```
SWxy[U_, kk_] :=
  SWxy[U, kk] = Block[{$U = U, $k = kk, $p = kk},
    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, $k}, {i, 0, l},
          {j, 0, l}, {k, 0, l}];
      F = fs.(bs = fs /. f_{l_,i_,j_,k_}[η] :> ϵ^l U@{y^i, a^j, x^k});
      es = Flatten[Table[Coefficient[e, b] == 0,
          {e, {F - 1_U /. η → 0, F ** G - y_U ** F - ∂_η F}},
          {b, bs}]];
      F = F /. DSolve[es, fs, η][[1]];
      E[0,
        ξ x + η y + (U /. {CU → -t η ξ, QU → η ξ (1 - T) / ℏ}),
        F + O_$k /. {e- → 1, U → Times}
       ] /. (v : η | ξ | t | T | y | a | x) → v_1
    ]];
tSWxy,i_,j_→k_ :=
  SWxy[$U, $k] /. {ξ_1 → ξ_i, η_1 → η_j, (v : t | T | y | a | x)_1 → v_k};
tSWxa,i_,j_→k_ := E[α_j a_k, e^{-γ α_j} ξ_i x_k, 1];
tSWay,i_,j_→k_ := E[α_i a_k, e^{-γ α_i} η_j y_k, 1];
```

Exponentials as needed.

Task. Define $\mathrm{Exp}_{U_i,k}[\xi, P]$ which computes $e^{\xi\,\mathbb{O}(P)}$ to ϵ^k in the algebra U_i, where ξ is a scalar, X is x_i or y_i, and P is an ϵ-dependent near-docile element, giving the answer in $\mathbb{E}$-form. Should satisfy $U@\mathrm{Exp}_{U_i,k}[\xi, P] == \mathbb{S}_U[e^{\xi x},\ x \to \mathbb{O}(P)]$.

Methodology. If $P_0 := P_{\epsilon=0}$ and $e^{\xi\,\mathbb{O}(P)} = \mathbb{O}(e^{\xi P_0} F(\xi))$, then $F(\xi=0)=1$ and we have:

$$\mathbb{O}(e^{\xi P_0}(P_0 F(\xi) + \partial_\xi F) = \mathbb{O}(\partial_\xi e^{\xi P_0} F(\xi)) =$$
$$\partial_\xi \mathbb{O}(e^{\xi P_0} F(\xi)) = \partial_\xi e^{\xi\,\mathbb{O}(P)} = e^{\xi\,\mathbb{O}(P)}\,\mathbb{O}(P) = \mathbb{O}(e^{\xi P_0} F(\xi))\,\mathbb{O}(P)$$

This is an ODE for F. Setting inductively $F_k = F_{k-1} + \epsilon^k \varphi$ we find that $F_0 = 1$ and solve for φ.

```
(* Bug: The first line is valid only if O(e^P0) == e^O(P0). *)
(* Bug: ξ must be a symbol. *)
Exp_{U_i_,0}[ξ_, P_] := Module[{LQ = Normal@P /. ϵ → 0},
   E[ξ LQ /. (x | y)_i → 0, ξ LQ /. (t | a)_i → 0, 1]];
Exp_{U_i_,k_}[ξ_, P_] := Block[{$U = U, $k = k},
  Module[{P0, φ, φs, F, j, rhs, at0, atξ},
   P0 = Normal@P /. ϵ → 0;
   φs = Flatten@Table[φ_{j1,j2,j3}[ξ], {j2, 0, k},
       {j1, 0, 2 k + 1 - j2}, {j3, 0, 2 k + 1 - j2 - j1}];
   F = Normal@Last@Exp_{U_i,k-1}[ξ, P] +
      ϵ^k φs.(φs /. φ_js__[ξ] :> Times @@ {y_i, a_i, x_i}^{js});
   rhs =
    Normal@
     Last@
      m_{i,j→i}[E[ξ P0 /. (x | y)_i → 0, ξ P0 /. (t | a)_i → 0, F + O_k]
        m_{i→j}@E[0, 0, P + O_k]];
   at0 = (# == 0) & /@
     Flatten@CoefficientList[F - 1 /. ξ → 0, {y_i, a_i, x_i}];
   atξ = (# == 0) & /@
     Flatten@CoefficientList[(∂_ξ F) + P0 F - rhs,
       {y_i, a_i, x_i}];
   E[ξ P0 /. (x | y)_i → 0, ξ P0 /. (t | a)_i → 0, F + O_k] /.
    DSolve[And @@ (at0 ⋃ atξ), φs, ξ][[1]] ]]
```

Zip and Bind

```
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];
{t*, y*, a*, x*, z*} = {τ, η, α, ξ, ζ};
{τ*, η*, α*, ξ*, ζ*} = {t, y, a, x, z};
(u_i_)* := (u*)_i;
Zip_{}[P_] := P;
Zip_{ζ_,ζs___}[P_] :=
  (Expand[P // Zip_{ζs}] /. f_. ζ^d_. :> ∂_{ζ*,d} f) /. ζ* → 0
```

QZip implements the "Q-level zips" on $\mathbb{E}(L, Q, P) = P e^{L+Q}$. Such zips regard the L variables as scalars.

```
QZip_{ζs_List,simp_}@E[L_, Q_, P_] :=
  Module[{ζ, z, zs, c, ys, ηs, qt, zrule, Q1, Q2},
   zs = Table[ζ*, {ζ, ζs}];
   c = Q /. Alternatives @@ (ζs ⋃ zs) → 0;
   ys = Table[∂_ζ (Q /. Alternatives @@ zs → 0), {ζ, ζs}];
   ηs = Table[∂_z (Q /. Alternatives @@ ζs → 0), {z, zs}];
   qt = Inverse@Table[K δ_{z,ζ*} - ∂_{z,ζ} Q, {ζ, ζs}, {z, zs}];
   zrule = Thread[zs → qt.(zs + ys)];
   Q2 = (Q1 = c + ηs.zs /. zrule) /. Alternatives @@ zs → 0;
   simp /@ E[L, Q2, Det[qt] e^{-Q2} Zip_ζs[e^{Q1} (P /. zrule)]] ];
QZip_{ζs_List} := QZip_{ζs,CF};
```

LZip implements the "L-level zips" on $\mathbb{E}(L, Q, P) = P e^{L+Q}$. Such zips regard all of $P e^{Q}$ as a single"P". Here the z's are t and α and the ζ's are τ and a.

```
LZip_{ζs_List,simp_}@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[K δ_{z,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 /. T2t /. zrule) /. Alternatives @@ zs → 0;
   simp /@
    E[L2, Q2, Det[lt] e^{-L2-Q2}
      Zip_ζs[e^{L1+Q1} (P /. T2t /. zrule)]] //. t2T ];
LZip_{ζs_List} := LZip_{ζs,CF};
Bind_{}[L_, R_] := L R;
Bind_{is__}[L_E, R_E] := Module[{n},
   Times[
     L /. Table[(v : T | t | a | x | y)_i → v_{n@i}, {i, {is}}],
     R /. Table[(v : τ | α | ξ | η)_i → v_{n@i}, {i, {is}}]
    ] // LZip_{Flatten@Table[{τ_{n@i}, a_{n@i}}, {i, {is}}]} //
   QZip_{Flatten@Table[{ξ_{n@i}, y_{n@i}}, {i, {is}}]} ];
B_{L_List} := Bind_L; B_{is___} := Bind_{{is}};
Bind[ℰ_E] := ℰ;
Bind[Ls__, ζs_List, R_] := Bind_ζs[Bind[Ls], R];
```

Tensorial Representations

```
tη = t1 = E[0, 0, 1 + O_$k];
tm_{i_,j_→k_} := Module[{tk},
   E[(τ_i + τ_j) t_k + α_i a_k + α_j a_k, η_i y_k + ξ_j x_k, 1]
    (tSWxy,i,j→tk /. {t_tk → t_k, T_tk → T_k, y_tk → e^{-γ α_i} y_k,
        a_tk → a_k, x_tk → e^{-γ α_j} x_k})];
m_{j_→k_}[ℰ_E] := ℰ ~B_{j,k}~ tm_{j,k→k};
```

```
tm_{1,2→3}
```

Video and more at http://www.math.toronto.edu/~drorbn/Talks/Matemale-1804/