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/.

remove cell tags

DocileQ

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);
```

fix

Initialization / Utilities

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

TD

```
$p = 2; $k = 1; $U = QU; $E := {$k, $p};
$trim := {ħ^p_. /; p > $p → 0, ϵ^k_. /; k > $k → 0};
SetAttributes[{SS, SST}, HoldAll];
q_ħ = e^(γ ϵ ħ);
(* Upper to lower and lower to Upper: *)
U2l = {B_i_^p_. → e^(-p ħ γ b_i), B^p_. → e^(-p ħ γ b), T_i_^p_. → e^(p ħ t_i), T^p_. → e^(p ħ t), 𝒜_i_^p_. → e^(p γ α_i), 𝒜^p_. → e^(p γ α)};
l2U = {e^(c_. b_i_ + d_.) :→ B_i^(-c/(ħ γ)) e^d, e^(c_. b + d_.) :→ B^(-c/(ħ γ)) e^d,
    e^(c_. t_i_ + d_.) :→ T_i^(c/ħ) e^d, e^(c_. t + d_.) :→ T^(c/ħ) e^d,
    e^(c_. α_i_ + d_.) :→ 𝒜_i^(c/γ) e^d, e^(c_. α + d_.) :→ 𝒜^(c/γ) e^d,
    e^ℰ_ :→ e^(Expand@ℰ)};
SS[ℰ_, op_] := Collect[
    Normal@Series[If[$p > 0, ℰ, ℰ /. U2l], {ħ, 0, $p}],
    ħ, op];
SS[ℰ_] := SS[ℰ, Together];
SST[ℰ_, op___] := SS[ℰ /. U2l, 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);
```

remove $p!

$ħ \to \epsilon$.

CF

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

SeriesData

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):

SP

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

DeclareAlgebra

QLImplementation

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];
```

QLImplementation

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];
   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[])]; ]
```

DeclareMorphism

QLImplementation

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

QLImplementation

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 $\Delta_{i,j_1,j_2,\ldots}$.

CU

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

QU

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};
```

X

Implementing θ

theta

$$\text{DeclareMorphism}\left[C\theta,\ CU\to CU,\ \{y\to -x_{CU},\ a\to -a_{CU},\ x\to -y_{CU}\},\ \{t_{i\_}\to -t_i,\ T_{i\_}\to T_i^{-1},\ t\to -t,\ T\to T^{-1}\}\right];$$

$$\text{DeclareMorphism}\left[Q\theta,\ QU\to QU,\ \left\{y:\to \mathbb{O}_{QU}\left[\{a,x\},\ SS\left[-T^{-1/2}\,e^{\hbar\,\epsilon\,a}\,x\right]\right],\ a\to -a_{QU},\ x:\to \mathbb{O}_{QU}\left[\{a,y\},\ SS\left[-T^{-1/2}\,e^{\hbar\,\epsilon\,a}\,y\right]\right]\right\},\ \{t_{i\_}\to -t_i,\ T_{i\_}\to T_i^{-1},\ t\to -t,\ T\to T^{-1}\}\right]$$

The Asymmetric Dequantizator

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

ADeq

In[∘]:=
$$\text{A}\mathbb{D}\$f = \gamma\left(\left(\text{Cosh}\left[\hbar\left(a\,\epsilon + \frac{\gamma\,\epsilon}{2} - \frac{t}{2}\right)\right] - \text{Cosh}\left[\hbar\sqrt{\left(\frac{t-\gamma\,\epsilon}{2}\right)^2 + \epsilon\,\omega}\right]\right)\Big/\left(\hbar\,e^{\hbar\,((a+\gamma)\,\epsilon - t/2)}\,\text{Sinh}\left[\frac{\gamma\,\epsilon\,\hbar}{2}\right]\left(a^2\,\epsilon + a\,\gamma\,\epsilon - a\,t - \omega\right)\right)\right);$$

ADeq

In[∘]:=
$$\text{A}\mathbb{D}\$\omega = \gamma\,\text{CU}[y,x] + \epsilon\,\text{CU}[a,a] - (t-\gamma\,\epsilon)\,\text{CU}[a];$$

ADeq

In[∘]:=
$$\text{DeclareMorphism}\left[\text{A}\mathbb{D},\ QU\to CU,\ \{a\to a_{CU},\ x\to CU@x,\ y:\to \mathbb{S}_{CU}[SS[\text{A}\mathbb{D}\$f],\ a\to a_{CU},\ \omega\to \text{A}\mathbb{D}\$\omega] ** y_{CU}\}\right]$$

The Symmetric Dequantizator

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

SDeq

In[∘]:=
$$\text{S}\mathbb{D}\$g = \sqrt{\left(\left(2\,\gamma\left(\text{Cosh}\left[\frac{\hbar}{2}\sqrt{t^2+\gamma^2\,\epsilon^2+4\,\epsilon\,\varpi}\right] - \text{Cosh}\left[\frac{t-\epsilon\,\gamma-2\,\epsilon\,a}{2/\hbar}\right]\right)\right)\Big/\left(\text{Sinh}\left[\frac{\gamma\,\epsilon\,\hbar}{2}\right]\left(t\,(2\,a+\gamma) - 2\,a\,(a+\gamma)\,\epsilon + 2\,\varpi\right)\hbar\right)\right)};$$

SDeq

In[∘]:=
$$\text{S}\mathbb{D}\$f = \text{Simplify}\left[e^{\hbar\,(t/2-\epsilon\,a)}\left(\text{S}\mathbb{D}\$g\ /.\ \{a\to -a,\ t\to -t\}\right)\right];$$

SDeq

In[∘]:=
$$\text{S}\mathbb{D}\$\varpi = \gamma\,\text{CU}[y,x] + \epsilon\,\text{CU}[a,a] - (t-\gamma\,\epsilon)\,\text{CU}[a] - t\,\gamma\,1_{CU}/2;$$

SDeq

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

rho

```
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^(-γ ϵ ℏ)) / (ϵ ℏ)
                              0            0            );
ρ[e^ℰ_] := MatrixExp[ρ[ℰ]];
ρ[ℰ_] := (ℰ /. U2l /. t → γ ϵ /. (U : CU | QU)[u___] :→ Fold[Dot, ( 1  0
                                                                    0  1 ), ρ /@ U /@ {u}])
```

tSW

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

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 $\partial_\eta F = \partial_\eta\left(e^{-\eta y} e^{\eta G}\right) = -yF + FG$ with initial conditions $F(\eta = 0) = 1$. So we set it up and solve:

tSW

In[ ]:=
```
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, $k}, {i, 0, l}, {j, 0, l}, {k, 0, l}];
     F = fs.(bs = fs /. f_{l_,i_,j_,k_}[η] :→ ϵ^l U@{y^i, a^j, x^k});
     es = Flatten[
        Table[Coefficient[e, b] == 0, {e, {F - 1_U /. η → 0, F ** G - y_U ** F - ∂_η F}}, {b, bs}]];
     F = F /. DSolve[es, fs, η][[1]];
     𝔼[0,
        ξ x + η y + (U /. {CU → -t η ξ, QU → η ξ (1 - T) / ℏ}),
        F + O_$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];
```

R in QU.

The Faddeev-Quesne formula:

Faddeev

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

R

In[∘]:=
```
QU[R_{i_,j_}] := 𝕆_QU[{y₁, a₁}_i, {a₂, x₂}_j, SS[𝕖^(ℏ b₁ a₂) 𝕖_{q_ℏ}[ℏ y₁ x₂] /. b₁ → γ^-1 (ϵ a₁ - t_i)]];
QU[R^-1_{i_,j_}] := S_j@QU[R_{i,j}];
```

Exponentials as needed.

Exp

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

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

Exp

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_] := 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]] ]]
```

Zip and Bind

E

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];
```

Zip

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

Zip

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.

Zip

```
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[K δ_{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 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.

Zip

```
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[K δ_{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 /. U2l /. zrule) /. Alternatives @@ zs → 0;
  simp /@ 𝔼[L2, Q2, Det[lt] e^(-L2-Q2) Zip_ζs[e^(L1+Q1) (P /. U2l /. zrule)]] //. l2U ];
LZip_{ζs_List} := LZip_{ζs,CF};
```

Bind

```
Bind_{}[L_, R_] := L R;
Bind_{is__}[L_𝔼, R_𝔼] := Module[{n},
  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}}]
  ] // LZip_{Flatten@Table[{β_{n@i}, τ_{n@i}, a_{n@i}}, {i, {is}}]} // QZip_{Flatten@Table[{x_{n@i}, η_{n@i}}, {i, {is}}]} ];
B_{l_List}[L_, R_] := Bind_l[L, R]; B_{is___}[L_, R_] := Bind_{is}[L, R];
```

Tensorial Representations

t1

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

tm

In[∘]:=
```
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};
```

In[∘]:= `tm_{1,2→3}`

Out[∘]=
$$\mathbb{E}\Big[a_3\,\alpha_1 + a_3\,\alpha_2 + t_3\,(\tau_1+\tau_2),\; y_3\,\eta_1 + e^{-\gamma\alpha_1}\,y_3\,\eta_2 + e^{-\gamma\alpha_2}\,x_3\,\xi_1 + \frac{(1-T_3)\,\eta_2\,\xi_1}{\hbar} + x_3\,\xi_2,$$
$$1 + \frac{1}{4\hbar}\,\eta_2\,\xi_1\,\big(8\,\hbar\,a_3\,T_3 + 4\,e^{-\gamma\alpha_1-\gamma\alpha_2}\,\gamma\,\hbar^2\,x_3\,y_3 + 2\,e^{-\gamma\alpha_1}\,\gamma\,\hbar\,y_3\,\eta_2 - 6\,e^{-\gamma\alpha_1}\,\gamma\,\hbar\,T_3\,y_3\,\eta_2 +$$
$$2\,e^{-\gamma\alpha_2}\,\gamma\,\hbar\,x_3\,\xi_1 - 6\,e^{-\gamma\alpha_2}\,\gamma\,\hbar\,T_3\,x_3\,\xi_1 + \gamma\,\eta_2\,\xi_1 - 4\,\gamma\,T_3\,\eta_2\,\xi_1 + 3\,\gamma\,T_3^2\,\eta_2\,\xi_1\big)\,\epsilon + O[\epsilon]^2\Big]$$

tS

In[∘]:=
```
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,$k}[ξ, S_3[QU[x_3]] /. QU → Times]];
    𝔼[-t_1 τ_1 + OE⟦1⟧, OE⟦2⟧, OE⟦3⟧] /. {η → η_1, α → α_1, 𝒜 → 𝒜_1, ξ → ξ_1}];
tS_{i_} := S[$U, $k] /. {(v : τ | η | α | 𝒜 | ξ)_1 → v_i, (v : t | T | y | a | x)_1 → v_i};
```

In[∘]:= `tS_1`

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

tDelta

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[ (* 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};
```

In[∘]:= $t\Delta_{1\to 1,2}$

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

tR

In[∘]:=
```
ŒQU,k_[Ri_,j_] := ŒQU[{yi, ai, xi}i, {yj, aj, xj}j, -ℏ γ^-1 ti aj + ℏ yi xj,
   Series[e^(ℏ γ^-1 ti aj - ℏ yi xj) (e^(ℏ bi aj) e_(qℏ,k)[ℏ yi xj] /. bi → γ^-1 (ϵ ai - ti)), {ϵ, 0, k}]];
R[QU, kk_] := R[QU, kk] = Module[{OE},
    OE = Simplify /@ ŒQU,kk @ R1,2;
    𝔼[-(ℏ a2 t1)/γ, ℏ x2 y1, Last@OE] ];
tRi_,j_ := R[$U, $k] /. {(v : t | T | y | a | x)1 → vi, (v : t | T | y | a | x)2 → vj};
tR̄i_,j_ := tR̄i,j = tRi,j ~ Bj ~ tSj;
```

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

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

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

tC

In[∘]:=
```
tCi_ := 𝔼[0, 0, Ti^(1/2) e^(-ϵ ai ℏ) + 0$k];
tC̄i_ := 𝔼[0, 0, Ti^(-1/2) e^(ϵ ai ℏ) + 0$k];
```

In[∘]:= `Block[{$k = 3}, {tC1, tC̄2}]`

Out[∘]= $\Big\{\mathbb{E}\Big[0,\ 0,\ \sqrt{T_1}-\hbar\,a_1\,\sqrt{T_1}\,\epsilon+\frac{1}{2}\hbar^2\,a_1^2\,\sqrt{T_1}\,\epsilon^2-\frac{1}{6}\left(\hbar^3\,a_1^3\,\sqrt{T_1}\right)\epsilon^3+O[\epsilon]^4\Big],$
$\mathbb{E}\Big[0,\ 0,\ \frac{1}{\sqrt{T_2}}+\frac{\hbar\,a_2\,\epsilon}{\sqrt{T_2}}+\frac{\hbar^2\,a_2^2\,\epsilon^2}{2\,\sqrt{T_2}}+\frac{\hbar^3\,a_2^3\,\epsilon^3}{6\,\sqrt{T_2}}+O[\epsilon]^4\Big]\Big\}$

tKink

In[∘]:=
```
Kink[QU, kk_] := Kink[QU, kk] = Block[{$k = kk}, (tR1,3 tC̄2) ~ B1,2 ~ tm1,2→1 ~ B1,3 ~ tm1,3→1];
tKinki_ := Kink[$U, $k] /. {(v : t | T | y | a | x)1 → vi};
K̄ink[QU, kk_] := K̄ink[QU, kk] = Block[{$k = kk}, (tR̄1,3 tC2) ~ B1,2 ~ tm1,2→1 ~ B1,3 ~ tm1,3→1];
tK̄inki_ := K̄ink[$U, $k] /. {(v : t | T | y | a | x)1 → vi}
```

Alternative Algorithms

AltLogos

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}]]];
```

Asides

Aside

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

Aside

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