Pensieve header: A unified verification program for the sl_2-portfolio project, Uxi version. Continues pensieve://Projects/SL2Portfolio/nb/Verification.pdf.

Also continues pensieve://Projects/PPSA/nb/Verification.pdf and pensieve://2017-06/ and pensieve://2017-08/.

DocileQ

```
In[∘]:= DQ[ℰ_] := (Exponent[Normal@ℰ /.
        {a → a / ϵ, a_i_ → a_i / ϵ, (u : x | y) ⧴ ϵ^-1/2 u, (u : x | y)_i_ ⧴ ϵ^-1/2 u_i}, ϵ, Min] ≥ 0);
```

Initialization / Utilities

It is verification-risky to work with low $E!

```
In[∘]:= $p = 2; $k = 1; $U = QU; $E := {$k, $p};
$trim := {ħ^p_· /; p > $p → 0, ϵ^k_· /; k > $k → 0};
SetAttributes[{SS, SST}, HoldAll];
T2t = {T_i_^p_· → ⅇ^(p ħ t_i), T^p_· → ⅇ^(p ħ t)}; q_ħ = ⅇ^(γ ϵ ħ);
t2T = {ⅇ^(c_· t_i_ + b_·) ⧴ T_i^(c/ħ) ⅇ^b, ⅇ^(c_· t + b_·) ⧴ T^(c/ħ) ⅇ^b, ⅇ^ℰ_ ⧴ ⅇ^Expand@ℰ};
SS[ℰ_, op_] := Collect[
   Normal@Series[If[$p > 0, ℰ, ℰ /. T2t], {ħ, 0, $p}],
   ħ, op];
SS[ℰ_] := SS[ℰ, Together];
SST[ℰ_, op___] := SS[ℰ /. T2t, op];
Simp[ℰ_, op_] := Collect[ℰ, _CU | _QU, op];
Simp[ℰ_] := Simp[ℰ, SS[#, Expand] &];
SimpT[ℰ_] := Collect[ℰ, _CU | _QU, SST[#, Expand] &];
Kδ /: Kδ_i_,j_ := If[i === j, 1, 0];
```

Differential polynomials (DP):

```
In[∘]:= DP_(α_→D_x_, β_→D_y_)[P_][λ_] :=
 Total[CoefficientRules[Normal@P, {α, β}] /. ({m_, n_} → c_) ⧴ c ∂_{x,m},{y,n} λ]
```

```
In[∘]:= CF[ℰ_] := ExpandDenominator@
   ExpandNumerator@Together[Expand[ℰ] //. ⅇ^x_ ⅇ^y_ ⧴ ⅇ^(x+y) /. ⅇ^x_ ⧴ ⅇ^CF[x]];
```

```
In[∘]:= Unprotect[SeriesData];
SeriesData /: CF[sd_SeriesData] := MapAt[CF, sd, 3];
SeriesData /: Expand[sd_SeriesData] := MapAt[Expand, sd, 3];
SeriesData /: Simplify[sd_SeriesData] := MapAt[Simplify, sd, 3];
SeriesData /: Together[sd_SeriesData] := MapAt[Together, sd, 3];
SeriesData /: Collect[sd_SeriesData, specs__] := MapAt[Collect[#, specs] &, sd, 3];
Protect[SeriesData];
```

Self-Pair (SP):

In[∘]:=
```
SP{}[P_] := P; SP{ξ_→x_,ps___}[P_] := Expand[P // SP{ps}] /. f_. ξ^d_. ⧴ ∂{x,d} f
```

DeclareAlgebra

In[∘]:=
```
Unprotect[NonCommutativeMultiply]; Attributes[NonCommutativeMultiply] = {};
(NCM = NonCommutativeMultiply)[x_] := x;
NCM[x_, y_, z__] := (x ** y) ** z;
0 ** _ = _ ** 0 = 0;
(x_Plus) ** y_ := (# ** y) & /@ x; x_ ** (y_Plus) := (x ** #) & /@ y;
B[x_, x_] = 0; B[x_, y_] := x ** y - y ** x;
B[x_, y_, e_] := B[x, y, e] = B[x, y];
```

In[•]:=

```
DeclareAlgebra[U_Symbol, opts__Rule] := Module[{gp, sr, g, cp, M, CE, pow, k = 0,
    gs = Generators /. {opts},
    cs = Centrals /. {opts} /. Centrals → {} },
   (#_U = U@#) & /@ gs;
   gp = Alternatives @@ gs; gp = gp | gp_; (* gens *)
   sr = Flatten@Table[{g → ++k, g_i_ → {i, k}}, {g, gs}]; (* sorting → *)
   cp = Alternatives @@ cs; (* cents *)
   SetAttributes[M, HoldRest]; M[0, _] = 0; M[a_, x_] := a x;
   CE[ℰ_] := Collect[ℰ, _U, Expand] /. $trim;
   U_i_ [ℰ_] := ℰ /. {t : cp :→ t_i, u_U :→ (#_i &) /@ u};
   U_i_ [NCM[]] = pow[ℰ_, 0] = U@{} = 1_U = U[];
   B[U@(x_)_i_, U@(y_)_i_] := U_i@B[U@x, U@y];
   B[U@(x_)_i_, U@(y_)_j_] /; i =!= j := 0;
   B[U@y_, U@x_] := CE[-B[U@x, U@y]];
   x_ ** (c_. 1_U) := CE[c x]; (c_. 1_U) ** x_ := CE[c x];
   (a_. U[xx___, x_]) ** (b_. U[y_, yy___]) := If[OrderedQ[{x, y} /. sr],
     CE@M[a b /. $trim, U[xx, x, y, yy]],
     U@xx ** CE@M[a b /. $trim, U@y ** U@x + B[U@x, U@y, $E]] ** U@yy ];
   U@{c_. * (l : gp)^n_, r___} /; FreeQ[c, gp] := CE[c U@Table[l, {n}] ** U@{r}];
   U@{c_. * l : gp, r___} := CE[c U[l] ** U@{r}];
   U@{c_, r___} /; FreeQ[c, gp] := CE[c U@{r}];
   U@{l_Plus, r___} := CE[U@{#, r} & /@ l];
   U@{l_, r___} := U@{Expand[l], r};
   U[ℰ_NonCommutativeMultiply] := U /@ ℰ;
   𝕆_U[specs___, poly_] := Module[{sp, null, vs, us},
     sp = Replace[{specs}, l_List :→ l_null, {1}];
     vs = Join @@ (First /@ sp);
     us = Join @@ (sp /. l_s_ :→ (l /. x_i_ :→ x_s));
     CE[Total[
        CoefficientRules[poly, vs] /. (p_ → c_) :→ c U@(us^p)
       ]] /. x_null :→ x];
   pow[ℰ_, n_] := pow[ℰ, n - 1] ** ℰ;
   𝕊_U[ℰ_, ss___Rule] := CE@Total[
      CoefficientRules[ℰ, First /@ {ss}] /.
       (p_ → c_) :→ c NCM @@ MapThread[pow, {Last /@ {ss}, p}]];
   σ_rs___ [c_. * u_U] := (c /. (t : cp)_j_ :→ t_{j /. {rs}}) U[List @@ (u /. v_j_ :→ v_{j /. {rs}})];
   m_{j_→k_} [c_. * u_U] := CE[((c /. (t : cp)_j → t_k) DeleteCases[u, _j|k]) **
      U @@ Cases[u, w_j :→ w_k] ** U @@ Cases[u, _k]];
   U /: c_. * u_U * v_U := CE[c u ** v];
   S_i_ [c_. * u_U] := CE[((c /. S_i[U, Centrals]) DeleteCases[u, _i]) **
      U_i[NCM @@ Reverse@Cases[u, x_i :→ S@U@x]]];
   Δ_{i_→j_,k_} [c_. * u_U] := CE[((c /. Δ_{i→j,k}[U, Centrals]) DeleteCases[u, _i]) **
      (NCM @@ Cases[u, x_i :→ σ_{1→j,2→k}@Δ@U@x] /. NCM[] → U[])]; ]
```

DeclareMorphism

In[∘]:=
```
DeclareMorphism[m_, U_ → V_, ongs_List, oncs_List: {}] := (
  Replace[ongs,
   {(g_ → img_) ⧴ (m[U[g]] = img), (g_ ⧴ img_) ⧴ (m[U[g]] := img /. $trim)}, {1}];
  m[1_U] = 1_V;
  m[U[g_i_]] := V_i[m[U@g]];
  m[U[vs__]] := NCM @@ (m /@ U /@ {vs});
  m[ℰ_] := Simp[ℰ /. oncs /. u_U ⧴ m[u]] /. $trim; )
```

Meta-Operations

In[∘]:=
```
σ_rs___[ℰ_Plus] := σ_rs /@ ℰ;
m_j_→j_ = Identity; m_j_→k_[0] = 0;
m_j_→k_[ℰ_Plus] := Simp[m_j→k /@ ℰ];
m_is___,i_,j_→k_[ℰ_] := m_j→k@m_is,i→j@ℰ;
S_i_[ℰ_Plus] := Simp[S_i /@ ℰ];
Δ_is___[ℰ_Plus] := Simp[Δ_is /@ ℰ];
```

Implementing CU = $\mathcal{U}(sl_2^{\gamma\epsilon})$

Verify σ and Δ! Also Generalize Δ to $\Delta_{i,j_1,j_2,\ldots}$.

In[∘]:=
```
DeclareAlgebra[CU, Generators → {y, a, x}, Centrals → {t}];
B[a_CU, y_CU] = -γ y_CU; B[x_CU, a_CU] = -γ x_CU;
B[x_CU, y_CU] = 2 ϵ a_CU - t 1_CU;
(S@y_CU = -y_CU; S@a_CU = -a_CU; S@x_CU = -x_CU;)
S_i_[CU, Centrals] = {t_i → -t_i};
Δ@y_CU = CU@y_1 + CU@y_2; Δ@a_CU = CU@a_1 + CU@a_2; Δ@x_CU = CU@x_1 + CU@x_2;
Δ_i_→j_,k_[CU, Centrals] = {t_i → t_j + t_k};
```

Implementing QU = $\mathcal{U}_q(sl_2^{\gamma\epsilon})$

In[∘]:=
```
DeclareAlgebra[QU, Generators → {y, a, x}, Centrals → {t, T}];
B[a_QU, y_QU] = -γ y_QU; B[x_QU, a_QU] = -γ QU@x;
B[x_QU, y_QU] := SS[q_ℏ - 1] QU@{y, x} + 𝕆_QU[{a}, SS[(1 - T ⅇ^(-2 ϵ a ℏ)) / ℏ]];
(S@y_QU := 𝕆_QU[{a, y}, SS[-T^-1 ⅇ^(ℏ ϵ a) y]]; S@a_QU = -a_QU; S@x_QU := 𝕆_QU[{a, x}, SS[-ⅇ^(ℏ ϵ a) x]];)
S_i_[QU, Centrals] = {t_i → -t_i, T_i → T_i^-1};
Δ@y_QU := 𝕆_QU[{y_1, a_1}_1, {y_2}_2, SS[y_1 + T_1 ⅇ^(-ℏ ϵ a_1) y_2]];
Δ@a_QU = QU@a_1 + QU@a_2; Δ@x_QU := 𝕆_QU[{a_1, x_1}_1, {x_2}_2, SS[x_1 + ⅇ^(-ℏ ϵ a_1) x_2]];
Δ_i_→j_,k_[QU, Centrals] = {t_i → t_j + t_k, T_i → T_j T_k};
```

Implementing θ

```
In[∘]:= DeclareMorphism[Cθ, CU → CU, {y → -x_CU, a → -a_CU, x → -y_CU}, {t → -t, T → T^-1}];
DeclareMorphism[Qθ, QU → QU, {y :→ 𝕆_QU[{a, x}, SS[-T^(-1/2) ℯ^(ℏ ϵ a) x]],
   a → -a_QU, x :→ 𝕆_QU[{a, y}, SS[-T^(-1/2) ℯ^(ℏ ϵ a) y]]}, {t → -t, T → T^-1}]
```

The Asymmetric Dequantizator

Following pensieve://People/VanDerVeen/Dequant1.pdf.

```
In[∘]:= A𝔻$f = γ ((Cosh[ℏ (a ϵ + γ ϵ/2 - t/2)] - Cosh[ℏ Sqrt[((t - γ ϵ)/2)^2 + ϵ ω]]) /
      (ℏ ℯ^(ℏ ((a+γ) ϵ-t/2)) Sinh[γ ϵ ℏ/2] (a^2 ϵ + a γ ϵ - a t - ω)));
```

```
In[∘]:= A𝔻$ω = γ CU[y, x] + ϵ CU[a, a] - (t - γ ϵ) CU[a];
```

```
In[∘]:= DeclareMorphism[A𝔻, QU → CU,
  {a → a_CU, x → CU@x, y :→ 𝕊_CU[SS[A𝔻$f], a → a_CU, ω → A𝔻$ω] ** y_CU}]
```

The Symmetric Dequantizator

Following pensieve://People/VanDerVeen/Dequant1.pdf.

```
In[∘]:= S𝔻$g = Sqrt[((2 γ (Cosh[ℏ/2 Sqrt[t^2 + γ^2 ϵ^2 + 4 ϵ ϖ]] - Cosh[(t - ϵ γ - 2 ϵ a)/(2/ℏ)])) /
      (Sinh[γ ϵ ℏ/2] (t (2 a + γ) - 2 a (a + γ) ϵ + 2 ϖ) ℏ))];
```

```
In[∘]:= S𝔻$f = Simplify[ℯ^(ℏ (t/2-ϵ a)) (S𝔻$g /. {a → -a, t → -t})];
```

```
In[∘]:= S𝔻$ϖ = γ CU[y, x] + ϵ CU[a, a] - (t - γ ϵ) CU[a] - t γ 1_CU / 2;
```

```
In[∘]:= DeclareMorphism[S𝔻, QU → CU, {a → a_CU,
   x :→ 𝕊_CU[SS[S𝔻$f], a → a_CU, ϖ → S𝔻$ϖ] ** x_CU,
   y :→ 𝕊_CU[SS[S𝔻$g], a → a_CU, ϖ → S𝔻$ϖ] ** y_CU }]
```

The representation ρ

```
In[∘]:= ρ@y_CU = ρ@y_QU = (0 0
                          ∈ 0); ρ@a_CU = ρ@a_QU = (γ 0
                                                  0 0);
ρ@x_CU = (0 γ
          0 0); ρ@x_QU = (0 (1 - e^(-γ∈ℏ)) / (∈ ℏ)
                          0 0                    );
ρ[e^ℰ_] := MatrixExp[ρ[ℰ]];
ρ[ℰ_] := (ℰ /. T2t /. t → γ ∈ /. (U : CU | QU) [u___] :→ Fold[Dot, (1 0
                                                                  0 1), ρ /@ U /@ {u}])
```

Œ and the logoi Λ

Logoi from Pensieve://Talks/Toulouse-1705/DogmaDemo.nb and from Pensieve://Talks/Sydney-1708/ExtraDetails@@.nb.

```
In[∘]:= Œ_U_ [s1__, Q1_, P1_] Œ_U_ [s2__, Q2_, P2_] ^:= Œ_U[s1, s2, Q1 + Q2, P1 P2];
```

```
In[∘]:= CU@Œ_CU[specs___, Q_, P_] := 𝕆_CU[specs, SS[e^Q P]];
QU@Œ_QU[specs___, Q_, P_] := 𝕆_QU[specs, SS[e^Q P]];
```

```
In[∘]:= c_Integer_(k_Integer) := c + O[∈]^(k+1);
Λ_(U_,k_) [{α_, β_}, {x_, x_}] := Œ_U[{x}, (α + β) x, 1_k];
Λ_(U_,k_) [{ξ_, α_}, {x, a}] := Œ_U[{a, x}, α a + e^(-γ α) ξ x, 1_k];
Λ_(U_,k_) [{α_, η_}, {a, y}] := Œ_U[{y, a}, α a + e^(-γ α) η y, 1_k];
```

Goal. In either U, compute $F = e^{-\eta y} e^{\xi x} e^{\eta y} e^{-\xi x}$. First compute $G = e^{\xi x} y e^{-\xi x}$, a finite sum. Now F satisfies the ODE $\partial_\eta F = \partial_\eta(e^{-\eta y} e^{\eta G}) = -yF + FG$ with initial conditions $F(\eta = 0) = 1$. So we set it up and solve:

```
If[$k > 0, With[{U = CU},
  Module[{G, F, fs, bs, e, b, es, sol},
   G = Echo@Simp[Table[ξ^k / k !, {k, 0, $k + 1}].NestList[Simp[B[x_U , #]] &, y_U, $k + 1]];
   fs = Echo@Flatten@Table[f_(l,i,j,k)[η], {l, 0, $k}, {i, 0, 1}, {j, 0, 1}, {k, 0, 1}];
   F = Echo[fs.(bs = fs /. f_(l_,i_,j_,k_) [η] :→ ∈^l U@{y^i, a^j, x^k})];
   es = Flatten[
     Table[Coefficient[e, b] == 0, {e, {F - 1_U /. η → 0, F ** G - y_U ** F - ∂_η F}}, {b, bs}]];
   sol = Echo@First[F /. DSolve[es, fs, η]];
   Echo[sol /. {e- → 1, U → Times}];
   Collect[sol /. {e- → 1, U → Times}, ∈, Simplify]
  ]]]
```

```
-t ξ CU[] + 2 ∈ ξ CU[a] - γ ∈ ξ^2 CU[x] + CU[y]

{f0,0,0,0[η], f1,0,0,0[η], f1,0,0,1[η], f1,0,1,0[η],
 f1,0,1,1[η], f1,1,0,0[η], f1,1,0,1[η], f1,1,1,0[η], f1,1,1,1[η]}

CU[] f0,0,0,0[η] + ∈ CU[] f1,0,0,0[η] + ∈ CU[x] f1,0,0,1[η] + ∈ CU[a] f1,0,1,0[η] + ∈ CU[a, x] f1,0,1,1[η] +
 ∈ CU[y] f1,1,0,0[η] + ∈ CU[y, x] f1,1,0,1[η] + ∈ CU[y, a] f1,1,1,0[η] + ∈ CU[y, a, x] f1,1,1,1[η]

» e^(-t η ξ) CU[] + 1/2 e^(-t η ξ) t γ ∈ η^2 ξ^2 CU[] + 2 e^(-t η ξ) ∈ η ξ CU[a] - e^(-t η ξ) γ ∈ η ξ^2 CU[x] - e^(-t η ξ) γ ∈ η^2 ξ CU[y]

» 1 + 2 a ∈ η ξ - y γ ∈ η^2 ξ - x γ ∈ η ξ^2 + 1/2 t γ ∈ η^2 ξ^2

1 + 1/2 ∈ η ξ (4 a + γ (-2 y η - 2 x ξ + t η ξ))
```

```
In[•]:= Λ_{U_,kk_}[{ξ1_, η1_}, {x, y}] := Λ_{U,kk}[{ξ1, η1}, {x, y}] =
    Block[{$k = kk, $p = kk}, Module[{ξ, η, G, F, fs, f, bs, e, b, es},
      G = Simp[Table[ξ^k / k!, {k, 0, $k + 1}].NestList[Simp[B[x_U, #]] &, y_U, $k + 1]];
      fs = Flatten@Table[f_{l,i,j,k}[η], {l, 0, $k}, {i, 0, l}, {j, 0, l}, {k, 0, l}];
      F = fs.(bs = fs /. f_{l_,i_,j_,k_}[η] :→ e^l U@{y^i, a^j, x^k});
      es = Flatten[
        Table[Coefficient[e, b] == 0, {e, {F - 1_U /. η → 0, F ** G - y_U ** F - ∂_η F}}, {b, bs}]];
      F = F /. DSolve[es, fs, η][[1]];
      Œ_U[{y, a, x},
        ξ x + η y + (U /. {CU → -t η ξ, QU → η ξ (1 - T) / ℏ}),
        F + O_$k /. {e- → 1, U → Times}
      ] /. {ξ → ξ1, η → η1}]];
```

```
In[•]:= Simp[Œ_U_[specs___, Q_, P_]] := Œ_U[specs, CF[Q], CF[P]];
```

```
In[•]:= Λ_{U_,k_}[{υ1_, ω1_, δ_}, {u_, w_}] := Simp@Module[{υ, ω, yax, q, p, Q, d},
     {yax, q, p} = List @@ Λ_{U,k}[{υ, ω}, {u, w}];
     Œ_U[yax, Q = (υ u + ω w + δ u w + d υ ω) / (1 - d δ),
       Expand[(1 - d δ)^-1 e^-Q DP_{υ→D_u, ω→D_w}[p][e^Q]] + O_k] /. {d → ∂_{υ,ω} q} /. {υ → υ1, ω → ω1}];
```

Reorderings with Rord

```
In[•]:= Rord_{u_i_, w_j_ → k_}[Œ_U_[L___, {l___, u_i_, w_j_, r___}_s_, R___, Q_, P_]] :=
   Simp@Module[{υ, ω, δ, Λ1, yax, q, p, kk = P[[5]], δ1 = ∂_{u_i,w_j} Q},
     {yax, q, p} = Echo[List @@ If[δ1 === 0, Λ_{U,kk}[{υ, ω}, {u, w}],
          Λ_{U,kk}[{υ, ω, δ}, {u, w}]] /. {y → y_k, a → a_k, x → x_k, t → t_s, T → T_s}];
     Œ_U[L, {l, Sequence @@ yax, r}_s, R, q + (Q /. u_i | w_j → 0), e^-q DP_{u_i→D_υ, w_j→D_ω}[P][p e^q]] /.
      {υ → ∂_{u_i} Q /. w_j → 0, ω → ∂_{w_j} Q /. u_i → 0, δ → δ1}];
```

In[∘]:=
```
Rord_{u_i_,w_j_→k_}[Œ_U_[L___, {l___, u_i_, w_j_, r___}_s_, R___, Q_, P_]] :=
  Simp@Module[{υ, ω, δ, Λ1, yax, q, p, n, kk = P[[5]], δ1 = ∂_{u_i,w_j} Q},
    {yax, q, p} = List @@ If[δ1 === 0, Λ_{U,kk}[{υ, ω}, {u, w}], Λ_{U,kk}[{υ, ω, δ}, {u, w}]] /.
      {y → y_n, a → a_n, x → x_n, t → t_s, T → T_s};
    (*Echo@{{{u_i,υ},{w_j,ω}},P,p e^q};*)
    Œ_U[L, {l, Sequence @@ yax, r}_s, R, q + (Q /. u_i | w_j → 0), e^-q SP_{u_i→υ,w_j→ω}[P p e^q]] /.
     {n → k, υ → ∂_{u_i} Q /. w_j → 0, ω → ∂_{w_j} Q /. u_i → 0, δ → δ1}];
```

Canonical ordering with Cord

In[∘]:=
```
Cord[Œ_U_[L___, {l___, u_i_, w_j_, r___}_s_, R___, Q_, P_]] /;
   OrderedQ[{w, u} /. {y → 1, a → 2, x → 3}] :=
  ((*Echo@{u_i,w_j};*)Cord[Rord_{u_i,w_j→Unique[]}[Œ_U[L, {l, u_i, w_j, r}_s, R, Q, P]]]);
Cord[Œ_U_[specs__, Q_, P_]] := Œ_U[Sequence @@ Sort@{specs}, Q, P] /.
  Flatten[{specs} /. {yax__}_s_ :> ({yax} /. u_i_ :> (u_i → u_s))]
```

Stitching Œ’s.

In[∘]:=
```
m_{j_→k_}[Œ_U_[specs__, Q_, P_]] := Cord[Œ_U[Sequence @@ Append[DeleteCases[{specs}, {__}_{j|k}],
      Flatten[{Cases[{specs}, {us__}_j :> {us}], Cases[{specs}, {us__}_k :> {us}]}]_k],
    Q, P] /. {t_j → t_k, T_j → T_k}]
```

In[∘]:=
```
Œ_U_[sp1__, Q1_, P1_] ≡ Œ_U_[sp2__, Q2_, P2_] :=
 Sort[{sp1}] == Sort[{sp2}] ∧ Simplify[Q1 == Q2] ∧ Simplify[Normal[P1 - P2] == 0]
```

R in QU.

The Faddeev-Quesne formula:

In[∘]:=
```
e_{q_,k_}[x_] := e^(Sum_{j=1}^{k+1} ((1 - q)^j x^j)/(j (1 - q^j))); e_{q_}[x_] := e_{q,$k}[x]
```

In[∘]:=
```
QU[R_{i_,j_}] := O_QU[{y_1, a_1}_i, {a_2, x_2}_j, SS[e^{ħ b_1 a_2} e_{q_ħ}[ħ y_1 x_2] /. b_1 → γ^-1 (ϵ a_1 - t_i)]];
QU[R^-1_{i_,j_}] := S_j@QU[R_{i,j}];
```

R in $Œ_{QU}$.

In[∘]:=
```
Œ_{QU,k_}[R_{i_,j_}] := Œ_QU[{y_i, a_i, x_i}_i, {y_j, a_j, x_j}_j, -ħ γ^-1 t_i a_j + ħ y_i x_j,
  Series[e^{ħ γ^-1 t_i a_j - ħ y_i x_j} (e^{ħ b_i a_j} e_{q_ħ,k}[ħ y_i x_j] /. b_i → γ^-1 (ϵ a_i - t_i)), {ϵ, 0, k}]]
```

The morphism $Œ_{U,k}$.

```
In[◦]:= Œ_{U_,k_}[a_ * b_] := Œ_{U,k}[a] Œ_{U,k}[b];
        Œ_{U_,k_}[m_{is__}[a_]] := m_{is}[Œ_{U,k}[a]];
```

Exponentials as needed.

Task. Define $\text{Exp}_{U_i,k}[\xi, P]$ which computes $e^{\xi\,\mathbb{O}(P)}$ to ϵ^k in the algebra U_i, where ξ is a scalar, X is x_i or y_i, and P is an ϵ-dependent near-docile element, giving the answer in Œ-form. Should satisfy $U@\text{Exp}_{U_i,k}[\xi, P] == \mathbb{S}_U[e^{\xi x},\ x \to \mathbb{O}(P)]$.

Methodology. If $P_0 := P_{\epsilon=0}$ and $e^{\xi\,\mathbb{O}(P)} = \mathbb{O}(e^{\xi P_0} F(\xi))$, then $F(\xi = 0) = 1$ and we have:

$\mathbb{O}(e^{\xi P_0}(P_0 F(\xi) + \partial_\xi F) = \mathbb{O}(\partial_\xi e^{\xi P_0} F(\xi)) = \partial_\xi \mathbb{O}(e^{\xi P_0} F(\xi)) = \partial_\xi e^{\xi\,\mathbb{O}(P)} = e^{\xi\,\mathbb{O}(P)}\,\mathbb{O}(P) = \mathbb{O}(e^{\xi P_0} F(\xi))\,\mathbb{O}(P)$.

This is an ODE for F. Setting inductively $F_k = F_{k-1} + \epsilon^k \varphi$ we find that $F_0 = 1$ and solve for φ.

```
In[◦]:= (* Bug: The first line is valid only if 𝕆(e^{P0}) == e^{𝕆(P0)}. *)
        (* Bug: ξ must be a symbol. *)
        Exp_{U_i_,0}[ξ_, P_] := Œ_U[{y_i, a_i, x_i}_i, Normal@P /. ϵ → 0, 1 + O_0];
        Exp_{U_i_,k_}[ξ_, P_] := Module[{yax = {y_i, a_i, x_i}, P0, φ, φs, F, j, rhs, at0, atξ},
          P0 = Normal@P /. ϵ → 0;
          φs =
           Flatten@Table[φ_{j1,j2,j3}[ξ], {j2, 0, k}, {j1, 0, 2 k + 1 - j2}, {j3, 0, 2 k + 1 - j2 - j1}];
          F = Normal@Last@Exp_{U_i,k-1}[ξ, P] + ϵ^k φs.(φs /. φ_{js__}[ξ] :→ Times @@ yax^{js});
          rhs = Normal@Last@m_{i,j→i}[Œ_U[yax_i, ξ P0, F + O_k] m_{i→j}@Œ_U[{y_i, a_i, x_i}_i, 0, P + O_k]];
          at0 = (# == 0) & /@ Flatten@CoefficientList[F - 1 /. ξ → 0, yax];
          atξ = (# == 0) & /@ Flatten@CoefficientList[(∂_ξ F) + P0 F - rhs, yax];
          Œ_U[yax_i, ξ P0, F + O_k] /. DSolve[And @@ (at0 ⋃ atξ), φs, ξ][[1]] ]
```

Zip and Bind

```
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];
```

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

```
In[◦]:= Zip_{}[P_] := P; Zip_{ζ_,ζs___}[P_] := (Expand[P // Zip_{ζs}] /. f_. ζ^{d_.} :→ ∂_{ζ*,d} f) /. ζ* → 0
```

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

```
𝔼 /: QZip_{ζs_List}@𝔼[L_, Q_, P_] := 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[Kδ_{z,ζ*} - ∂_{z,ζ}Q, {ζ, ζs}, {z, zs}];
   zrule = Thread[zs → qt.(zs + ys)];
   Q2 = (Q1 = c + ηs.zs /. zrule) /. Alternatives @@ zs → 0;
   CF /@ 𝔼[L, Q2, Det[qt] ℯ^{-Q2} Zip_{ζs}[ℯ^{Q1} (P /. zrule)]] ];
```

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 t and α and the ζ’s are τ and a.

```
𝔼 /: LZip_{ζs_List}@𝔼[L_, Q_, P_] := Module[{ζ, z, zs, c, ys, ηs, lt, zrule, L1, L2, Q1, Q2},
   zs = Table[ζ*, {ζ, ζs}];
   c = L /. Alternatives @@ (ζs ⋃ zs) → 0;
   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)];
   L2 = (L1 = c + ηs.zs /. zrule) /. Alternatives @@ zs → 0;
   Q2 = (Q1 = Q /. T2t /. zrule) /. Alternatives @@ zs → 0;
   CF /@ 𝔼[L2, Q2, Det[lt] ℯ^{-L2-Q2} Zip_{ζs}[ℯ^{L1+Q1} (P /. T2t /. zrule)]] //. t2T ];
```

```
In[∘]:= Bind_{}[L_, R_] := L R;
Bind_{is__}[L_𝔼, R_𝔼] := Module[{n},
   Times[
      L /. Table[(v : T | t | a | x | y)_i → v_{n@i}, {i, {is}}],
      R /. Table[(v : τ | α | ξ | η)_i → v_{n@i}, {i, {is}}]
   ] // LZip_{Flatten@Table[{τ_{n@i},a_{n@i}},{i,{is}}]} // QZip_{Flatten@Table[{ξ_{n@i},y_{n@i}},{i,{is}}]} ];
B_{L_List} := Bind_L; B_{is___} := Bind_{is};
Bind[ℰ_𝔼] := ℰ;
Bind[Ls__, ζs_List, R_] := Bind_{ζs}[Bind[Ls], R];
```

Tensorial Representations

```
In[∘]:= tη = t𝟙 = 𝔼[0, 0, 1 + 0_{$k}];
```

```
In[∘]:= m[U_, kk_] := m[U, kk] = Module[{OE},
     OE = Simplify /@
       m_{1,2→1}@ℭ_U[{y1, a1, x1}_1, {y2, a2, x2}_2, η1 y1 + α1 a1 + ξ1 x1 + η2 y2 + α2 a2 + ξ2 x2, 1 + 0_{kk}];
     𝔼[t1 (τ1 + τ2) + (OE⟦2⟧ /. (ξ | η)_{1|2} → 0), OE⟦2⟧ /. a1 → 0, OE⟦3⟧] ];
tm_{i_,j_→k_} :=
  m[$U, $k] /. {(v : τ | η | α | ξ)_1 → v_i, (v : τ | η | α | ξ)_2 → v_j, (v : t | T | y | a | x)_1 → v_k};
```

In[∘]:= $\mathbf{tm}_{1,2\to 3}$

Out[∘]= $\mathbb{E}\Big[a_3\ (\alpha_1+\alpha_2)+t_3\ (\tau_1+\tau_2),\ y_3\ \left(\eta_1+e^{-\gamma\,\alpha_1}\,\eta_2\right)-\frac{(-1+T_3)\ \eta_2\ \xi_1}{\hbar}+x_3\ \left(e^{-\gamma\,\alpha_2}\ \xi_1+\xi_2\right),$
$1+\frac{1}{4\,\hbar}\ e^{-\gamma\ (\alpha_1+\alpha_2)}\ \eta_2\ \xi_1\ \Big(8\ e^{\gamma\ (\alpha_1+\alpha_2)}\ \hbar\ a_3\ T_3+e^{\gamma\,\alpha_2}\ \gamma\ (-1+3\ T_3)\ \eta_2\ \left(-2\ \hbar\ y_3+e^{\gamma\,\alpha_1}\ (-1+T_3)\ \xi_1\right)+$
$2\ \gamma\ \hbar\ x_3\ \left(2\ \hbar\ y_3-e^{\gamma\,\alpha_1}\ (-1+3\ T_3)\ \xi_1\right)\Big)\ \epsilon+O[\epsilon]^2\Big]$

In[∘]:=
```
S[U_, kk_] := S[U, kk] = Module[{OE},
    OE = m_{3,2,1→1}[Exp_{U_1,$k}[η, S_1[U[y_1]] /. U → Times]
        Exp_{U_2,$k}[α, S_2[U[a_2]] /. U → Times] Exp_{U_3,$k}[ξ, S_3[U[x_3]] /. U → Times]];
    𝔼[-t_1 τ_1 + (OE⟦2⟧ /. ξ | η → 0), OE⟦2⟧ /. a_1 → 0, OE⟦3⟧] /. {η → η_1, α → α_1, ξ → ξ_1}];
tS_i_ := S[$U, $k] /. {(v : τ | η | α | ξ)_1 → v_i, (v : t | T | y | a | x)_1 → v_i};
```

In[∘]:= $\mathbf{tS}_1$

Out[∘]= $\mathbb{E}\Big[-a_1\ \alpha_1-t_1\ \tau_1,\ \frac{1}{\hbar\ T_1}\ \left(-e^{\gamma\,\alpha_1}\ \hbar\ y_1\ \eta_1-e^{\gamma\,\alpha_1}\ \hbar\ T_1\ x_1\ \xi_1+e^{\gamma\,\alpha_1}\ \eta_1\ \xi_1-e^{\gamma\,\alpha_1}\ T_1\ \eta_1\ \xi_1\right),$
$1+\frac{1}{4\ \hbar\ T_1^2}\ \Big(4\ e^{\gamma\,\alpha_1}\ \gamma\ \hbar^2\ T_1\ y_1\ \eta_1-4\ e^{\gamma\,\alpha_1}\ \hbar^2\ a_1\ T_1\ y_1\ \eta_1-2\ e^{2\,\gamma\,\alpha_1}\ \gamma\ \hbar^2\ y_1^2\ \eta_1^2-4\ e^{\gamma\,\alpha_1}\ \hbar^2\ a_1\ T_1^2\ x_1\ \xi_1-$
$4\ e^{\gamma\,\alpha_1}\ \gamma\ \hbar\ T_1\ \eta_1\ \xi_1+8\ e^{\gamma\,\alpha_1}\ \hbar\ a_1\ T_1\ \eta_1\ \xi_1+4\ e^{\gamma\,\alpha_1}\ \gamma\ \hbar\ T_1^2\ \eta_1\ \xi_1-4\ e^{2\,\gamma\,\alpha_1}\ \gamma\ \hbar^2\ T_1\ x_1\ y_1\ \eta_1\ \xi_1+$
$6\ e^{2\,\gamma\,\alpha_1}\ \gamma\ \hbar\ y_1\ \eta_1^2\ \xi_1-2\ e^{2\,\gamma\,\alpha_1}\ \gamma\ \hbar\ T_1\ y_1\ \eta_1^2\ \xi_1-2\ e^{2\,\gamma\,\alpha_1}\ \gamma\ \hbar^2\ T_1^2\ x_1^2\ \xi_1^2+6\ e^{2\,\gamma\,\alpha_1}\ \gamma\ \hbar\ T_1\ x_1\ \eta_1\ \xi_1^2-$
$2\ e^{2\,\gamma\,\alpha_1}\ \gamma\ \hbar\ T_1^2\ x_1\ \eta_1\ \xi_1^2-3\ e^{2\,\gamma\,\alpha_1}\ \gamma\ \eta_1^2\ \xi_1^2+4\ e^{2\,\gamma\,\alpha_1}\ \gamma\ T_1\ \eta_1^2\ \xi_1^2-e^{2\,\gamma\,\alpha_1}\ \gamma\ T_1^2\ \eta_1^2\ \xi_1^2\Big)\ \epsilon+O[\epsilon]^2\Big]$

In[∘]:=
```
Δ[U_, kk_] := Δ[U, kk] = Module[{OE},
    OE = Block[{$k = kk, $p = kk + 1}, m_{1,3,5→1}@m_{2,4,6→2}@Times[
          Prepend[{y_2}_2]@Exp_{U_1,$k}[η, Δ_{1→1,2}[U[y_1]] /. U → Times],
          Prepend[{a_4}_4]@Exp_{U_3,$k}[α, Δ_{3→3,4}[U[a_3]] /. U → Times],
          Prepend[{x_6}_6]@Exp_{U_5,$k}[ξ, Δ_{5→5,6}[U[x_5]] /. U → Times]
        ] /. {η → η_1, α → α_1, ξ → ξ_1}];
    𝔼[τ_1 (t_1 + t_2) + α_1 (a_1 + a_2), OE⟦3⟧ /. α_1 → 0, OE⟦4⟧]];
tΔ_{i_→j_,k_} :=
  Δ[$U, $k] /. {(v : τ | η | α | ξ)_1 → v_i, (v : t | T | y | a | x)_1 → v_j, (v : t | T | y | a | x)_2 → v_k};
```

In[∘]:= $\mathbf{t\Delta}_{1\to 1,2}$

Out[∘]= $\mathbb{E}\Big[(a_1+a_2)\ \alpha_1+(t_1+t_2)\ \tau_1,\ y_1\ \eta_1+T_1\ y_2\ \eta_1+x_1\ \xi_1+x_2\ \xi_1,$
$1+\frac{1}{2}\ \left(-2\ \hbar\ a_1\ T_1\ y_2\ \eta_1+\gamma\ \hbar\ T_1\ y_1\ y_2\ \eta_1^2-2\ \hbar\ a_1\ x_2\ \xi_1+\gamma\ \hbar\ x_1\ x_2\ \xi_1^2\right)\ \epsilon+O[\epsilon]^2\Big]$

In[∘]:=
```
R[QU, kk_] := R[QU, kk] = Module[{OE},
    OE = Simplify /@ Œ_{QU,kk}@R_{1,2};
    𝔼[-ℏ a_2 t_1 / γ, ℏ x_2 y_1, Last@OE]];
tR_{i_,j_} := R[$U, $k] /. {(v : t | T | y | a | x)_1 → v_i, (v : t | T | y | a | x)_2 → v_j};
overline(tR)_{i_,j_} := overline(tR)_{i,j} = tR_{i,j} ~B_j~ tS_j;
```

In[∘]:= $\{tR_{1,2},\ \overline{tR}_{1,2}\}$

Out[∘]= $\left\{\mathbb{E}\left[-\frac{\hbar\, a_2\, t_1}{\gamma},\ \hbar\, x_2\, y_1,\ 1+\left(\frac{\hbar\, a_1\, a_2}{\gamma}-\frac{1}{4}\gamma\,\hbar^3\, x_2^2\, y_1^2\right)\epsilon+O[\epsilon]^2\right],\ \mathbb{E}\left[\frac{\hbar\, a_2\, t_1}{\gamma},\ -\frac{\hbar\, x_2\, y_1}{T_1},\ 1-\frac{1}{4\left(\gamma\, T_1^2\right)}\left(\hbar\left(4\, a_1\, T_1\, (a_2\, T_1+\gamma\,\hbar\, x_2\, y_1)+\gamma\,\hbar\, x_2\, y_1\left(4\, a_2\, T_1+3\,\gamma\,\hbar\, x_2\, y_1\right)\right)\right)\epsilon+O[\epsilon]^2\right]\right\}$

tC is the counterclockwise spinner; $\overline{tC}$ is its inverse.

In[∘]:=
```
tC_i_ := 𝔼[0, 0, T_i^(1/2) ⅇ^(-ϵ a_i ℏ) + O_$k];
tC̄_i_ := 𝔼[0, 0, T_i^(-1/2) ⅇ^(ϵ a_i ℏ) + O_$k];
```

In[∘]:= `Block[{$k = 3}, {tC_1, tC̄_2}]`

Out[∘]= $\left\{\mathbb{E}\left[-\frac{t_1}{2},\ 0,\ 1+a_1\,\epsilon+\frac{1}{2}a_1^2\,\epsilon^2+\frac{1}{6}a_1^3\,\epsilon^3+O[\epsilon]^4\right],\ \mathbb{E}\left[\frac{t_2}{2},\ 0,\ 1-a_2\,\epsilon+\frac{1}{2}a_2^2\,\epsilon^2-\frac{1}{6}a_2^3\,\epsilon^3+O[\epsilon]^4\right]\right\}$

In[∘]:=
```
Kink[QU, kk_] := Kink[QU, kk] = Block[{$k = kk}, (tR_{1,3} tC̄_2) ~B_{1,2}~ tm_{1,2→1} ~B_{1,3}~ tm_{1,3→1}];
tKink_i_ := Kink[$U, $k] /. {(v : t | T | y | a | x)_1 → v_i};
K̄ink[QU, kk_] := K̄ink[QU, kk] = Block[{$k = kk}, (tR̄_{1,3} tC_2) ~B_{1,2}~ tm_{1,2→1} ~B_{1,3}~ tm_{1,3→1}];
tK̄ink_i_ := K̄ink[$U, $k] /. {(v : t | T | y | a | x)_1 → v_i}
```

Alternative Algorithms

In[∘]:=
```
λ_{alt,k_}[CU] := If[k == 0, 1, Module[{eq, d, b, c, so},
    eq = ρ@ⅇ^(ξ x_CU).ρ@ⅇ^(η y_CU) == ρ@ⅇ^(d y_CU).ρ@ⅇ^(c (t 1_CU - 2 ϵ a_CU)).ρ@ⅇ^(b x_CU);
    {so} = Solve[Thread[Flatten /@ eq], {d, b, c}] /. C@1 → 0;
    Series[ⅇ^(-η y-ξ x+η ξ t+c t + d y - 2 ϵ c a + b x) /. so, {ϵ, 0, k}]]];
```