

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, E1, Es, hA, hm, hS, hΔ, hη,
hσ, tA, tha, tm, tS, tΔ, tη, tσ, Γ, Δ, RandomEsSeries, RandomE1Series},
"."];
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_I of A^W

```
E1[λ_, ω_][d_] := E1[λ[d], ω[d]];
E1 /: E1[λ1_, ω1_] ≡ E1[λ2_, ω2_] := (λ1 ≡ λ2) && (ω1 ≡ ω2);
E1 /: E1[λ1_, ω1_] E1[λ2_, ω2_] /; Support[λ1] ∩ Support[λ2] == {} :=
E1[λ1 ∪ λ2, ω1 + ω2];
```

E1StackingDef

```
E1 /: E1[λ1_, ω1_] ** E1[λ2_, ω2_] /; Support[λ1] == Support[λ2] :=
E1[BCHtb[λ1, λ2], e-Dλ2[ω1] + ω2];
```

```

E1 /: E1[λ1_, ω1_] ** E1[λ2_, ω2_] := NonCommutativeMultiply[
  E1[λ1 ∪ ((# → LS[0]) & /@ Complement[Support@λ2, Support@λ1]), ω1],
  E1[λ2 ∪ ((# → LS[0]) & /@ Complement[Support@λ1, Support@λ2]), ω2]
]

E1 /: E1[λ_, ω_]^-1 := E1[-λ, -eDλ[ω]]

E1[λ_, ω_] // dη[s_] :=
  E1[(λ \ s) // LieMorphism[LW[s] → 0], ω // LieMorphism[LW[s] → 0]];
dη /: dηa := dη[a];

```

EidA

```

E1[λ_, ω_] // dA := E1[-λ, eDλ[ω] - j[λ]];

ξ_E1 // DegreeScale[h_] := DegreeScale[h] /@ ξ;
ξ_E1 // dS := ξ // dA // (-1)deg;

```

EidDelta

```

E1[λ_, ω_] // dΔ[a_, b_, c_] := E1[
  (λ \ a) ∪ ⟨b → λa, c → λa⟩ // 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 // RCu[λx], (ω + Ju[λx]) // RCu[λx]];

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[λ_x, λ_y]⟩];
tha[u_LW, x_][λ_AngleBracket] := λ // RC_u[λ_x];
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[Δ[ξ]-1];

```

```

tA[u_][expr_] := expr;
hA[x_][Es[λ_, ω_]] := Es[Union[λ \ x, ⟨x → -λ_x⟩], 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 /: hηx := hη[x];
Es[λ_, ω_] // tη[u_] := LieMorphism[LW@u → 0] /@ Es[λ, ω];
tη /: tηu := 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 // do[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 E1 ↔ Es Conversions

```

Γ[E1[λ_, ω_]] := Es[Γ[λ], ω];
Λ[Es[λ_, ω_]] := E1[Λ[λ], ω];

```

Random Series

```

RandomE1Series[seed_, S_List] := (SeedRandom[seed];
  E1[⟨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[];

```