**The Algebras $H$ and $H^*$.** Let $q = \mathbb{e}^{\hbar\epsilon\gamma}$ and set $H = \langle a, x\rangle/([a,x] = \gamma x)$ 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

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

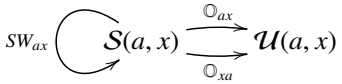
Pairing by $(a,x)^* = (b,y)$ ($\Rightarrow \langle B, A\rangle = q$) making $\langle y^l b^i, a^j x^k\rangle = \delta_{ij}\delta_{kl} j![k]_q!$ so $R = \sum \frac{y^k b^j \otimes a^j x^k}{j![k]_q!}$.

**The Algebra $QU$.** Using the Drinfel'd double procedure, $QU_{\gamma,\epsilon} := H^{*cop}\otimes H$ 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

$$S(y,b,a,x) = (-B^{-1}y, -b, -a, -A^{-1}x),$$
$$\Delta(y,b,a,x) = (y_1 + y_2B_1, b_1 + b_2, a_1 + a_2, x_1 + A_1x_2).$$

Note also that $t := \epsilon a - \gamma b$ is central and can replace $b$, and set $QU = QU_\epsilon = QU_{1,\epsilon}$.

**The 2D Lie Algebra.** One may show* that if $[a,x] = \gamma x$ then $\mathbb{e}^{\xi x}\mathbb{e}^{\alpha a} = \mathbb{e}^{\alpha a}\mathbb{e}^{\mathbb{e}^{-\gamma\alpha}\xi x}$. Ergo with




we have $\widetilde{SW}_{ax} = \mathbb{e}^{\alpha a + \mathbb{e}^{-\gamma\alpha}\xi x}$.

* Indeed $xa = (a-\gamma)x$ thus $xa^n = (a-\gamma)^n x$ thus $x\mathbb{e}^{\alpha a} = \mathbb{e}^{\alpha(a-\gamma)}x = \mathbb{e}^{-\gamma\alpha}\mathbb{e}^{\alpha a}x$ thus $x^n\mathbb{e}^{\alpha a} = \mathbb{e}^{\alpha a}(\mathbb{e}^{-\gamma\alpha})^n x^n$ thus $\mathbb{e}^{\xi x}\mathbb{e}^{\alpha a} = \mathbb{e}^{\alpha a}\mathbb{e}^{\mathbb{e}^{-\gamma\alpha}\xi x}$.

**Faddeev's Formula** (In as much as we can tell, first appeared without proof in Faddeev [Fa], rediscovered and proven in Quesne [Qu], and again with easier proof, in Zagier [Za]). With $[n]_q := \frac{q^n-1}{q-1}$, with $[n]_q! := [1]_q[2]_q\cdots[n]_q$ and with $\mathbb{e}_q^x := \sum_{n\ge 0}\frac{x^n}{[n]_q!}$, we have

$$\log \mathbb{e}_q^x = \sum_{k\ge1}\frac{(1-q)^k x^k}{k(1-q^k)} = x + \frac{(1-q)^2x^2}{2(1-q^2)} + \dots.$$

Proof. We have that $\mathbb{e}_q^x = \frac{\mathbb{e}_q^{qx} - \mathbb{e}_q^x}{qx-x}$ ("the $q$-derivative of $\mathbb{e}_q^x$ is itself"), and hence $\mathbb{e}_q^{qx} = (1+(1-q)x)\mathbb{e}_q^x$, and

$$\log \mathbb{e}_q^{qx} = \log(1+(1-q)x) + \log \mathbb{e}_q^x.$$

Writing $\log \mathbb{e}_q^x = \sum_{k\ge1} a_k x^k$ and comparing powers of $x$, we get $q^k a_k = -(1-q)^k/k + a_k$, or $a_k = \frac{(1-q)^k}{k(1-q^k)}$. □

**A Full Implementation.** ωεβ/Full

## Utilities

```
CF[sd_SeriesData] := MapAt[CF, sd, 3];
CF[ℰ_] := ExpandDenominator@ExpandNumerator@Together[
     Expand[ℰ] //. e^x_ e^y_ :> e^(x+y) /. e^x_ :> e^CF[x]];
Kδ /: Kδ_i_,j_ := If[i === j, 1, 0];
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];
𝔼[L_, Q_, P_]$k_ := 𝔼[L, Q, Series[Normal@P, {ϵ, 0, $k}]];
```

## Zip and Bind

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

```
collect[sd_SeriesData, ζ_] :=
  MapAt[collect[#, ζ] &, sd, 3];
collect[ℰ_, ζ_] := Collect[ℰ, ζ];
Zip{} [P_] := P; Zip{ζ_,ζs___} [P_] :=
 (collect[P // Zip{ζs}, ζ] /. f_. ζ^d_. :> ∂{ζ*,d} f) /. ζ* → 0
```

```
QZipζs_List@𝔼[L_, Q_, P_] :=
  Module[{ζ, z, zs, c, ys, ηs, qt, zrule, ζrule},
   zs = Table[ζ*, {ζ, ζs}];
   c = CF[Q /. Alternatives @@ (ζs ⋃ zs) → 0];
   ys = CF@Table[∂ζ (Q /. Alternatives @@ zs → 0), {ζ, ζs}];
   ηs = CF@Table[∂z (Q /. Alternatives @@ ζs → 0), {z, zs}];
   qt = CF@Inverse@Table[Kδz,ζ* - ∂z,ζ Q, {ζ, ζs}, {z, zs}];
   zrule = Thread[zs → CF[qt.(zs + ys)]];
   ζrule = Thread[ζs → ζs + ηs.qt];
   CF /@ 𝔼[L, c + ηs.qt.ys,
     Det[qt] Zipζs[P /. (zrule ⋃ ζrule)]] ];
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};
LZipζs_List@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[Kδz,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;
   CF /@ 𝔼[L2, Q2, Det[lt] e^-L2-Q2
        Zipζs[e^(L1+Q1) (P /. U2l /. zrule)]] //. l2U];
B{} [L_, R_] := L R;
B{is__} [L_, R_] := 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}}]
    ] // LZipJoin@@Table[{β_n@i, τ_n@i, a_n@i}, {i, {is}}] //
   QZipJoin@@Table[{ξ_n@i, y_n@i}, {i, {is}}]];
B_is___ [L_, R_] := B{is} [L, R];
```

## 𝔼 morphisms with domain and range.

```
B_is_List [𝔼_d1_→r1_ [L1_, Q1_, P1_], 𝔼_d2_→r2_ [L2_, Q2_, P2_]] :=
  𝔼(d1 ⋃ Complement[d2, is])→(r2 ⋃ Complement[r1, is]) @@
   B_is [𝔼 [L1, Q1, P1], 𝔼 [L2, Q2, P2]];
𝔼_d1_→r1_ [L1_, Q1_, P1_] // 𝔼_d2_→r2_ [L2_, Q2_, P2_] :=
  B_r1∩d2 [𝔼_d1→r1 [L1, Q1, P1], 𝔼_d2→r2 [L2, Q2, P2]];
𝔼_d1_→r1_ [L1_, Q1_, P1_] ≡ 𝔼_d2_→r2_ [L2_, Q2_, P2_] ^:=
  (d1 == d2) ∧ (r1 == r2) ∧ (𝔼 [L1, Q1, P1] ≡ 𝔼 [L2, Q2, P2]);
𝔼_d1_→r1_ [L1_, Q1_, P1_] 𝔼_d2_→r2_ [L2_, Q2_, P2_] ^:=
  𝔼(d1⋃d2)→(r1⋃r2) @@ (𝔼 [L1, Q1, P1] 𝔼 [L2, Q2, P2]);
𝔼_d_→r_ [L_, Q_, P_]$k_ := 𝔼_d→r @@ 𝔼 [L, Q, P]$k;
𝔼_ [ℰ___] [i_] := {ℰ}⟦i⟧;
```

## "Define" code

```
SetAttributes[Define, HoldAll];
Define[def_, defs__] := (Define[def]; Define[defs];);
```