```
QZip_ξs_List@𝔼[L_, Q_, P_] :=
  PP_QZip@Module[{ξ, z, zs, c, ys, ηs, qt, zrule, ξrule, out},
    zs = Table[ξ*, {ξ, ξs}];
    c = CF[Q /. Alternatives @@ (ξs ⋃ zs) → 0];
    ys = CF@Table[∂_ξ (Q /. Alternatives @@ zs → 0),
       {ξ, ξs}];
    ηs = CF@Table[∂_z (Q /. Alternatives @@ ξs → 0), {z, zs}];
    qt = CF@Inverse@Table[Kδ_{z,ξ*} - ∂_{z,ξ}Q, {ξ, ξs}, {z, zs}];
    zrule = Thread[zs → CF[qt.(zs + ys)]];
    ξrule = Thread[ξs → ξs + ηs.qt];
    CF /@ 𝔼[L, c + ηs.qt.ys,
      Det[qt] Zip_ξs[P /. (zrule ⋃ ξrule)]] ];
```

LZip implements the "L-level zips" on $\mathbb{E}(L, Q, P) = P\mathscr{e}^{L+Q}$. Such zips regard all of $P\mathscr{e}^{Q}$ as a single"P". Here the z's are b and α and the ζ's are β and a.

```
LZip_ξs_List@𝔼[L_, Q_, P_] :=
  PP_LZip@Module[{ξ, z, zs, Zs, c, ys, ηs, lt, zrule,
     Zrule, ξrule, Q1, EEQ, EQ},
    zs = Table[ξ*, {ξ, ξs}];
    Zs = zs /. {b → B, t → T, α → 𝔄};
    c = L /. Alternatives @@ (ξs ⋃ zs) → 0 /.
      Alternatives @@ Zs → 1;
    ys = Table[∂_ξ (L /. Alternatives @@ zs → 0), {ξ, ξs}];
    ηs = Table[∂_z (L /. Alternatives @@ ξs → 0), {z, zs}];
    lt = Inverse@Table[Kδ_{z,ξ*} - ∂_{z,ξ}L, {ξ, ξs}, {z, zs}];
    zrule = Thread[zs → lt.(zs + ys)];
    Zrule = Join[zrule,
      zrule /.
       r_Rule :↦ ((U = r[[1]] /. {b → B, t → T, α → 𝔄}) →
          (U /. U2l /. r //. l2U))];
    ξrule = Thread[ξs → ξs + ηs.lt];
    Q1 = Q /. (Zrule ⋃ ξrule);
    EEQ[ps___] :=
     EEQ[ps] =
      PP_"EEQ"@(CF[ℯ^-Q1 D_Thread[{zs,{ps}}][ℯ^Q1]] /.
         {Alternatives @@ zs → 0, Alternatives @@ Zs → 1});
    CF@𝔼[c + ηs.lt.ys,
      Q1 /. {Alternatives @@ zs → 0, Alternatives @@ Zs → 1},
      Det[lt]
       (Zip_ξs[(EQ @@ zs) (P /. (Zrule ⋃ ξrule))] /.
          Derivative[ps___][EQ][___] :↦ EEQ[ps] /.
        _EQ → 1) ]];
```

```
B_{}[L_, R_] := L R;
B_{is__}[L_𝔼, R_𝔼] := PP_B@Module[{n},
    Times[
      L /. Table[(ν : b | B | t | T | a | x | y)_i → v_{n@i},
        {i, {is}}],
      R /. Table[(ν : β | τ | α | 𝔄 | ξ | η)_i → v_{n@i}, {i, {is}}]
     ] // LZip_Join@@Table[{β_{n@i}, τ_{n@i}, a_{n@i}}, {i, {is}}] //
     QZip_Join@@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_dr_[L_, Q_, P_]_$k_ := 𝔼_dr @@ 𝔼[L, Q, P]_$k;
𝔼_[ℰ___][i_] := {ℰ}[[i]];
```

𝔼[Λ]

```
𝔼_dr_[Λ_] :=
 CF@Module[{L, Λ0 = Limit[Λ, ϵ → 0]},
   𝔼_dr[L = Λ0 /. (η | y | ξ | x)_ → 0, Λ0 - L, ℯ^(Λ-Λ0)]_$k /. l2U]
```

Exponentials as needed.

Task. Define $\mathrm{Exp}_{m,i,k}[P]$ to compute $\mathscr{e}^{\mathbb{O}(P)}$ to ϵ^k in the using the $m_{i,i\to i}$ multiplication, where P is an ϵ-dependent near-docile element, giving the answer in $\mathbb{E}$-form.

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

$$\mathbb{O}\left(\mathscr{e}^{\lambda P_0}(P_0 F(\lambda) + \partial_\lambda F)\right) = \mathbb{O}\left(\partial_\lambda \mathscr{e}^{\lambda P_0} F(\lambda)\right) = \partial_\lambda \mathbb{O}\left(\mathscr{e}^{\lambda P_0} F(\lambda)\right) = \partial_\lambda \mathscr{e}^{\lambda\,\mathbb{O}(P)} = \mathscr{e}^{\lambda\,\mathbb{O}(P)}\,\mathbb{O}(P) = \mathbb{O}\left(\mathscr{e}^{\lambda P_0} F(\lambda)\right)\mathbb{O}(P)$$

This is a linear 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). *)
Exp_m_,i_,0[P_] := Module[{LQ = Normal@P /. ϵ → 0},
   𝔼[LQ /. (x | y)_i → 0, LQ /. (b | a | t)_i → 0, 1] ];
Exp_m_,i_,k_[P_] := Block[{$k = k},
  Module[{P0, λ, φ, φs, F, j, rhs, eqn, pows, at0, atλ},
   P0 = Normal@P /. ϵ → 0;
   F = Normal@Last@Exp_{m,i,k-1}[λ P];
   While[
    rhs =
     m_{i,j→i}[
       𝔼_{}→{i}[λ P0 /. (x | y)_i → 0, λ P0 /. (b | a | t)_i → 0,
          F]_k so_{i→j} @ 𝔼_{}→{i}[0, 0, P]_k] // Last // Normal;
    eqn = CF[(∂_λ F) + P0 F - rhs];
    eqn =!= 0, (*do*)
    pows = First /@ CoefficientRules[eqn, {y_i, b_i, a_i, x_i}];
    F += Sum[ϵ^k φ_js[λ] Times @@ {y_i, b_i, a_i, x_i}^js,
      {js, pows}];
    rhs =
     m_{i,j→i}[
       𝔼_{}→{i}[λ P0 /. (x | y)_i → 0, λ P0 /. (b | a | t)_i → 0,
          F]_k so_{i→j} @ 𝔼_{}→{i}[0, 0, P]_k] // Last // Normal;
    eqn = CF[(∂_λ F) + P0 F - rhs];
    φs = Table[φ_js[λ], {js, pows}];
    at0 = Table[φ_js[0] == 0, {js, pows}];
    atλ = (# == 0) & /@
      (pows /. CoefficientRules[eqn, {y_i, b_i, a_i, x_i}]);
    F = F /. DSolve[And @@ (at0 ⋃ atλ), φs, λ][[1]]
   ];
   𝔼_{}→{i}[P0 /. (x | y)_i → 0, P0 /. (b | a | t)_i → 0,
    F + O[ϵ]^(k+1) /. λ → 1]] ]
```

"Define" Code