Cheat Sheet PPSA

(formulas for the PPSA paper)

$\mathcal{U}_{\gamma\epsilon;\hbar}$ 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 $\langle y^l b^i, a^j x^k\rangle = \delta_{ij}\delta_{kl}\hbar^{-(j+k)} j![k]_q!$ so $R = \sum \frac{\hbar^{j+k} y^k b^j \otimes a^j x^k}{j![k]_q!}$. 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 $t := \epsilon a - \gamma b$, $T := e^{\hbar t/2} = A^{-1/2}B^{1/2}$ 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 $\epsilon = 0$, $\mathcal{U}_{\hbar;\gamma 0} = \langle b,y,a,x\rangle/([b,\cdot] = 0, [a,x] = \gamma x, [a,y] = -\gamma y, [x,y] = (1 - e^{-\hbar\gamma b})/\hbar)$ 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).

```
$TℏD = 3; $TϵD = 2; ϵ /: ϵ^d_· /; d > $TϵD := 0;
(* $TϵD can't be ∞ at least because of Quesne. Can't be ≤
 1 at least because of the explicit ϵ^2 in S𝔻$g. *)
SetAttributes[{SS, SST}, HoldAll];
SS[ℰ_] := Block[{ℏ, ϵ}, (* Shielded Series *)
   Collect[Normal@Series[ℰ, {ℏ, 0, $TℏD}], 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, $TℏD}], ℏ, Together] ];
Simp[ℰ_, op_] := Collect[ℰ, _CU | _QU, op];
Simp[ℰ_] :=
  Simp[ℰ, Collect[Normal@Series[#, {ℏ, 0, $TℏD}],
     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, $TℏD}], 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 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)]; (*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 - T^2 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 \$TϵD. Or perhaps better, these should be written in implicit form and solved by power series.

```
A𝔻$f = γ/ℏ e^(ℏ (t/2 - (a+γ) ϵ))
  (Cosh[ℏ (a ϵ + γ ϵ/2 - t/2)] - Cosh[ℏ Sqrt[((t - γ ϵ)/2)^2 + ϵ ω]]) /
  (Sinh[γ ϵ ℏ/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[(Cosh[ℏ/2 Sqrt[t^2 + γ^2 ϵ^2 + 4 ϵ ϖ]] - Cosh[ℏ/2 (t - (2 a + γ) ϵ)]) /
   (Sinh[γ ϵ ℏ/2] (t (2 a + γ) - 2 a (a + γ) ϵ + 2 ϖ) ℏ / (2 γ))];
```

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

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

```
e_{q_,n_}[x_] := e^(Sum[(1 - q)^k x^k / (k (1 - q^k)), {k, 1, n}]); e_q_[x_] := e_{q,$TϵD}[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^-1_{i_,j_}] := S_j@QU[R_{i,j}];
```

```
SetAttributes[ℂ𝕆, Orderless];
CU@ℂ𝕆[specs___, 𝔼[L_, Q_, P_]] := 𝕆_CU[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}])
```

```
SSϵ[ℰ_] :=
  Block[{ϵ}, Collect[Normal@Series[ℰ, {ϵ, 0, $TϵD}],
    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 CU[] - 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→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}
  ];
QΛ[T_, y1_, a1_, x1_, ξ1_, η1_, δ_] := Module[
   {adx, G, F, f, unowns, bas, eqns, sol, λ, q, ν, ξ, η, t},
   adx[ℰ_] := Simp[x_QU ** ℰ - ℰ ** x_QU];
   G = Simp[NestList[adx, y_QU, $TϵD + 1].
      Table[ξ^k / k!, {k, 0, $TϵD + 1}]];
   F = Sum[f_{l,i,j,k}[η] ϵ^l QU@{y^i, a^j, x^k}, {l, 0, $TϵD},
     {i, 0, l}, {j, 0, l}, {k, 0, Min[l, 2 l - i - j]}];
   unowns = Cases[F, f___[η], ∞];
   bas =
    Union @@ Table[ϵ^l Cases[Coefficient[F, ϵ, l], _QU, ∞],
      {l, 0, $TϵD}];
   eqns =
    Flatten[
     {(Coefficient[F - QU[], #] /. η → 0) == 0,
        Expand[Coefficient[Simp[F ** G - y_QU ** F - ∂_η F],
           #]] == 0} & /@ bas];
   {sol} = DSolve[eqns, unowns, η];
   λ = Collect[F /. sol /. {e- → 1, QU → Times}, ϵ,
     Simplify];
   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^2 - 1) / ℏ, ϵ, Simplify] /.
    {y → y1, a → a1, x → x1, ξ → ξ1, η → η1}
  ];
```

```
SW_{x_i_, a_j_}[ℂ𝕆[{lh___, x_i_, a_j_, rh___}_s_, more___,
    𝔼[L_, Q_, P_]]] := ℂ𝕆[{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_{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}]]
```

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.

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 > $TħD ↦ 0) &];
  U_i_[ℰ_] :=
   ℰ /. {t : cp ↦ t_i, u_U ↦ Replace[u, x_ ↦ x_i, 1]};
  U_i_[NCM[]] := 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_ ** U[] := x; U[] ** 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@{} = U[];
  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_, 0] = U[]; 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[U[]] = 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.