Cheat Sheet SL2Portfolio on 180214

February 14, 2018 8:28 AM

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

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

Rescale $T^2 \to T$ back again!

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 13, 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 $(\phi f)(\psi g)=\langle\psi_1S^{-1}f_3\rangle\langle\psi_3,f_1\rangle(\phi\psi_2)(f_2g)$ 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 $t:=\epsilon a-\gamma b$, $T:=e^{\hbar t/2}=A^{-1/2}B^{1/2}$ get

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

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

At $\epsilon=0$, $\mathcal{U}_{\hbar;\gamma0}=\langle b,y,a,x\rangle/([b,\cdot]=0,[a,x]=\gamma x,[a,y]=-\gamma y,[x,y]=(1-e^{-\hbar\gamma b})/\hbar)$ 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 verify!

```
$p = 3; $k = 1; ϵ /: ϵ^k_ /; k > $k := 0;
(* $k can't be ∞ at least because of Quesne. ~~Can't be ≤~~
~~1 at least because of the explicit ϵ² in SD$g.~~ *)
SetAttributes[{SS, SST}, HoldAll];
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 /. {T_i_ → e^(ℏ t_i/2), T → e^(ℏ t/2)},
      {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[# /. {T_i_ → e^(ℏ t_i/2), T → e^(ℏ t/2)},
      {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};
```

Faddeev

```
DeclareAlgebra[QU, Generators → {y, a, x},
  Centrals → {t, T}];
q = SS[e^(γ ϵ ℏ)]; (*T=SS[e^(ℏ t/2)];*)
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 - T² e^(-2 ϵ a ℏ)) / ℏ], {a}];
(S@y_QU = 𝕆_QU[SS[-T^-2 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 e^(ℏ ϵ a) x], {a, x}], a → -a_QU,
   x → 𝕆_QU[SS[-T^-1 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 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] (a² ϵ + a γ ϵ - a t - ω));
```

ϵ→e

```
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}]
```

/. e→ϵ

```
SD$g = Sqrt[(2 γ (Cosh[ℏ/2 Sqrt[t² + γ² ϵ² + 4 ϵ ϖ]] - Cosh[(t-ϵγ-2ϵa)/(2/ℏ)])) /
   (Sinh[γϵℏ/2] (t (2 a + γ) - 2 a (a + γ) ϵ + 2 ϖ) ℏ)];
```

ϵ→e

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

ϵ→e

```
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 }]
```

/. e→ϵ

```
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/2)} /.
   (U : CU | QU)[u___] ⧴ Fold[Dot, ({{1, 0}, {0, 1}}), ρ /@ U /@ {u}])
```

try to eliminate.

```
SS_ϵ[ℰ_] :=
  Block[{ϵ}, Collect[Normal@Series[ℰ, {ϵ, 0, $k}],
    ϵ, Together]]; (* Shielded ϵ-Series *)
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]] /.
```

```
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_i_,y_j_[
```

try to eliminate.

```
SSϵ[ℰ_] :=
  Block[{ϵ}, Collect[Normal@Series[ℰ, {ϵ, 0, $k}],
    ϵ, Together]]; (* Shielded ϵ-Series *)
CΛ[t1_, y1_, a1_, x1_, ξ1_, η1_, δ_] := Module[
  {eqn, d, b, c, sol, λ, q, ν, ξ, η},
  eqn = ρ[e^(ξ xCU)].ρ[e^(η yCU)] ==
    ρ[e^(d yCU)].ρ[e^(c (t 1CU - 2 ϵ aCU))].ρ[e^(b xCU)];
  sol = Solve[Thread[Flatten /@ eqn], {d, b, c}][[1]] /.
    C[1] → 0;
  λ = Simplify[e^(-η y-ξ x+η ξ t) SSϵ[e^(c t+d y-2 ϵ c a+b x) /. sol]];
  q = e^(ν (-t ξ η+η y+ξ x+δ y x));
  Collect[ν q^-1 DP_{ξ→Dx, η→Dy}[λ][q] /. ν → (1 + t δ)^-1,
    ϵ, Simplify] /. {t → t1, y → y1, a → a1, x → x1,
    ξ → ξ1, η → η1}];
```

abstract & merge, "compute once"

$e^{\xi x} e^{\eta y} = e^{\eta y} \lambda e^{\xi x} \Rightarrow \lambda = e^{-\eta y} e^{\xi x} e^{\eta y} e^{-\xi x}$

commut! $= e^{\xi x} y e^{-\xi x}$

```
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[xQU ** # - # ** xQU] &, yQU, $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 - 1QU /. η → 0, F ** G - yQU ** F - ∂_η F}}, {b, bs}]];
  {λ} = F /. DSolve[es, fs, η] /. {ϵ → 1, QU → Times};
  q = e^(ν (-t ξ η+η y+ξ x+δ y x));
  Collect[ν q^-1 DP_{ξ→Dx, η→Dy}[λ][q] /. ν → (1 + t δ)^-1 /.
     t → (T^2 - 1) / ℏ, ϵ, Simplify] /.
   {T → T1, y → y1, a → a1, x → x1, ξ → ξ1, η → η1}];
```

```
SW_{xi_, aj_}[
  (O : CO | QO)[OrderlessPatternSequence[
    {Lh___, xi_, aj_, rh___}s_, more___, 𝔼[L_, Q_, P_]]]] :=
 O[{Lh, aj, xi, rh}s, more,
  With[{q = e^(-γ α) ξ xi + α aj},
   𝔼[L, e^(-γ α) ξ xi + (Q /. xi → 0), e^-q DP_{xi→Dξ, aj→Dα}[P][e^q]] /.
    {α → ∂_{aj} L, ξ → ∂_{xi} Q}]]
SW_{aj_, yi_}[
  (O : CO | QO)[OrderlessPatternSequence[
    {Lh___, aj_, yi_, rh___}s_, more___, 𝔼[L_, Q_, P_]]]] :=
 O[{Lh, yi, aj, rh}s, more,
  With[{q = e^(-γ α) η yi + α aj},
   𝔼[L, e^(-γ α) η yi + (Q /. yi → 0), e^-q DP_{yi→Dη, aj→Dα}[P][e^q]] /.
    {α → ∂_{aj} L, η → ∂_{yi} Q}]]
```

$\nu = \text{null}, \lambda = 1$

```
SW_{xi_, yj_ → k_}[CO[{Lh___, xi_, yj_, rh___}s_, more___,
   𝔼[L_, Q_, P_]]] := CO[{Lh, yk, ak, xk, rh}s, more,
 With[{q = ν (ξ xk + η yk + δ xk yk - tk ξ η)},
  𝔼[L, q + (Q /. xi | yj → 0),
    e^-q DP_{xi→Dξ, yj→Dη}[P][CΛ[tk, yk, ak, xk, ξ, η, δ] e^q]] /.
   ν → (1 + tk δ)^-1 /.
   {ξ → (∂_{xi} Q /. yj → 0), η → (∂_{yj} Q /. xi → 0), δ → ∂_{xi,yj} Q}]]
SW_{xi_, yj_ → k_}[QO[{Lh___, xi_, yj_, rh___}s_, more___,
   𝔼[L_, Q_, P_]]] := QO[{Lh, yk, ak, xk, rh}s, more,
 With[{q = ν (ξ xk + η yk + δ xk yk - ℏ^-1 (Tk^2 - 1) ξ η)},
  𝔼[L, q + (Q /. xi | yj → 0),
    e^-q DP_{xi→Dξ, yj→Dη}[P][QΛ[Tk, yk, ak, xk, ξ, η, δ] e^q]] /.
   ν → (1 + ℏ^-1 (Tk^2 - 1) δ)^-1 /.
   {ξ → (∂_{xi} Q /. yj → 0), η → (∂_{yj} Q /. xi → 0), δ → ∂_{xi,yj} 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.

Replace swaps by "Rewrite Rules" (RR):

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

revised quadratic

the linear Logos!

Alt:

$RR_\upsilon[e^{\xi x_i} e^{\eta y_j} == \mathcal{O}_\upsilon[\mathbb{E}(RQ, \lambda) : \{y, a, x\}_k]]$

Alt:

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

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_;
  (* gen's pattern *)
  sr = Thread[gs → Range@Length@gs]; (* sorting rule *)
  cp = Alternatives @@ cs; (* cent's pattern *)
  CE[ℰ_] := Collect[ℰ, _U,
    (Expand[#] /. ħ^d_ /; d > $p :> 0) &];
  U_{i_}[ℰ_] :=
   ℰ /. {t : cp :> t_i, u_U :> Replace[u, x_ :> x_i, 1]};
  U_{i_}[NCM[]] = pow[ℰ_, 0] = U@{} = 1_U = U[];
  B[U@(x_)_{i_}, U@(y_)_{i_}] :=
   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_ ** 1_U := x; 1_U ** x_ := x;
  (a_. * x_U) ** (b_. * y_U) :=
```

improve

```
   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;
  𝕆_U[poly_, specs___] := 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}]];
  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]]]; ]
```

```
DeclareMorphism[m_, U_ → V_, ongs_List, oncs_List: {}] := (
  Replace[ongs, (g_ → img_) :> (m[U[g]] = img), {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]]; )
S_{i_}[ℰ_Plus] := Simp[S_i /@ ℰ];
```

(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.