc. The Weyl-term in wm probably needs to be 1 not 1-T so we could rescale x,y or both to make their commutator 1. Also the product is opposite.

In[∘]:=
```
Prod = 𝔼_{{}→{a,b,S}}[0,
   α y_a x_a + β y_a x_b + g y_b x_a + δ y_b x_b + Θ y_a x_S + e y_b x_S + Ξ y_S x_S + ϕ y_S x_a + ψ y_S x_b, MM] // wm_{b,a→c}
```

Out[∘]=
$$\mathbb{E}_{\{\}\to\{c,S\}}\Big[0,\ \frac{1}{1-\beta+T\beta}\,(g\,x_c\,y_c+\alpha\,x_c\,y_c+\beta\,x_c\,y_c-g\,\beta\,x_c\,y_c+g\,T\,\beta\,x_c\,y_c+\delta\,x_c\,y_c+\alpha\,\delta\,x_c\,y_c-T\,\alpha\,\delta\,x_c\,y_c+e\,x_S\,y_c-e\,\beta\,x_S\,y_c+e\,T\,\beta\,x_S\,y_c+\Theta\,x_S\,y_c+\delta\,\Theta\,x_S\,y_c-T\,\delta\,\Theta\,x_S\,y_c+\phi\,x_c\,y_S-\beta\,\phi\,x_c\,y_S+T\,\beta\,\phi\,x_c\,y_S+\psi\,x_c\,y_S+\alpha\,\psi\,x_c\,y_S-T\,\alpha\,\psi\,x_c\,y_S+\Xi\,x_S\,y_S-\beta\,\Xi\,x_S\,y_S+T\,\beta\,\Xi\,x_S\,y_S+\Theta\,\psi\,x_S\,y_S-T\,\Theta\,\psi\,x_S\,y_S),\ \frac{MM}{1-\beta+T\beta}\Big]$$

In[∘]:=
```
Coefficient[Prod[[2]], y_c x_c] - g // FullSimplify
Coefficient[Prod[[2]], y_c x_S] - e // FullSimplify
Coefficient[Prod[[2]], y_S x_c] - ϕ // FullSimplify
Coefficient[Prod[[2]], y_S x_S] - Ξ // FullSimplify
```

Out[∘]= $$\frac{\beta+\delta+\alpha\,(1+\delta-T\,\delta)}{1+(-1+T)\,\beta}$$

Out[∘]= $$\frac{(1+\delta-T\,\delta)\,\Theta}{1+(-1+T)\,\beta}$$

Out[∘]= $$\frac{(1+\alpha-T\,\alpha)\,\psi}{1+(-1+T)\,\beta}$$

Out[∘]= $$-\frac{(-1+T)\,\Theta\,\psi}{1+(-1+T)\,\beta}$$