# Cheat Sheet SL2Portfolio on 180220, take 2

February 20, 2018 10:32 AM

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

http://drorbn.net/AcademicPensieve/Projects/SL2Portfolio/
modified February 20, 2018.

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

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

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

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

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

Pairing by $(a,x)^* = \hbar(b,y)$ ($\Rightarrow \langle B, A\rangle = q$) making $\langle y^ib^i, a^jx^k\rangle = \delta_{ij}\delta_{kl}\hbar^{-(j+k)}j![k]_q!$ 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 $(\phi f)(\psi g) = \langle\psi_1 S^{-1}f_3\rangle\langle\psi_3, f_1\rangle(\phi\psi_2)(f_2 g)$ 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 $\epsilon = 0$, $\mathcal{U}_{\hbar;\gamma 0} = \langle b,y,a,x\rangle/([b,\cdot] = 0, [a,x] = \gamma x, [a,y] = -\gamma y, [x,y] = (1 - \mathbb{e}^{-\hbar\gamma b})/\hbar)$ with $\Delta(b,y,a,x) = (b_1 + b_2, y_1 + \mathbb{e}^{-\hbar\gamma b_1}y_2, a_1 + a_2, x_1 + x_2)$ and $\theta(y,b,a,x) = (-\mathbb{e}^{\hbar\gamma b/2}x, -b, -a, -\mathbb{e}^{\hbar\gamma b/2}y)$.

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

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

```
$p = 2; $k = 2;
(* $k can't be ∞ at least because of Faddeev-Quesne. *)
If[$k == 0, ϵ = 0, ϵ /: ϵ^k_ /; k > $k := 0];
(* $k=0 fails in Series[..{ϵ,...}] *)
SetAttributes[{SS, SST}, HoldAll];
TRule = {T_ → e^ħ t_, 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 $\epsilon$? This may allow more flexibility with $k. Or perhaps better, these should be written in implicit form and solved by power series.

```
AD$f = γ (Cosh[ħ (a e + (γ e)/2 - t/2)] - Cosh[ħ Sqrt[((t - γ e)/2)^2 + e ω]])
       / (ħ e^ħ ((a+γ) e - t/2) Sinh[(γ e ħ)/2] (a^2 e + a γ e - a t - ω));
AD$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$g = Sqrt[(2 γ (Cosh[ħ/2 Sqrt[t^2 + γ^2 e^2 + 4 e ϖ]] - Cosh[(t - e γ - 2 e a)/(2 ħ)]))
       / (Sinh[(γ e ħ)/2] (t (2 a + γ) - 2 a (a + γ) e + 2 ϖ) ħ)];
SD$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 b_1 a_2 e_q[ħ y_1 x_2] /. b_1 → γ^-1 (ϵ a_1 - t_i)],
   {y_1, a_1}_i, {a_2, x_2}_j];
QU[R_i_,j_]^-1 := S_j@QU[R_i,j];
SetAttributes[{CΘ, QΘ}, Orderless];
CU@CΘ[specs___, E[L_, Q_, P_]] := O_CU[SS[e^-ħ Q P], specs];
QU@QΘ[specs___, E[L_, Q_, P_]] := O_QU[SS[e^-ħ 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}])
```

Fix

X ✓

CΘ & CΘ are replaced by QE [a container] ✓

**Fear Not.** If $G = e^{\xi x} y e^{-\xi x}$ 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 $F_{\eta=0} = 1$:

```
λ[U_] := 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_{1,i,j,k}[η], {l, 0, $k}, {i, 0, l},
      {j, 0, l}, {k, 0, l}];
  F = fs.(bs = fs /. f_{_,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 /. DSolve[es, fs, η][[1]] /. {e- → 1, U → Times}];
```

```
wc[CU] = t; wc[QU] = (T - 1)/ħ;
Λ[U_] :=
  Λ[U] = Module[{Q, w},
    Q = (-w ξ η + η y + ξ x + δ y x)/(1 + w δ);
    Collect[(1 + w δ)^-1 e^-Q DP_{ξ→D_x, η→D_y}[λ[U]][e^Q] /. w → wc[U],
     ε, Simplify]];
Λ[U_, t1_, T1_, y1_, a1_, x1_, ξ1_, η1_, δ1_] :=
  Λ[U] /. {t → t1, T → T1, y → y1, a → a1, x → x1, ξ → ξ1,
    η → η1, δ → δ1};
```

```
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 = v (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][Λ[CU, t_k, T_k, y_k, a_k, x_k, ξ, η, δ]
      e^q]] /. v → (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 = v (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][Λ[QU, t_k, T_k, y_k, a_k, x_k, ξ, η, δ]
      e^q]] /. v → (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}]]
```

```
RR[{u_i_, w_j_} → {vs___, k_}, {v_, ω_}, RQ_, λ_][
  (O : CO | QO)[OrderlessPatternSequence[
    {Lh___, u_i_, w_j_, rh___}_s_, more___, E[Q_, P_]]]] :=
 O[{Lh, Sequence @@ (#_k & /@ {vs}), rh}_s, more, E[
  (RQ /. (v : u | w | t | T) :> v_k) + (Q /. u_i | w_j → 0),
  e^-RQ DP_{u_i→D_v, w_j→D_ω}[P][Λ[O, t_k, T_k, y_k, a_k, x_k, v, ω, δ]
    e^RQ] /. {v → (∂_{v_i} Q /. ω_j → 0), ω → (∂_{ω_j} Q /. v_i → 0),
   δ → ∂_{v_i, ω_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. • 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 $\hbar$.

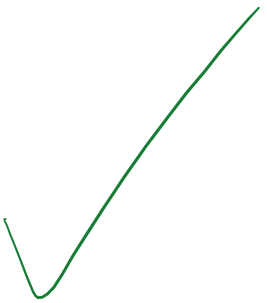
**Aside.**

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

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

λ should return the $\mathbb{E}$ which satisfies

$$\mathbb{E}_v[\{u,w\}, \mathbb{E}[vv+ww, 1]] == \mathbb{E}_v[\{y,a,x\}, \lambda_v[v,w,v,w]]$$



Λ should return the $\mathbb{E}$ which satisfies

$$\mathbb{E}_v[\{u,w\}, \mathbb{E}[vv+ww+\delta vw, 1]] == \mathbb{E}_v[\{y,a,x\}, \Lambda_v[v,w,v,w]]$$

**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;
DeclareAlgebra[U_Symbol, opts__Rule] :=
 Module[{gp, sr, cp, CE, pow,
   gs = Generators /. {opts}, cs = Centrals /. {opts}},
  (#U = U@#) & /@ gs;
  gp = Alternatives @@ gs; gp = gp | gp_; (* gens *)
  sr = Thread[gs → Range@Length@gs]; (* sorting → *)
  cp = Alternatives @@ cs; (* cents *)
  CE[ℰ_] := Collect[ℰ, _U,
    (Expand[#] /. ħ^d_ /; d > $p :→ 0) &];
  Ui_[ℰ_] :=
   ℰ /. {t : cp :→ ti, u_U :→ Replace[u, x_ :→ xi, 1]};
  Ui_[NCM[]] = pow[ℰ_, 0] = U@{} = 1U = U[];
  B[U@(x_)i_, U@(y_)i_] :=
   B[U@xi, U@yi] = Ui@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_ ** 1U := x; 1U ** x_ := x;
  (a_. * x_U) ** (b_. * y_U) :=
   If[a b === 0, 0, CE[a b (x ** y)]];
  U[xx___, x_] ** U[y_, yy___] :=
   If[OrderedQ[{x, y} /. sr], U[xx, x, y, yy],
    U@xx ** (U@y ** U@x + B[U@x, U@y]) ** 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;
  OU[poly_, specs___] := Module[{sp, null, vs, us},
    sp = Replace[{specs}, l_List :→ lnull, {1}];
    vs = Join @@ (First /@ sp);
    us = Join @@ (sp /. l_s_ :→ (l /. x_i :→ xi));
    CE[Total[
       CoefficientRules[poly, vs] /. (p_ → c_) :→ c U@(us^p)
     ]] /. x_null :→ x];
  pow[ℰ_, n_] := pow[ℰ, n - 1] ** ℰ;
  SU[ℰ_, ss___Rule] := CE@Total[
    CoefficientRules[ℰ, First /@ {ss}] /.
     (p_ → c_) :→
      c NCM @@ MapThread[pow, {Last /@ {ss}, p}]];
  Si_[c_. * u_U] :=
   CE[(c /. Si[U, Centrals])
     DeleteCases[u, _i] **
      Ui[NCM @@ Reverse@Cases[u, x_i :→ S@U@x]]]];
```

```
DeclareMorphism[m_, U_ → V_, ongs_List, oncs_List: {}] := (
  Replace[ongs, (g_ → img_) :→ (m[U[g]] = img), {1}];
  m[1U] = 1V;
  m[U[g_i_]] := Vi[m[U@g]];
  m[U[vs__]] := NCM @@ (m /@ U /@ {vs});
  m[ℰ_] := Simp[ℰ /. oncs /. u_U :→ m[u]]; )
Si_[ℰ_Plus] := Simp[Si /@ ℰ];
```

### Alternative Algorithms.

```
λalt[CU] := Module[{eq, d, b, c, so},
  eq = ρ@e^(ξ xCU).ρ@e^(η yCU) == ρ@e^(d yCU).ρ@e^(c (t 1CU - 2 ϵ aCU)).ρ@e^(b xCU);
  {so} = Solve[Thread[Flatten /@ eq], {d, b, c}] /. C@1 → 0;
  Normal@Series[e^(-η y-ξ x+η ξ t+c t+d y-2 ϵ c a+b x) /. so,
    {ϵ, 0, $k}]];
```

**(Proposed) Agenda.** Using Århus-like techniques, construct a map $Z\colon \mathcal{T}_{vous} \to \mathcal{A}_{vous}$, where $\mathcal{T}_{vous}$ is the space of VOUS-tangles: Virtual tangles with only Over or Under strands, some labeled as Surgery strands, with a non-singular linking matrix between the surgery strands, modulo acyclic Reidemeister 2 moves and Kirby slide relations, and where $\mathcal{A}_{vous}$ is some space of arrow diagrams modulo appropriate relations. The construction will either fix the definitions of $\mathcal{T}_{vous}$ and $\mathcal{A}_{vous}$ or will allow some flexibility that will be fixed so that the following will hold true:

1. $\mathcal{T}_{vous}$ should have a clearer topological interpretation, perhaps in terms of Heegaard diagrams.
2. $\mathcal{A}_{vous}$ should pair with some kind of Lie bialgebras.
3. $\mathcal{A}_{vous}$ should be the associated graded of $\mathcal{T}_{vous}$ and $Z$ should be an expansion.
4. Ordinary tangles $\mathcal{T}_{ord}$ and ordinary virtual tangles $\mathcal{T}_{v\text{-}ord}$ should map into $\mathcal{T}_{vous}$, and when viewed on $\mathcal{T}_{(v\text{-})ord}$, the invariant $Z$ should explain the Drinfel'd double construction.

It may be better to first construct a $Z$ and only later worry about the numbered properties. Yet property 4 has stand-alone topological content which may be very interesting: $\mathcal{T}_{vous}$ is a space with an $R3$-free presentation and which contains $\mathcal{T}_{(v\text{-})ord}$, at least nearly faithfully. What does it mean? To what extent does it make $R3$ superfluous in knot theory?

As for constructing $Z$, the first step should be a $Z\colon \mathcal{T}_{vou} \to \mathcal{A}_{vou}$ (no surgery), which would have a prescribed behaviour on strand-doubling.