Cheat Sheet sl_2-Portfolio (an implementation of the sl_2 portfolio)

http://drorbn.net/AcademicPensieve/Projects/SL2Portfolio/
modified 20/4/18, 04:07

$\mathcal{U}_{\gamma\epsilon;\hbar}$ conventions. "consolidate"

$q=e^{\hbar\gamma\epsilon}$, $H=\langle a,x\rangle/([a,x]=\gamma x)$ with

$$A=e^{-\hbar\epsilon a},\quad xA=qAx,\quad S_H(a,A,x)=(-a,A^{-1},-A^{-1}x),$$
$$\Delta_H(a,A,x)=(a_1+a_2,A_1A_2,x_1+A_1x_2)$$

and dual $H^*=\langle b,y\rangle/([b,y]=-\epsilon y)$ with

$$B=e^{-\hbar\gamma b},\quad By=qyB,\quad S_{H^*}(b,B,y)=(-b,B^{-1},-yB^{-1}),$$
$$\Delta_{H^*}(b,B,y)=(b_1+b_2,B_1B_2,y_1B_2+y_2).$$

Pairing by $(a,x)^*=\hbar(b,y)$ ($\Rightarrow\langle B,A\rangle=q$) making $\langle y^lb^i,a^jx^k\rangle=\delta_{ij}\delta_{kl}\hbar^{-(j+k)}j![k]_q!$ so $R=\sum\frac{\hbar^{j+k}y^kb^j\otimes a^jx^k}{j![k]_q!}$. Then $\mathcal{U}=H^{*cop}\otimes H$ with $(\phi f)(\psi g)=\langle\psi_1S^{-1}f_3\rangle\langle\psi_3,f_1\rangle(\phi\psi_2)(f_2g)$ and

$$S(y,b,a,x)=(-B^{-1}y,-b,-a,-A^{-1}x),$$
$$\Delta(y,b,a,x)=(y_1+y_2B_1,b_1+b_2,a_1+a_2,x_1+A_1x_2).$$

With the central $t:=\epsilon a-\gamma b$, $T:=e^{\hbar t}=A^{-1}B$ get

$$[a,y]=-\gamma y,\quad [b,x]=\epsilon x,\quad xy-qyx=(1-TA^2)/\hbar.$$

Cartan: $\theta(y,b,a,x)=(-B^{-1}T^{1/2}x,-b,-a,-A^{-1}T^{-1/2}y)$. (Suggesting that it may be better to redefine $y\to y'=\theta x=A^{-1}T^{-1/2}y$.)

At $\epsilon=0$, $\mathcal{U}_{\hbar;\gamma0}=\langle t,y,a,x\rangle/([t,\cdot]=0,[a,x]=\gamma x,[a,y]=-\gamma y,[x,y]=(1-T)/\hbar)$ with $\Delta(t,y,a,x)=(t_1+t_2,y_1+T_1y_2,a_1+a_2,x_1+x_2)$ and $\theta(y,b,a,x)=(-T^{-1/2}x,-b,-a,-T^{-1/2}y)$.

Working Hypothesis. $(\hbar,t,y,a,x)$ makes a PBW basis.

Casimir. $\omega=\gamma yx+\epsilon a^2-(t-\gamma\epsilon)a$, satisfies.... Roland in MixOrder.pdf: Centrals are valuable; perhaps we should write everything in CU/QU as $(x\vee y)\cdot$(functions of a)$\cdot$(centrals).

Scaling with deg: $\{\gamma,\epsilon,a,b,x,y\}\to 1$, $\{\hbar\}\to -2$, $\{t\}\to 2$, $\{\omega\}\to 3$.

Verification (as in Projects/PPSA/Verification.nb).

```
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);
```

```
$p = 2; $k = 1; $U = QU; $E := {$k, $p};
$trim := {ℏ^p_. /; p > $p → 0, ϵ^k_. /; k > $k → 0};
SetAttributes[{SS, SST}, HoldAll];
q_ℏ = e^(γ ϵ ℏ);
T2t = {T_i_^p_. → e^(p ℏ t_i), T^p_. → e^(p ℏ t)}; (* "T to lower t" *)
t2T = {e^(c_. t_i_ + b_.) :→ T_i^(c/ℏ) e^b, e^(c_. t + b_.) :→ T^(c/ℏ) e^b, e^ℰ_ :→ e^(Expand@ℰ)};
(* "t to upper T" *)
SS[ℰ_, op_] := Collect[
   Normal@Series[If[$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];
c_Integer_k_Integer := c + O[ϵ]^(k+1);
```

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

```
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];
```

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

```
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};
```

```
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};
```

```
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}]
```

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

```
A𝔻$ω = γ CU[y, x] + ϵ CU[a, a] - (t - γ ϵ) CU[a];
DeclareMorphism[A𝔻, QU → CU,
  {a → a_CU, x → CU@x,
   y :→ 𝕊_CU[SS[A𝔻$f], a → a_CU, ω → A𝔻$ω] ** y_CU}]
```

```
S𝔻$g = Sqrt[ (2 γ (Cosh[ℏ/2 Sqrt[t^2 + γ^2 ϵ^2 + 4 ϵ ϖ]] - Cosh[(t-ϵγ-2ϵa)/(2/ℏ)])) /
             (Sinh[γϵℏ/2] (t (2 a + γ) - 2 a (a + γ) ϵ + 2 ϖ) ℏ) ];
S𝔻$f = Simplify[e^(ℏ (t/2 - ϵ a)) (S𝔻$g /. {a → -a, t → -t})];
```

```
S𝔻$ϖ = γ CU[y, x] + ϵ CU[a, a] - (t - γ ϵ) CU[a] - t γ 1_CU / 2;
DeclareMorphism[S𝔻, QU → CU, {a → a_CU,
   x :→ 𝕊_CU[SS[S𝔻$f], a → a_CU, ϖ → S𝔻$ϖ] ** x_CU,
   y :→ 𝕊_CU[SS[S𝔻$g], a → a_CU, ϖ → S𝔻$ϖ] ** y_CU }]
```

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

Fear Not. If $G=\mathbb{e}^{\xi x}y\mathbb{e}^{-\xi x}$ then $F=\mathbb{e}^{-\eta y}\mathbb{e}^{\xi x}\mathbb{e}^{\eta y}\mathbb{e}^{-\xi x}=\mathbb{e}^{-\eta y}\mathbb{e}^{\eta G}$ satisfies $\partial_\eta F=-yF+FG$ and $F_{\eta=0}=1$:

```
SW_xy[U_, kk_] :=
  SW_xy[U, kk] = Block[{$U = U, $k = kk, $p = kk},
    Module[{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, $k}, {i, 0, l},
        {j, 0, l}, {k, 0, l}];
     F = fs.(bs = fs /. f_{l_,i_,j_,k_}[η] ⧴ ϵ^l U@{y^i, a^j, x^k});
     es = Flatten[Table[Coefficient[e, b] == 0,
        {e, {F - 1_U /. η → 0, F ** G - y_U ** F - ∂_η F}},
        {b, bs}]];
     F = F /. DSolve[es, fs, η][[1]];
     𝔼[0,
       ξ x + η y + (U /. {CU → -t η ξ, QU → η ξ (1 - T) / ℏ}),
       F + 0_$k /. {e- → 1, U → Times}
      ] /. (v : η | ξ | t | T | y | a | x) → v_1
    ]];
tSW_{xy,i_,j_→k_} :=
  SW_xy[$U, $k] /. {ξ_1 → ξ_i, η_1 → η_j, (v : t | T | y | a | x)_1 → v_k};
tSW_{xa,i_,j_→k_} := 𝔼[α_j a_k, e^(-γ α_j) ξ_i x_k, 1];
tSW_{ay,i_,j_→k_} := 𝔼[α_i a_k, e^(-γ α_i) η_j y_k, 1];
e_{q_,k_}[x_] := e^(Σ_{j=1}^{k+1} (1 - q)^j x^j / (j (1 - q^j))); e_q_[x_] := e_{q,$k}[x]
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}];
```

Task. Define $\text{Exp}_{U_i,k}[\xi,P]$ which computes $\mathbb{e}^{\xi\,\mathbb{O}(P)}$ 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 $\mathbb{E}$-form. Should satisfy $U@\text{Exp}_{U_i,k}[\xi,P] == \mathbb{S}_U[\mathbb{e}^{\xi x},\ x\to\mathbb{O}(P)]$.

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

$$\mathbb{O}\big(\mathbb{e}^{\xi P_0}(P_0F(\xi)+\partial_\xi F)=\mathbb{O}\big(\partial_\xi\mathbb{e}^{\xi P_0}F(\xi)\big)= \qquad .$$
$$\partial_\xi\mathbb{O}\big(\mathbb{e}^{\xi P_0}F(\xi)\big)=\partial_\xi\mathbb{e}^{\xi\,\mathbb{O}(P)}=\mathbb{e}^{\xi\,\mathbb{O}(P)}\,\mathbb{O}(P)=\mathbb{O}\big(\mathbb{e}^{\xi P_0}F(\xi)\big)\,\mathbb{O}(P)$$

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.

```
(* 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_] := Module[{LQ = Normal@P /. ϵ → 0},
   𝔼[ξ LQ /. (x | y)_i → 0, ξ LQ /. (t | a)_i → 0, 1]];
Exp_{U_i_,k_}[ξ_, P_] := Block[{$U = U, $k = k},
  Module[{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 @@ {y_i, a_i, x_i}^{js});
   rhs =
    Normal@
     Last@
      m_{i,j→i}[𝔼[ξ P0 /. (x | y)_i → 0, ξ P0 /. (t | a)_i → 0, F + 0_k]
       m_{i→j}@𝔼[0, 0, P + 0_k]];
   at0 = (# == 0) & /@
     Flatten@CoefficientList[F - 1 /. ξ → 0, {y_i, a_i, x_i}];
   atξ = (# == 0) & /@
     Flatten@CoefficientList[(∂_ξ F) + P0 F - rhs,
      {y_i, a_i, x_i}];
   𝔼[ξ P0 /. (x | y)_i → 0, ξ P0 /. (t | a)_i → 0, F + 0_k] /.
    DSolve[And @@ (at0 ⋃ atξ), φs, ξ][[1]] ]]
```

To do. • Consider renormalizing x and y. • Can everything be done at $\hbar=1$ defining a filtration by other means? That ought to be possible as the end results depend on t/T and not on $\hbar$. • Bound the degrees of the logoi! • $r=\theta r$? • θ is a global symmetry. Can it be "gauged"? • Global $\eta\to\psi$?

Alternative Algorithms.

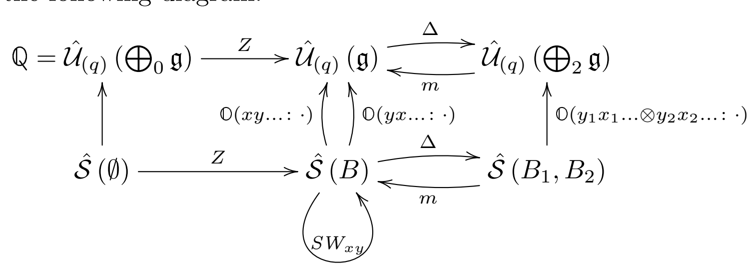
```
λ_{alt,k_}[CU] := If[k == 0, 1, Module[{eq, d, b, c, so},
   eq = ρ@e^(ξ x_CU).ρ@e^(η y_CU) == ρ@e^(d y_CU).ρ@e^(c (t 1_CU - 2 ϵ a_CU)).ρ@e^(b x_CU);
   {so} = Solve[Thread[Flatten /@ eq], {d, b, c}] /.
     C@1 → 0;
   Series[e^(-η y - ξ x + η ξ t + c t + d y - 2 ϵ c a + b x) /. so, {ϵ, 0, k}]]];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

Asides.

```
Series[(1 - T e^(-2 ϵ a ℏ)) / ℏ, {a, 0, 3}]
```

$$\frac{1-\mathrm{T}}{\hbar}+2\,\mathrm{T}\,\epsilon\,\mathrm{a}-2\left(\mathrm{T}\,\epsilon^2\,\hbar\right)\mathrm{a}^2+\frac{4}{3}\,\mathrm{T}\,\epsilon^3\,\hbar^2\,\mathrm{a}^3+\mathrm{O}[\mathrm{a}]^4$$

$\mathcal{GDO}$-Categories. Given $\mathfrak{g}$ with basis $B=\{x,y,\ldots\}$, consider the following diagram:



Hence Z, SW_{xy}, m, Δ, (and likewise S and θ) are morphisms in the *completion* of the monoidal category $\mathcal{F}$ whose objects are finite sets B and whose morphism are $\text{mor}_{\mathcal{F}}(B,B'):=\text{Hom}_{\mathbb{Q}}(\mathcal{S}(B)\to\mathcal{S}(B'))=\mathcal{S}(B^*,B')$ (by convention, $x^*=\xi$, $y^*=\eta$, etc.). Ergo we need to *consolidate* (at least parts of) said completion.

Aside. "Consolidate" means "give a finite name to an infinite object, and figure out how to sufficiently manipulate such finite names". E.g., solving $f''=-f$ we encounter and set

$\sum \frac{(-1)^k x^{2k}}{(2k)!} \rightsquigarrow \cos x$, $\sum \frac{(-1)^k x^{2k+1}}{(2k+1)!} \rightsquigarrow \sin x$, and then $\cos^2 x + \sin^2 x = 1$ and $\sin(x+y) = \sin x \cos y + \cos x \sin y$.

Example.

Example. In $QU/(\epsilon^2 = 0)$ using the yax order over $\mathbb{Q}[\![\hbar]\!]$, with $T = \mathbb{e}^{\hbar t}$, $\bar{T} = T^{-1}$, $\mathcal{A} = \mathbb{e}^{\gamma\alpha}$, and $\bar{\mathcal{A}} = \mathcal{A}^{-1}$,

$$R_{ij} = \mathbb{e}^{\hbar(y_i x_j - t_i a_j/\gamma)} \left(1 + \epsilon\hbar\left(a_i a_j/\gamma - \gamma\hbar^2 y_i^2 x_j^2/4\right)\right) \in \mathcal{S}(B_i, B_j),$$

$$m = \mathbb{e}^{(\alpha_1+\alpha_2)a + \eta_2\xi_1(1-T)/\hbar + (\xi_1\bar{\mathcal{A}}_2 + \xi_2)x + (\eta_1 + \eta_2\bar{\mathcal{A}}_1)y} (1 + \epsilon\lambda_m) \in \mathcal{S}(B_1^*, B_2^*, B),$$

with $\lambda_m = 2a\eta_2\xi_1 T + \frac{1}{4}\gamma\eta_2^2\xi_1^2\left(3T^2 - 4T + 1\right)/\hbar - \frac{1}{2}\gamma\eta_2\xi_1^2(3T-1)x\bar{\mathcal{A}}_2 - \frac{1}{2}\gamma\eta_2^2\xi_1(3T-1)y\bar{\mathcal{A}}_1 + \gamma\eta_2\xi_1 xy\hbar\bar{\mathcal{A}}_1\bar{\mathcal{A}}_2$,

$$\Delta = \mathbb{e}^{\tau(t_1+t_1) + \eta(y_1 + T_1 y_2) + \alpha(a_1+a_2) + \xi(x_1+x_2)} (1 + \epsilon\lambda_\Delta) \in \mathcal{S}(B^*, B_1, B_2),$$

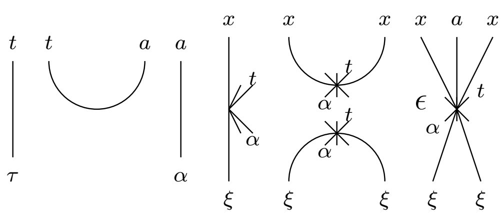
with $\lambda_\Delta = -a_1\eta T_1 y_2\hbar - a_1\xi x_2\hbar + \frac{1}{2}\gamma\eta^2 T_1 y_1 y_2\hbar + \frac{1}{2}\gamma\xi^2 x_1 x_2\hbar$, and

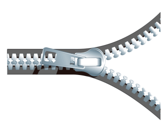
$$S = \mathbb{e}^{-\tau t - \alpha a - \eta\xi(1-\bar{T})\mathcal{A}/\hbar - \bar{T}\eta y\mathcal{A} - \xi x\mathcal{A}} (1 + \epsilon\lambda_S) \in \mathcal{S}(B^*, B),$$

with $\lambda_S = 2\bar{T}\mathcal{A}a\eta\xi - \bar{T}\mathcal{A}a\eta y\hbar - a\xi x\hbar\mathcal{A} - \frac{1}{4}\gamma\eta^2\xi^2\left(1 - 4\bar{T} + 3\bar{T}^2\right)\mathcal{A}^2/\hbar - \frac{1}{2}\gamma\eta^2 y^2\hbar\bar{T}^2\mathcal{A}^2 - \frac{1}{2}\gamma\eta^2\xi\bar{T}(1-3\bar{T})y\mathcal{A}^2 + \gamma\eta\xi(1-\bar{T})\mathcal{A} - \frac{1}{2}\gamma\eta\xi^2(1-3\bar{T})x\mathcal{A}^2 - \gamma\eta\xi xy\hbar\bar{T}\mathcal{A}^2 + \gamma\eta y\hbar\bar{T}\mathcal{A} - \frac{1}{2}\gamma\xi x^2\hbar\mathcal{A}^2$.

Problem. Compute the likes of $m /\!\!/ \Delta = \left(m|_{b\to\partial_\beta}\Delta\right)_{\beta=0}$ and $(R_{12}R_{34}) /\!\!/ m_2^{13} = \left((R_{12}R_{34})|_{b\to\partial_\beta} m_2^{13}\right)_{\beta=0}$.

A generic morphism:





The Zipping Issue.

The Contraction Theorem. If P has a finite ζ-degree and the y's and the q's are "small",

$$\left\langle P(z_i, \zeta^j)\right\rangle_{(\zeta_i)} = P\left(z_i, \overleftrightarrow{\partial}_{z_j}\right)\Big|_{z_i=0},$$

$$\left\langle P(z_i, \zeta^j)\mathbb{e}^{\eta^i z_i + y_j\zeta^j}\right\rangle_{(\zeta_i)} = \left\langle P(z_i + y_i, \zeta^j)\mathbb{e}^{\eta^i(z_i+y_i)}\right\rangle_{(\zeta_i)},$$

(proof: replace $y_j \to \hbar y_j$ and test at $\hbar = 0$ and at $\partial_\hbar$), and

$$\left\langle P(z_i, \zeta^j)\mathbb{e}^{c + \eta^i z_i + y_j\zeta^j + q^i_j z_i\zeta^j}\right\rangle_{(\zeta_i)} = \det(\tilde{q})\left\langle P(\tilde{q}^k_i(z_k + y_k), \zeta^j)\mathbb{e}^{c + \eta^i\tilde{q}^k_i(z_k+y_k)}\right\rangle_{(\zeta_i)}$$

where $\tilde{q}$ is the inverse matrix of $1 - q$: $(\delta^i_j - q^i_j)\tilde{q}^j_k = \delta^i_k$ (proof: replace $q^i_j \to \hbar q^i_j$ and test at $\hbar = 0$ and at $\partial_\hbar$).

```
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];
```

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

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

```
QZip_{ζs_List,simp_}@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;
   simp /@ 𝔼[L, Q2, Det[qt] e^(-Q2) Zip_{ζs}[e^Q1 (P /. zrule)]]];
QZip_{ζs_List} := QZip_{ζs,CF};
```

```
LZip_{ζs_List,simp_}@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;
   simp /@
    𝔼[L2, Q2, Det[lt] e^(-L2-Q2)
     Zip_{ζs}[e^(L1+Q1) (P /. T2t /. zrule)]] //. t2T];
LZip_{ζs_List} := LZip_{ζs,CF};
```

```
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];
```

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

```
tm_{i_,j_→k_} := Module[{tk},
   𝔼[(τ_i + τ_j) t_k + α_i a_k + α_j a_k, η_i y_k + ξ_j x_k, 1]
    (tSW_{xy,i,j→tk} /. {t_tk → t_k, T_tk → T_k, y_tk → e^(-γ α_i) y_k,
       a_tk → a_k, x_tk → e^(-γ α_j) x_k})];
m_{j_→k_}[ℰ_𝔼] := ℰ ~B_{j,k}~ tm_{j,k→k};
```

```
S[U_, kk_] := S[U, kk] = Module[{OE},
    OE = m_{3,2,1→1}[Exp_{QU_1,$k}[η, S_1[QU[y_1]] /. QU → Times]
       Exp_{QU_2,$k}[α, S_2[QU[a_2]] /. QU → Times]
       Exp_{QU_3,$k}[ξ, S_3[QU[x_3]] /. QU → Times]];
    𝔼[-t_1 τ_1 + OE⟦1⟧, OE⟦2⟧, OE⟦3⟧] /.
     {η → η_1, α → α_1, ξ → ξ_1}];
tS_{i_} := S[$U, $k] /. {(v : τ | η | α | ξ)_1 → v_i,
    (v : t | T | y | a | x)_1 → v_i};
```

```
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[ (* Warning:
           wrong unless $p≥$k+1! *)
           ReplacePart[1 → 0]@
            Exp_{QU_1,$k}[η, Δ_{1→1,2}[QU[y_1]] /. QU → Times],
           ReplacePart[2 → 0]@
            Exp_{QU_3,$k}[α, Δ_{3→3,4}[QU[a_3]] /. QU → Times],
           ReplacePart[1 → 0]@
            Exp_{QU_5,$k}[ξ, Δ_{5→5,6}[QU[x_5]] /. QU → Times]
         ] /. {η → η_1, α → α_1, ξ → ξ_1}];
    𝔼[τ_1 (t_1 + t_2) + α_1 (a_1 + a_2), OE⟦2⟧, OE⟦3⟧]];
tΔ_{i_→j_,k_} :=
  Δ[$U, $k] /. {(v : τ | η | α | ξ)_1 → v_i,
    (v : t | T | y | a | x)_1 → v_j, (v : t | T | y | a | x)_2 → v_k};
ℭ_{QU,k_}[R_{i_,j_}] := ℭ_{QU}[{y_i, a_i, x_i}_i, {y_j, a_j, x_j}_j,
   -ℏ γ^-1 t_i a_j + ℏ y_i x_j,
   Series[ⅇ^(ℏ γ^-1 t_i a_j - ℏ y_i x_j)
     (ⅇ^(ℏ b_i a_j) ⅇ_{q_ℏ,k}[ℏ y_i x_j] /. b_i → γ^-1 (ϵ a_i - t_i)), {ϵ, 0, k}]];
R[QU, kk_] := R[QU, kk] = Module[{OE},
    OE = Simplify /@ ℭ_{QU,kk}@R_{1,2};
    𝔼[-(ℏ a_2 t_1)/γ, ℏ x_2 y_1, Last@OE]];
tR_{i_,j_} :=
  R[$U, $k] /. {(v : t | T | y | a | x)_1 → v_i,
    (v : t | T | y | a | x)_2 → v_j};
t̅R̅_{i_,j_} := t̅R̅_{i,j} = tR_{i,j}~B_j~tS_j;
tC_{i_} := 𝔼[0, 0, T_i^{1/2} ⅇ^(-ϵ a_i ℏ) + 0_{$k}];
t̅C̅_{i_} := 𝔼[0, 0, T_i^{-1/2} ⅇ^(ϵ a_i ℏ) + 0_{$k}];
Kink[QU, kk_] :=
  Kink[QU, kk] =
   Block[{$k = kk}, (tR_{1,3} t̅C̅_2)~B_{1,2}~tm_{1,2→1}~B_{1,3}~tm_{1,3→1}];
tKink_{i_} := Kink[$U, $k] /. {(v : t | T | y | a | x)_1 → v_i};
K̅i̅n̅k̅[QU, kk_] :=
  K̅i̅n̅k̅[QU, kk] =
   Block[{$k = kk}, (t̅R̅_{1,3} tC_2)~B_{1,2}~tm_{1,2→1}~B_{1,3}~tm_{1,3→1}];
t̅K̅i̅n̅k̅_{i_} := K̅i̅n̅k̅[$U, $k] /. {(v : t | T | y | a | x)_1 → v_i}
```

Program (as in Projects/PPSA/Verification.nb).

```
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];
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; )
```

```
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];
```

```
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];
  𝕆_{U}[specs___, 𝔼[L_, Q_, P_]] :=
   𝕆_{U}[specs, SS@Normal[P ⅇ^{L+Q}]];
  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[])]; ]
```