Cheat Sheet SL2Portfolio on 180217

February 17, 2018 2:46 PM

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

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

Should all coefficients be series in $\hbar/\epsilon$?

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 $\hbar$.

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

http://drorbn.net/AcademicPensieve/Projects/SL2Portfolio/
modified February 15, 2018.

$\mathcal{U}_{\gamma\epsilon;\hbar}$ 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 $(\phi f)(\psi g) = \langle\psi_1 S^{-1}f_3\rangle\langle\psi_3, f_1\rangle(\phi\psi_2)(f_2g)$ 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 $t := \epsilon a - \gamma b$, $T := e^{\hbar t} = A^{-1/2}B^{1/2}$ 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 $\epsilon = 0$, $\mathcal{U}_{\hbar;\gamma 0} = \langle b, y, a, x\rangle/([b,\cdot] = 0, [a,x] = \gamma x, [a,y] = -\gamma y, [x,y] = (1 - e^{-\hbar\gamma b})/\hbar)$ with $\Delta(b, y, a, x) = (b_1 + b_2, y_1 + e^{-\hbar\gamma b_1}y_2, a_1 + a_2, x_1 + x_2)$ and $\theta(y, b, a, x) = (-e^{\hbar\gamma b/2}x, -b, -a, -e^{\hbar\gamma b/2}y)$.

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

Casimir. $\omega = \gamma yx + \epsilon a^2 - (t - \gamma\epsilon)a$, satisfies…

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).

Fix!

```
$p = 3; $k = 1;
(* $k can't be ∞ at least because of Faddeev-Quesne. *)
If[$k == 0, ϵ = 0, ϵ /: ϵ^k_ /; k > $k := 0];
(* $k=0 fails in Series[..{ϵ,...}] *)
SetAttributes[{SS, SST}, HoldAll];
TRule = {T_i_ → e^(ħ t_i), T → e^(ħ t)};
SS[ℰ_] := Block[{ħ, ϵ}, (* Shielded Series *)
   Collect[Normal@Series[ℰ, {ħ, 0, $p}], ħ, Together] ];
SST[ℰ_] :=
  Block[{ħ, ϵ},
   Collect[Normal@Series[ℰ /. TRule, {ħ, 0, $p}], ħ,
    Together] ];
Simp[ℰ_, op_] := Collect[ℰ, _CU | _QU, op];
Simp[ℰ_] :=
  Simp[ℰ, Collect[Normal@Series[#, {ħ, 0, $p}], ħ,
     Expand] &];
SimpT[ℰ_] := Collect[ℰ, _CU | _QU,
   Collect[Normal@Series[# /. TRule, {ħ, 0, $p}], ħ,
     Expand] &];
```

```
DP_{α_→D_{x_}, β_→D_{y_}}[P_][λ_] :=
 Total[CoefficientRules[P, {α, β}] /.
   ({m_, n_} → c_) :→ c D[λ, {x, m}, {y, n}]]
```

```
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@CU@y = -y_CU; S@a_CU = -a_CU; S@x_CU = -x_CU;)
S_i_[CU, Centrals] = {t_i → -t_i};
DeclareAlgebra[QU, Generators → {y, a, x},
  Centrals → {t, T}];
q = SS[e^(γ ϵ ħ)];
B[a_QU, y_QU] = -γ y_QU; B[x_QU, a_QU] = -γ QU@x;
B[x_QU, y_QU] = (q - 1) QU@{y, x} +
   𝕆_QU[SS[(1 - T e^(-2 ϵ a ħ)) / ħ], {a}];
(S@y_QU = 𝕆_QU[SS[-T^-1 e^(ħ ϵ a) y], {a, y}]; S@a_QU = -a_QU;
 S@x_QU = 𝕆_QU[SS[-e^(ħ ϵ a) x], {a, x}];)
S_i_[QU, Centrals] = {t_i → -t_i, T_i → T_i^-1};
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[SS[-T^(-1/2) e^(ħ ϵ a) x], {a, x}], a → -a_QU,
  x → 𝕆_QU[SS[-T^(-1/2) e^(ħ ϵ a) y], {a, y}]}, {t → -t, T → T^-1}]
```

Can the $A\mathbb{D}$ and $S\mathbb{D}$ formulas be written so as to manifestly see their lowest term in ϵ? This may allow more flexibility with \$k. Or perhaps better, these should be written in implicit form and solved by power series.

```
AD$f = γ (Cosh[ħ (a e + γe/2 - t/2)] - Cosh[ħ Sqrt[((t-γe)/2)^2 + e ω]]) /
        (ħ e^(ħ ((a+γ) e - t/2)) Sinh[γeħ/2] (a^2 e + a γ e - a t - ω));
AD$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$f] /. e → ϵ, a → a_CU, ω → AD$ω] ** y_CU}]
SD$g = Sqrt[(2 γ (Cosh[ħ/2 Sqrt[t^2 + γ^2 e^2 + 4 e ϖ]] - Cosh[(t-eγ-2ea)/(2/ħ)])) /
        (Sinh[γeħ/2] (t (2 a + γ) - 2 a (a + γ) e + 2 ϖ) ħ)];
SD$f = Simplify[e^(ħ (t/2-e 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$f] /. e → ϵ, a → a_CU, ϖ → SD$ϖ] ** x_CU,
  y → 𝕊_CU[SS[SD$g] /. e → ϵ, a → a_CU, ϖ → SD$ϖ] ** y_CU }]
e_{q_,k_}[x_] := e^(Sum[(1 - q)^j x^j / (j (1 - q^j)), {j, 1, k}]); e_{q_}[x_] := e_{q,$k}[x]
QU[R_{i_,j_}] := 𝕆_QU[SS[e^(ħ b1 a2) e_q[ħ y1 x2] /. b1 → γ^-1 (ϵ a1 - t_i)],
   {y1, a1}_i, {a2, x2}_j];
QU[R^-1_{i_,j_}] := S_j@QU[R_{i,j}];
SetAttributes[{C𝕆, Q𝕆}, Orderless];
CU@C𝕆[specs___, 𝔼[L_, Q_, P_]] := 𝕆_CU[SS[e^(L+Q) P], specs];
QU@Q𝕆[specs___, 𝔼[L_, Q_, P_]] := 𝕆_QU[SS[e^(L+Q) P], specs];
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 = SS@({0, (1 - e^(-γ ϵ ħ)) / (ϵ ħ)}, {0, 0});
ρ[e^ℰ_] := MatrixExp[ρ[ℰ]];
ρ[ℰ_] :=
 (ℰ /. {t → γ ϵ, T → e^(ħ γ ϵ)} /.
   (U : CU | QU)[u___] :→ Fold[Dot, ({1, 0}, {0, 1}), ρ /@ U /@ {u}])
```

```
CΛ[t1_, y1_, a1_, x1_, ξ1_, η1_, δ_] := Module[
  {eqn, d, b, c, sol, λ, q, ν, ξ, η},
  eqn = 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)];
  sol = Solve[Thread[Flatten /@ eqn], {d, b, c}]〚1〛 /.
    C[1] → 0;
  λ = 𝕖^(-η y - ξ x + η ξ t)
    Normal@Series[𝕖^(c t + d y - 2 ϵ c a + b x) /. sol, {ϵ, 0, $k}];
  q = 𝕖^(ν (-t ξ η + η y + ξ x + δ y x));
  Collect[ν q^-1 DP_{ξ→D_x, η→D_y}[λ][q] /. ν → (1 + t δ)^-1,
    ϵ, Simplify] /. {t → t1, y → y1, a → a1, x → x1,
    ξ → ξ1, η → η1}];
```

If $G=e^{\xi x}ye^{-\xi x}$ then $F=e^{-\eta y}e^{\xi x}e^{\eta y}e^{-\xi x}=e^{-\eta y}e^{G}$ satisfies $\partial_\eta F=-yF+FG$ and $F_{\eta=0}=1$:

```
QΛ[T1_, y1_, a1_, x1_, ξ1_, η1_, δ_] := Module[
  {G, F, fs, f, bs, e, b, es, λ, q, ν, ξ, η, t},
  G = Simp[Table[ξ^k / k!, {k, 0, $k + 1}].
     NestList[Simp[x_QU ** # - # ** x_QU] &, y_QU, $k + 1]];
  fs = Flatten@Table[f_{l,i,j,k}[η], {l, 0, $k}, {i, 0, 1},
     {j, 0, 1}, {k, 0, 1}];
  F = fs.(bs = fs /. f_{l_,i_,j_,k_}[η] :→ ϵ^l QU@{y^i, a^j, x^k});
  es = Flatten[Table[Coefficient[e, b] == 0,
     {e, {F - 1_QU /. η → 0, F ** G - y_QU ** F - ∂_η F}}, {b, bs}]];
  {λ} = F /. DSolve[es, fs, η] /. {𝕖- → 1, QU → Times};
  q = 𝕖^(ν (-t ξ η + η y + ξ x + δ y x));
  Collect[ν q^-1 DP_{ξ→D_x, η→D_y}[λ][q] /. ν → (1 + t δ)^-1 /.
     t → (T - 1) / ħ, ϵ, Simplify] /.
   {T → T1, y → y1, a → a1, x → x1, ξ → ξ1, η → η1}];
```

```
SW_{x_i_, a_j_}[
  (𝒪 : C𝕆 | Q𝕆)[OrderlessPatternSequence[
    {lh___, x_i_, a_j_, rh___}_s_, more___, 𝔼[L_, Q_, P_]]]] :=
 𝒪[{lh, a_j, x_i, rh}_s, more,
  With[{q = 𝕖^(-γ α) ξ x_i + α a_j},
   𝔼[L, 𝕖^(-γ α) ξ x_i + (Q /. x_i → 0), 𝕖^-q DP_{x_i→D_ξ, a_j→D_α}[P][𝕖^q]] /.
    {α → ∂_{a_j} L, ξ → ∂_{x_i} Q}]]
SW_{a_j_, y_i_}[
  (𝒪 : C𝕆 | Q𝕆)[OrderlessPatternSequence[
    {lh___, a_j_, y_i_, rh___}_s_, more___, 𝔼[L_, Q_, P_]]]] :=
 𝒪[{lh, y_i, a_j, rh}_s, more,
  With[{q = 𝕖^(-γ α) η y_i + α a_j},
   𝔼[L, 𝕖^(-γ α) η y_i + (Q /. y_i → 0), 𝕖^-q DP_{y_i→D_η, a_j→D_α}[P][𝕖^q]] /.
    {α → ∂_{a_j} L, η → ∂_{y_i} Q}]]
SW_{x_i_, y_j_ → k_}[C𝕆[{lh___, x_i_, y_j_, rh___}_s_, more___,
   𝔼[L_, Q_, P_]]] := C𝕆[{lh, y_k, a_k, x_k, rh}_s, more,
  With[{q = ν (ξ x_k + η y_k + δ x_k y_k - t_k ξ η)},
   𝔼[L, q + (Q /. x_i | y_j → 0),
      𝕖^-q DP_{x_i→D_ξ, y_j→D_η}[P][CΛ[t_k, y_k, a_k, x_k, ξ, η, δ] 𝕖^q]] /.
     ν → (1 + t_k δ)^-1 /.
    {ξ → (∂_{x_i} Q /. y_j → 0), η → (∂_{y_j} Q /. x_i → 0), δ → ∂_{x_i,y_j} Q}]]
SW_{x_i_, y_j_ → k_}[Q𝕆[{lh___, x_i_, y_j_, rh___}_s_, more___,
   𝔼[L_, Q_, P_]]] := Q𝕆[{lh, y_k, a_k, x_k, rh}_s, more,
  With[{q = ν (ξ x_k + η y_k + δ x_k y_k - ħ^-1 (T_k - 1) ξ η)},
   𝔼[L, q + (Q /. x_i | y_j → 0),
      𝕖^-q DP_{x_i→D_ξ, y_j→D_η}[P][QΛ[T_k, y_k, a_k, x_k, ξ, η, δ] 𝕖^q]] /.
     ν → (1 + ħ^-1 (T_k - 1) δ)^-1 /.
    {ξ → (∂_{x_i} Q /. y_j → 0), η → (∂_{y_j} Q /. x_i → 0), δ → ∂_{x_i,y_j} Q}]]
```

To do. • Consider renormalizing x and y. • Implement variable swaps. • Implement $m_{ij\to k}$. • Implement $\mathbb{E}$, $R\mathbb{E}$, and the casts CU and QU. • Reconsider the expansion of T and q in the hope of improving speed.

abstract & merge, "compute once"

Replace swaps by "Rewrite Rules" (RR):

$RR_U[\{x_i, y_j\} \to \{y_k, a_k, x_k\}, RQ, \lambda]$ (RQ replaces q)

revised quadratic

the linear Logos!

Alt:

$RR_U[e^{\xi x_i} e^{\eta y_j} == \mathbb{O}_U[\mathbb{E}(RQ, \lambda)! \{y, a, x\}_k]]$

Alt:

$\boxed{RR_U[\{x_i, y_j\} \to \{y_k, a_k, x_k\}, \{\xi, \eta\}, RQ, \lambda]}$