Aw-Calculus Programs for the WKO4 Paper

Pensieve header: Aw-calculus programs for the WKO4 paper.

Non-current versions of this package are available at http://drorbn.net/AcademicPensieve/Projects/WKO4/Archive/.

“d” is “ht”: along tube strands, heads appear before tails.

MORE: Complete the implementation of all relevant operators.

Welcome / Global Definitions

```
BeginPackage["AwCalculus`", {"FreeLie`"}];
Print["AwCalculus` implements / extends ",
  Sort@{"*", "**", "≡", dA, dc, deg, dm, dS, dΔ, dη, dσ, El, Es, hA, hm, hS, hΔ, hη,
    hσ, tA, tha, tm, tS, tΔ, tη, tσ, Γ, Λ, RandomEsSeries, RandomElSeries},
  "."];
Print["AwCalculus` is in the public domain. Dror Bar-Natan is committed to support
    it within reason until July 15, 2022. This is version 150909."];
Begin["`Private`"];
```

Utilities

```
deg /: (h_)^deg := DegreeScale[h];
```

The AT Presentation E_l of A^w

```
El[λ_, ω_][d_] := El[λ[d], ω[d]];
El /: El[λ1_, ω1_] ≡ El[λ2_, ω2_] := (λ1 ≡ λ2) && (ω1 ≡ ω2);
El /: El[λ1_, ω1_] El[λ2_, ω2_] /; Support[λ1] ⋂ Support[λ2] == {} :=
 El[λ1 ⋃ λ2, ω1 + ω2];
```

ElStackingDef

```
El /: El[λ1_, ω1_] ** El[λ2_, ω2_] /; Support[λ1] == Support[λ2] :=
  El[BCH_tb[λ1, λ2], e^(-D_λ2)[ω1] + ω2];
```

```
El /: El[λ1_, ω1_] ** El[λ2_, ω2_] := NonCommutativeMultiply[
  El[λ1 ⋃ ((# → LS[0]) & /@ ⟨Complement[Support@λ2, Support@λ1]⟩), ω1],
  El[λ2 ⋃ ((# → LS[0]) & /@ ⟨Complement[Support@λ1, Support@λ2]⟩), ω2]
 ]
```

```
El /: El[λ_, ω_]^-1 := El[-λ, -e^D_λ[ω]]
```

```
El[λ_, ω_] // dη[s_] :=
  El[(λ \ s) // LieMorphism[LW[s] → 0], ω // LieMorphism[LW[s] → 0]];
dη /: dη^a_- := dη[a];
```

EldA

```
El[λ_, ω_] // dA := El[-λ, e^D_λ[ω] - j[λ]];
```

```
ξ_El // DegreeScale[h_] := DegreeScale[h] /@ ξ;
ξ_El // dS := ξ // dA // (-1)^deg;
```

EldDelta

```
El[λ_, ω_] // dΔ[a_, b_, c_] := El[
  (λ \ a) ⋃ ⟨b → λ_a, c → λ_a⟩ // LieMorphism[LW@a → LW@b + LW@c],
  ω // LieMorphism[LW@a → LW@b + LW@c]]
```

The Split Presentation E_s of A^w

```
Es /: Es[λ1_, ω1_] ≡ Es[λ2_, ω2_] := (λ1 ≡ λ2) && (ω1 ≡ ω2);
Es[λ_, ω_][d_] := Es[λ[d], ω[d]];
```

EsSampleDefs

```
Es /: Es[λ1_, ω1_] Es[λ2_, ω2_] /; Support[λ1] ⋂ Support[λ2] == {} :=
  Es[λ1 ⋃ λ2, ω1 + ω2];
Es[λ_, ω_] // hm[x_, y_, z_] := Es[λ // hm[x, y, z], ω];
Es[λ_, ω_] // tm[u_, v_, w_] :=
  LieMorphism[LW@u → LW@w, LW@v → LW@w] /@ Es[λ, ω];
Es[λ_, ω_] // tha[u_, x_] := Es[λ // RC_u[λ_x], (ω + J_u[λ_x]) // RC_u[λ_x]];
```

```
tσ[us_List → vs_List][ser_LieSeries | ser_CWSeries | ser_AngleBracket] :=
  ser // LieMorphism[Thread[(LW /@ us) → (LW /@ vs)]];
tσ[u_, v_] := tσ[{u} → {v}];
tσ[us_List → vs_List][ξ_Es] := tσ[us → vs] /@ ξ;
hσ[xs_List → ys_List][λ_AngleBracket] :=
  Union[λ \ xs, ⟨Thread[ys → Table[λx, {x, xs}]]⟩];
hσ[x_, y_] := hσ[{x} → {y}];
hσ[xs_List → ys_List][Es[λ_, ω_]] := Es[λ // hσ[xs → ys], ω];
dσ[as_List → bs_List][ξ_] := ξ // tσ[as → bs] // hσ[as → bs];
dσ[a_, b_][ξ_] := ξ // tσ[a, b] // hσ[a, b];
```

Esdm

```
ξ_Es // dm[a_, b_, c_] := ξ // tha[a, b] // tm[a, b, c] // hm[a, b, c];
```

```
tm[u_, v_, w_][λ_AngleBracket] := λ // LieMorphism[LW@u → LW@w, LW@v → LW@w];
hm[x_, y_, z_][λ_AngleBracket] := Union[λ \ {x, y}, ⟨z → BCH[λx, λy]⟩];
tha[u_LW, x_][λ_AngleBracket] := λ // RCu[λx];
dm[a_, b_, rest__, c_][ξ_] := ξ // dm[b, rest, b] // dm[a, b, c];
```

```
bar[LW[a_]] := LW[bar[a]];
```

EsNCM

```
Es /: Es[λ1_, ω1_] ** Es[λ2_, ω2_] /; Support[λ1] == Support[λ2] := Module[
   {S = Support[λ1], ξ, a},
   ξ = Es[λ1, ω1] * (Es[λ2, ω2] // dσ[S → (bar /@ S)]);
   Table[ξ = ξ // dm[a, bar[a], a], {a, S}] // Last
  ];
```

```
Es /: Es[λ1_, ω1_] ** Es[λ2_, ω2_] := NonCommutativeMultiply[
   Es[λ1 ⋃ ((# → LS[0]) & /@ ⟨Complement[Support@λ2, Support@λ1]⟩), ω1],
   Es[λ2 ⋃ ((# → LS[0]) & /@ ⟨Complement[Support@λ1, Support@λ2]⟩), ω2]
  ];
Es /: (ξ_Es)^-1 := Γ[Λ[ξ]^-1];
```

```
tA[u_][expr_] := expr;
hA[x_][Es[λ_, ω_]] := Es[Union[λ \ x, ⟨x → -λx⟩], ω];
dA[a_][μ_] := μ // hA[a] // tha[LW@a, a];
dA[a_, rest__][μ_] := μ // dA[a] // dA[rest];
Es[λ_, ω_] // dA := Es[λ, ω] // (dA @@ Support[λ])
```

```
tS[u_][λ_AngleBracket] :=
  ⟨Table[x → LieMorphism[LW@u → -LW@u][λ_x], {x, Support[λ]}]⟩;
tS[u_][Es[λ_, ω_]] := Es[λ // tS[u], ω // LieMorphism[LW@u → -LW@u]];
hS[x_][Es[λ_, ω_]] := Es[Union[λ \ x, ⟨x → -λ_x⟩], ω];
dS[a_][μ_] := μ // tS[a] // hS[a] // tha[LW@a, a];
dS[a_, rest__][μ_] := μ // dS[a] // dS[rest];
Es[λ_, ω_] // dS := Es[λ, ω] // (dS @@ Support[λ])
```

```
ξ_Es // DegreeScale[h_] := DegreeScale[h] /@ ξ;
```

```
Es[λ_, ω_] // hη[s_] := Es[λ \ s, ω];
hη /: hη^x_ := hη[x];
Es[λ_, ω_] // tη[u_] := LieMorphism[LW@u → 0] /@ Es[λ, ω];
tη /: tη^u_ := tη[u];
```

```
Es[λ_, ω_] // hΔ[x_, y_, z_] := Es[(λ \ x) ⋃ ⟨y → λ_x, z → λ_x⟩, ω];
λ_AngleBracket // tΔ[u_, v_, w_] := λ // LieMorphism[LW@u → LW@v + LW@w];
ω_CWSeries // tΔ[u_, v_, w_] := ω // LieMorphism[LW@u → LW@v + LW@w];
Es[λ_, ω_] // tΔ[u_, v_, w_] := tΔ[u, v, w] /@ Es[λ, ω];
Es[λ_, ω_] // dΔ[a_, b_, c_] := Es[λ, ω] // tΔ[a, b, c] // hΔ[a, b, c];
```

```
Es[λ_, ω_] // dc[a_] := Es[λ, ω] // hS[a] // tha[a, a] // hS[a] // hη[a];
```

σ

```
ξ_Es // σ[s___List] := Module[{ξ1},
  ξ1 = ξ // dσ[Range[Length@{s}] → First /@ {s}];
  Do[
   ξ1 = ξ1 // dΔ[{s}⟦i, 1⟧, {s}⟦i, 1⟧, {s}⟦i, j⟧],
   {i, Length@{s}}, {j, 2, Length@{s}⟦i⟧}
  ];
  ξ1
 ]
```

The El ↔ Es Conversions

```
Γ[El[λ_, ω_]] := Es[Γ[λ], ω];
Λ[Es[λ_, ω_]] := El[Λ[λ], ω];
```

Random Series

```
RandomElSeries[seed_, S_List] := (SeedRandom[seed];
   El[⟨Table[a → RandomLieSeries[S], {a, S}]⟩, RandomCWSeries[S]]);
RandomEsSeries[seed_, S_List] := (SeedRandom[seed];
   Es[⟨Table[a → RandomLieSeries[S], {a, S}]⟩, RandomCWSeries[S]]);
```

Epilog

```
End[]; EndPackage[];
```