Pensieve header: The "Speedy" engine, from pensieve://Projects/SL2Portfolio2/.

Program

The "Speedy" Engine

Program

Internal Utilities

Program

Canonical Form:

Program

In[∘]:=
```
CCF[ℰ_] := PP_CCF@ExpandDenominator@ExpandNumerator@PP_Together@Together[PP_Exp[
      Expand[ℰ] //. e^x_ e^y_ :> e^(x+y) /. e^x_ :> e^CCF[x]]];
CF[ℰ_List] := CF /@ ℰ;
CF[sd_SeriesData] := MapAt[CF, sd, 3];
CF[ℰ_] := PP_CF@Module[
    {vs = Cases[ℰ, (y | b | t | a | x | η | β | τ | α | ξ)_, ∞] ⋃ {y, b, t, a, x, η, β, τ, α, ξ}},
    Total[CoefficientRules[Expand[ℰ], vs] /. (ps_ → c_) :> CCF[c] (Times @@ vs^ps)]
   ];
CF[ℰ_𝔼] := CF /@ ℰ; CF[𝔼_sp___[ℰs___]] := CF /@ 𝔼_sp[ℰs];
```

Program

The Kronecker δ:

Program

In[∘]:=
```
Kδ /: Kδ_(i_,j_) := If[i === j, 1, 0];
```

Program

Equality, multiplication, and degree-adjustment of perturbed Gaussians; $\mathbb{E}[L, Q, P]$ stands for $e^{L+Q} P$:

Program

In[∘]:=
```
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}]];
```

Program

Zip and Bind

Program

Variables and their duals:

Program

In[∘]:=
```
{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;
```

Program

```
In[]:= 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)};
```

Program

```
In[]:= D_b[f_] := ∂_b f - ℏ γ B ∂_B f; D_(b_i_)[f_] := ∂_(b_i) f - ℏ γ B_i ∂_(B_i) f;
D_t[f_] := ∂_t f + ℏ T ∂_T f; D_(t_i_)[f_] := ∂_(t_i) f + ℏ T_i ∂_(T_i) f;
D_α[f_] := ∂_α f + γ 𝒜 ∂_𝒜 f; D_(α_i_)[f_] := ∂_(α_i) f + γ 𝒜_i ∂_(𝒜_i) f;
D_v_[f_] := ∂_v f; D_{v_,0}[f_] := f; D_{}[f_] := f; D_{v_,n_Integer}[f_] := D_v[D_{v,n-1}[f]];
D_{L_List,Ls___}[f_] := D_{Ls}[D_L[f]];
```

Program

Finite Zips:

Program

```
In[]:= collect[sd_SeriesData, ζ_] := MapAt[collect[#, ζ] &, sd, 3];
collect[ℰ_, ζ_] := PP_Collect@Collect[ℰ, ζ];
Zip_{}[P_] := P;
Zip_ζs_[Ps_List] := Zip_ζs /@ Ps;
Zip_{ζ_,ζs___}[P_] := PP_Zip[
   (collect[P // Zip_{ζs}, ζ] /. f_. ζ^d_. :→ (D_{ζ*,d}[f])) /. ζ* → 0 /.
    ((ζ* /. {b → B, t → T, α → 𝒜}) → 1)]
```

Program

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

$$\left\langle P(z_i, \zeta^j) e^{c+\eta^i z_i + y_j \zeta^j + q_j^i z_i \zeta^j} \right\rangle = |\tilde{q}| \left\langle P(z_i, \zeta^j) e^{c+\eta^i z_i} \Big|_{z_i \to \tilde{q}_i^k (z_k + y_k)} \right\rangle$$
$$= |\tilde{q}| e^{c+\eta^i \tilde{q}_i^k y_k} \left\langle P\left(\tilde{q}_i^k (z_k + y_k), \zeta^j + \eta^i \tilde{q}_i^j\right) \right\rangle.$$

Program

```
QZip_ζs_List@𝔼[L_, Q_, P_] := PP_QZip@Module[{ζ, z, zs, c, ys, ηs, qt, zrule, ζrule, out},
    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[Kδ_(z,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] Zip_ζs[P /. (zrule ⋃ ζrule)]] ];
```

Program

Upper to lower and lower to Upper:

Program

In[∘]:=
```
U2l = {B_i_^p_. → e^(-p ħ ϒ b_i), B^p_. → e^(-p ħ ϒ b), T_i_^p_. → e^(p ħ t_i), T^p_. → e^(p ħ t), 𝒜_i_^p_. → e^(p ϒ α_i), 𝒜^p_. → e^(p ϒ α)};
l2U = {e^(c_. b_i_+d_.) :→ B_i^(-c/(ħ ϒ)) e^d, e^(c_. b+d_.) :→ B^(-c/(ħ ϒ)) 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/ϒ) e^d, e^(c_. α+d_.) :→ 𝒜^(c/ϒ) e^d,
   e^ℰ_ :→ e^(Expand@ℰ)};
```

Program

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

Program

```
LZip_ζs_List@𝔼[L_, Q_, P_] :=
  PP_LZip@Module[{ζ, z, zs, Zs, c, ys, ηs, lt, zrule, Zrule, ζrule, Q1, EEQ, EQ},
    zs = Table[ζ*, {ζ, ζs}];
    Zs = zs /. {b → B, t → T, α → 𝒜};
    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[Kδ_(z,ζ*) - ∂_(z,ζ) L, {ζ, ζs}, {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))];
    ζrule = Thread[ζs → ζs + ηs.lt];
    Q1 = Q /. (Zrule ⋃ ζrule);
    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_ζs[(EQ @@ zs) (P /. (Zrule ⋃ ζrule))] /.
         Derivative[ps___][EQ][___] :→ EEQ[ps] /. _EQ → 1) ]];
```

Program

In[∘]:=
```
B_{}[L_, R_] := L R;
B_{is__}[L_𝔼, R_𝔼] := PP_B@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}}]
     ] // 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];
```

Program

𝔼 morphisms with domain and range.

Program

In[∘]:=

```
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〛;
```

Program

“Define” Code

Program

Define[lhs = rhs,   ...] defines the lhs to be rhs, except that rhs is computed only once for each value of $k. Fancy Mathematica not for the faint of heart. Most readers should ignore.

Program

In[∘]:=

```
SetAttributes[Define, HoldAll];
Define[def_, defs__] := (Define[def]; Define[defs];);
Define[op_{is__} = ℰ_] := Module[{SD, ii, jj, kk, isp, nis, nisp, sis}, Block[{i, j, k},
    ReleaseHold[Hold[
        SD[op_{nisp,$k_Integer}, PP_{Boot}@Block[{i, j, k}, op_{isp,$k} = ℰ; op_{nis,$k}]];
        SD[op_{isp}, op_{{is},$k}]; SD[op_{sis__}, op_{{sis}}];
      ] /. {SD → SetDelayed,
        isp → {is} /. {i → i_, j → j_, k → k_},
        nis → {is} /. {i → ii, j → jj, k → kk},
        nisp → {is} /. {i → ii_, j → jj_, k → kk_}
      }] ]]
```