Loading packages

In[ ]:=
```
<< KnotTheory`
Get["C:\\drorbn\\AcademicPensieve\\People\\Frohlich\\221117/RVT.m"] (* RVT-conversion program was
```

```
Loading KnotTheory` version of February 2, 2020, 10:53:45.2097.
Read more at http://katlas.org/wiki/KnotTheory.
```

Formatting

In[ ]:=
```
Format[gdo_GDO] := Subsuperscript[𝔼, Row[{gdo // getCO, ",", gdo // getCC}],
    Row[{gdo // getDO, ",", gdo // getDC}]][gdo // getL, gdo // getQ, gdo // getP];
Format[pg_PG] := 𝔼[pg // getL, pg // getQ, pg // getP];

SubscriptFormat[v_] := (Format[v[i_]] := Subscript[v, i]);

SubscriptFormat /@ {y, b, t, a, x, η, β, α, ξ, A, B, T};
```

In[ ]:=
```
γ = 1; ℏ = 1; $k = 0;
```

In[ ]:=
```
setValue[value_, obj_, coord_] := Module[{b = Association @@ obj}, b[coord] = value;
  Head[obj] @@ Normal@b]
```

PG[“L”->L, “Q”->Q, “P”->P]=Perturbed Gaußian Pe^{L+Q}

In[ ]:=
```
fromE[e_𝔼] := toPG @@ e /.
  Subscript[(v : y | b | t | a | x | B | T | η | β | τ | α | ξ | A), i_] → v[i]
```

In[ ]:=
```
toPG[L_, Q_, P_] := PG["L" → L, "Q" → Q, "P" → P]

δ[i_, j_] := If[SameQ[i, j], 1, 0]

getL[pg_PG] := Lookup[Association @@ pg, "L", 0]
getQ[pg_PG] := Lookup[Association @@ pg, "Q", 0]
getP[pg_PG] := Lookup[Association @@ pg, "P", 1]

setL[L_][pg_PG] := setValue[L, pg, "L"];
setQ[Q_][pg_PG] := setValue[Q, pg, "Q"];
setP[P_][pg_PG] := setValue[P, pg, "P"];

applyToL[f_][pg_PG] := pg // setL[pg // getL // f]
applyToQ[f_][pg_PG] := pg // setQ[pg // getQ // f]
applyToP[f_][pg_PG] := pg // setP[pg // getP // f]
```

In[ ]:=
```
CCF[e_] := ExpandDenominator@
   ExpandNumerator@Together[Expand[e] //. E^x_ E^y_ :> E^(x + y) /. E^x_ :> E^CCF[x]];
CF[sd_SeriesData] := MapAt[CF, sd, 3];
CF[e_] := Module[{vs = Union[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_PG] := e // applyToL[CF] // applyToQ[CF] // applyToP[CF]
```

In[ ]:=
```
PG /: Congruent[pg1_PG, pg2_PG] := And[CF[getL@pg1 == getL@pg2],
  CF[getQ@pg1 == getQ@pg2], CF[Normal[getP@pg1 - getP@pg2] == 0]]

PG /: pg1_PG * pg2_PG :=
 toPG[getL@pg1 + getL@pg2, getQ@pg1 + getQ@pg2, getP@pg1 * getP@pg2]

setEpsilonDegree[k_Integer][pg_PG] := setP[Series[Normal@getP@pg, {ϵ, 0, k}]][pg]
```

In[ ]:=
```
ddsl2vars = {y, b, t, a, x};
ddsl2varsDual = {η, β, τ, α, ξ};

Evaluate[Dual /@ ddsl2vars] = ddsl2varsDual;
Evaluate[Dual /@ ddsl2varsDual] = ddsl2vars;
Dual@z = ζ;
Dual@ζ = z;

Dual[u_[i_]] := Dual[u][i]

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^expr_ :> E^Expand@expr};
```

Differentiation

In[ ]:=
```
DD[f_, b] := D[f, b] - ħ γ B D[f, B];
DD[f_, b[i_]] := D[f, b[i]] - ħ γ B[i] D[f, B[i]];

DD[f_, t] := D[f, t] - ħ T D[f, T];
DD[f_, t[i_]] := D[f, t[i]] - ħ T[i] D[f, T[i]];

DD[f_, α] := D[f, α] + γ A D[f, A];
DD[f_, α[i_]] := D[f, α[i]] + γ A[i] D[f, A[i]];

DD[f_, v_] := D[f, v];
DD[f_, {v_, 0}] := f;
DD[f_, {}] := f;
DD[f_, {v_, n_Integer}] := DD[DD[f, v], {v, n - 1}];
DD[f_, {l_List, ls___}] := DD[DD[f, l], {ls}];
```

Finite zips

In[ ]:=
```
collect[sd_SeriesData, ξ_] := MapAt[collect[#, ξ] &, sd, 3];
collect[expr_, ξ_] := Collect[expr, ξ];

Zip[{}][P_] := P;
Zip[ξs_List][Ps_List] := Zip[ξs] /@ Ps;
Zip[{ξ_, ξs___}][P_] := (collect[P // Zip[{ξs}], ξ] /.
     f_. ξ^d_. :→ DD[f, {Dual[ξ], d}]) /.
   Dual[ξ] → 0 /.
  ((Dual[ξ] /. {b → B, t → T, α → A}) → 1)
```

Q-zips

In[ ]:=
```
QZip[ζs_List][pg_PG] :=
 Module[{Q, P, ζ, z, zs, c, ys, ηs, qt, zrule, ζrule}, zs = Dual /@ ζs;
  Q = pg // getQ;
  P = pg // getP;
  c = CF[Q /. Alternatives @@ Union[ζs, zs] → 0];
  ys = CF /@ Table[D[Q, ζ] /. Alternatives @@ zs → 0, {ζ, ζs}];
  ηs = CF /@ Table[D[Q, z] /. Alternatives @@ ζs → 0, {z, zs}];
  qt = CF /@ # & /@ (Inverse@Table[δ[z, Dual[ζ]] - D[Q, z, ζ], {ζ, ζs}, {z, zs}]);
  zrule = Thread[zs → CF /@ (qt.(zs + ys))];
  ζrule = Thread[ζs → ζs + ηs.qt];
  CF@setQ[c + ηs.qt.ys]@setP[Det[qt] Zip[ζs][P /. Union[zrule, ζrule]]]@pg]
```

L - zips

In[ ]:=
```
LZip[ξs_List][pg_PG] := Module[{L, Q, P, ξ, z, zs, Zs,
    c, ys, ηs, lt, zrule, Zrule, ξrule, Q1, EEQ, EQ, U}, zs = Dual /@ ξs;
   {L, Q, P} = Through[{getL, getQ, getP}@pg];
   Zs = zs /. {b → B, t → T, α → A};
   c = CF[L /. Alternatives @@ Union[ξs, zs] → 0 /. Alternatives @@ Zs → 1];
   ys = CF /@ Table[D[L, ξ] /. Alternatives @@ zs → 0, {ξ, ξs}];
   ηs = CF /@ Table[D[L, z] /. Alternatives @@ ξs → 0, {z, zs}];
   lt = CF /@ # & /@ Inverse@Table[δ[z, Dual[ξ]] - D[L, z, ξ], {ξ, ξs}, {z, zs}];
   zrule = Thread[zs → CF /@ (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 /. Union[Zrule, ξrule];
   EEQ[ps___] := EEQ[ps] = (CF[E^-Q1 DD[E^Q1, Thread[{zs, {ps}}]]] /.
        {Alternatives @@ zs → 0, Alternatives @@ Zs → 1]);
   CF@toPG[c + ηs.lt.ys, Q1 /. {Alternatives @@ zs → 0, Alternatives @@ Zs → 1},
     Det[lt] (Zip[ξs][(EQ @@ zs) (P /. Union[Zrule, ξrule])] /.
         Derivative[ps___][EQ][___] :→ EEQ[ps] /. _EQ → 1)]]
```

Pairing of PG objects

In[ ]:=
```
Pair[{}][L_PG, R_PG] := L R;
Pair[is_List][L_PG, R_PG] :=
 Module[{n}, Times[L /. ((v : b | B | t | T | a | x | y)[#] → v[n@#] & /@ is),
      R /. ((v : β | τ | α | A | ξ | η)[#] → v[n@#] & /@ is)] //
     LZip[Join @@ Table[Through[{β, τ, a}[n@i]], {i, is}]] //
    QZip[Join @@ Table[Through[{ξ, y}[n@i]], {i, is}]]]
```

GDO=Gaußian Differential Operator (PG with domain and range)

In[ ]:=
```
toGDO[do_List, dc_List, co_List, cc_List, L_, Q_, P_] :=
 GDO["do" → do, "dc" → dc, "co" → co, "cc" → cc, "PG" → toPG[L, Q, P]]

toGDO[do_List, dc_List, co_List, cc_List, pg_PG] :=
 GDO["do" → do, "dc" → dc, "co" → co, "cc" → cc, "PG" → pg]

getDO[gdo_GDO] := Lookup[Association @@ gdo, "do", {}]
getDC[gdo_GDO] := Lookup[Association @@ gdo, "dc", {}]
getCO[gdo_GDO] := Lookup[Association @@ gdo, "co", {}]
getCC[gdo_GDO] := Lookup[Association @@ gdo, "cc", {}]

getPG[gdo_GDO] := Lookup[Association @@ gdo, "PG", PG[]]

getL[gdo_GDO] := gdo // getPG // getL
getQ[gdo_GDO] := gdo // getPG // getQ
getP[gdo_GDO] := gdo // getPG // getP

setPG[pg_PG][gdo_GDO] := setValue[pg, gdo, "PG"]

setL[L_][gdo_GDO] := setValue[setL[L][gdo // getPG], gdo, "PG"]
setQ[Q_][gdo_GDO] := setValue[setQ[Q][gdo // getPG], gdo, "PG"]
setP[P_][gdo_GDO] := setValue[setP[P][gdo // getPG], gdo, "PG"]

setDO[do_][gdo_GDO] := setValue[do, gdo, "do"]
setDC[dc_][gdo_GDO] := setValue[dc, gdo, "dc"]
setCO[co_][gdo_GDO] := setValue[co, gdo, "co"]
setCC[cc_][gdo_GDO] := setValue[cc, gdo, "cc"]

applyToDO[f_][gdo_GDO] := gdo // setDO[gdo // getDO // f]
applyToDC[f_][gdo_GDO] := gdo // setDC[gdo // getDC // f]
applyToCO[f_][gdo_GDO] := gdo // setCO[gdo // getCO // f]
applyToCC[f_][gdo_GDO] := gdo // setCC[gdo // getCC // f]

applyToPG[f_][gdo_GDO] := gdo // setPG[gdo // getPG // f]

applyToL[f_][gdo_GDO] := gdo // setL[gdo // getL // f]
applyToQ[f_][gdo_GDO] := gdo // setQ[gdo // getQ // f]
applyToP[f_][gdo_GDO] := gdo // setP[gdo // getP // f]

CF[e_GDO] :=
 e // applyToDO[Union] // applyToDC[Union] // applyToCO[Union] // applyToCC[Union] //
  applyToPG[CF]
```

Pairing of GDO’s

In[ ]:=
```
Pair[is_List][gdo1_GDO, gdo2_GDO] :=
 GDO["do" → Union[gdo1 // getDO, Complement[gdo2 // getDO, is]],
  "dc" → Union[gdo1 // getDC, gdo2 // getDC],
  "co" → Union[gdo2 // getCO, Complement[gdo1 // getCO, is]],
  "cc" → Union[gdo1 // getCC, gdo2 // getCC],
  "PG" → Pair[is][gdo1 // getPG, gdo2 // getPG]]

gdo1_GDO // gdo2_GDO := Pair[Intersection[gdo1 // getCO, gdo2 // getDO]][gdo1, gdo2];

GDO /: Congruent[gdo1_GDO, gdo2_GDO] := And[Sort@*getDO /@ Equal[gdo1, gdo2],
  Sort@*getDC /@ Equal[gdo1, gdo2], Sort@*getCO /@ Equal[gdo1, gdo2],
  Sort@*getCC /@ Equal[gdo1, gdo2], Congruent[gdo1 // getPG, gdo2 // getPG]]

GDO /: gdo1_GDO gdo2_GDO := GDO["do" → Union[gdo1 // getDO, gdo2 // getDO],
  "dc" → Union[gdo1 // getDC, gdo2 // getDC], "co" → Union[gdo1 // getCO, gdo2 // getCO],
  "cc" → Union[gdo1 // getCC, gdo2 // getCC], "PG" → (gdo1 // getPG) ** (gdo2 // getPG)]

setEpsilonDegree[k_Integer][gdo_GDO] := setP[Series[Normal@getP@gdo, {ϵ, 0, k}]][gdo]
```

Conversion maps from old notation

In[ ]:=
```
fromE[Subscript[𝔼, {do_List, dc_List} → {co_List, cc_List}][L_, Q_, P_]] :=
 toGDO[do, dc, co, cc, fromE[𝔼[L, Q, P]]]

fromE[Subscript[𝔼, dom_List → cod_List][L_, Q_, P_]] :=
 GDO["do" → dom, "co" → cod, "PG" → fromE[𝔼[L, Q, P]]]
```

Algebra building blocks

In[∘]:=
```
fromLog[ℓ_] := CF@Module[{L, l0 = Limit[ℓ, ϵ → 0]}, L = l0 /. (η | y | ξ | x)[_] → 0;
   PG["L" → L, "Q" → l0 - L] /. l2U]

cΛ = (η[i] + E^(-γ α[i] - ϵ β[i]) η[j] / (1 + γ ϵ η[j] ξ[i])) y[k] +
   (β[i] + β[j] + Log[1 + γ ϵ η[j] ξ[i]] / ϵ) b[k] + (α[i] + α[j] + Log[1 + γ ϵ η[j] ξ[i]] / γ)
    a[k] + (ξ[j] + E^(-γ α[j] - ϵ β[j]) ξ[i] / (1 + γ ϵ η[j] ξ[i])) x[k];

cm[i_, j_, k_] = GDO["do" → {i, j}, "co" → {k}, "PG" → fromLog[cΛ]];

cη[i_] = GDO["co" → {i}];
cσ[i_, j_] = GDO["do" → {i}, "co" → {j},
   "PG" → fromLog[β[i] b[j] + α[i] a[j] + η[i] y[j] + ξ[i] x[j]]];
cϵ[i_] = GDO["do" → {i}];
cΔ[i_, j_, k_] = GDO["do" → {i}, "co" → {j, k}, "PG" → fromLog[
     β[i] (b[j] + b[k]) + α[i] (a[j] + a[k]) + η[i] (y[j] + y[k]) + ξ[i] (x[j] + x[k])]];

sY[i_, j_, k_, l_, m_] = GDO["do" → {i}, "co" → {j, k, l, m},
   "PG" → fromLog[β[i] b[k] + α[i] a[l] + η[i] y[j] + ξ[i] x[m]]];

sS[i_] = GDO["do" → {i}, "co" → {i},
   "PG" → fromLog[-(β[i] b[i] + α[i] a[i] + η[i] y[i] + ξ[i] x[i])]];

cS[i_] = sS[i] // sY[i, 1, 2, 3, 4] // cm[4, 3, i] // cm[i, 2, i] // cm[i, 1, i];

cR[i_, j_] = GDO["co" → {i, j}, "PG" → toPG[ℏ a[j] b[i], (B[i] - 1) / (-b[i]) x[j] y[i], 1]]

cRi[i_, j_] =
 GDO["co" → {i, j}, "PG" → toPG[-ℏ a[j] b[i], (B[i] - 1) / (B[i] b[i]) x[j] y[i], 1]]

CC[i_] := GDO["co" → {i}, "PG" → PG["P" → B[i]^(1 / 2)]]
CCi[i_] := GDO["co" → {i}, "PG" → PG["P" → B[i]^(-1 / 2)]]
```

Out[∘]=

$$\mathbb{E}^{\{\},\{\}}_{\{i,j\},\{\}}\left[a_j\, b_i,\ -\frac{(-1+B_i)\, x_j\, y_i}{b_i},\ 1\right]$$

Out[∘]=

$$\mathbb{E}^{\{\},\{\}}_{\{i,j\},\{\}}\left[-a_j\, b_i,\ \frac{(-1+B_i)\, x_j\, y_i}{b_i\, B_i},\ 1\right]$$

Defining the trace

Coefficient extractors

In[∘]:=
```
getConstLCoef::usage =
 "getConstLCoef[i][gdo] returns the terms in the L-portion of a GDO
```

```
   expression which are not a function of y[i], b[i], a[i], nor x[i]."
getConstLCoef[i_][gdo_] := (SeriesCoefficient[#, {b[i], 0, 0}] &) @*
   (Coefficient[#, y[i], 0] &) @* (Coefficient[#, a[i], 0] &) @*
   (Coefficient[#, x[i], 0] &) @* ReplaceAll[U2l] @* getL@gdo

getConstQCoef::usage =
 "getConstQCoef[i][gdo] returns the terms in the Q-portion of a GDO
   expression which are not a function of y[i], b[i], a[i], nor x[i]."
getConstQCoef[i_][gdo_][bb_] :=
 ReplaceAll[{b[i] → bb}] @* (Coefficient[#, y[i], 0] &) @* (Coefficient[#, a[i], 0] &) @*
   (Coefficient[#, x[i], 0] &) @* ReplaceAll[U2l] @* getQ@gdo

getyCoef::usage = "getyCoef[i][gdo][b[i]]
   returns the linear coefficient of y[i] as a function of b[i]."
getyCoef[i_][gdo_][bb_] := ReplaceAll[{b[i] → bb}] @* ReplaceAll[U2l] @*
   (Coefficient[#, x[i], 0] &) @* (Coefficient[#, y[i], 1] &) @* getQ@gdo

getbCoef::usage = "getbCoef[i][gdo] returns the linear coefficient of b[i]."
getbCoef[i_][gdo_] := (SeriesCoefficient[#, {b[i], 0, 1}] &) @*
   (Coefficient[#, a[i], 0] &) @* (Coefficient[#, x[i], 0] &) @*
   (Coefficient[#, y[i], 0] &) @* ReplaceAll[U2l] @* getL@gdo

getPCoef::usage =
 "getPCoef[i][gdo] returns the perturbation P of a GDO as a function of b[i]."
getPCoef[i_][gdo_][bb_] :=
 ReplaceAll[{b[i] → bb}] @* (Coefficient[#, a[i], 0] &) @* (Coefficient[#, x[i], 0] &) @*
   (Coefficient[#, y[i], 0] &) @* ReplaceAll[U2l] @* getP@gdo

getaCoef::usage = "getaCoef[i][gdo] returns the linear coefficient of a[i]."
getaCoef[i_][gdo_] := (SeriesCoefficient[#, {b[i], 0, 0}] &) @*
   (Coefficient[#, a[i], 1] &) @* ReplaceAll[U2l] @* getL@gdo

getxCoef::usage = "getxCoef[i][gdo][b[i]]
   returns the linear coefficient of x[i] as a function of b[i]."
getxCoef[i_][gdo_][bb_] := ReplaceAll[{b[i] → bb}] @* ReplaceAll[U2l] @*
   (Coefficient[#, y[i], 0] &) @* (Coefficient[#, x[i], 1] &) @* getQ@gdo

getabCoef::usage = "getabCoef[i][gdo] returns the linear coefficient of a[i]b[i]."
getabCoef[i_][gdo_] := (SeriesCoefficient[#, {b[i], 0, 1}] &) @*
   (Coefficient[#, a[i], 1] &) @* ReplaceAll[U2l] @* getL@gdo

getxyCoef::usage = "getxyCoef[i][gdo][b[i]] returns
   the linear coefficient of x[i]y[i] as a function of b[i]."
getxyCoef[i_][gdo_][bb_] := ReplaceAll[{b[i] → bb}] @* ReplaceAll[U2l] @*
   (Coefficient[#, x[i], 1] &) @* (Coefficient[#, y[i], 1] &) @* getQ@gdo
```

Out[ ]=

getConstLCoef[i][gdo] returns the terms in the L-portion of a GDO expression which are not a function of y[i], b[i], a[i], nor x[i].

Out[ ]=

getConstQCoef[i][gdo] returns the terms in the Q-portion of a GDO expression which are not a function of y[i], b[i], a[i], nor x[i].

Out[ ]=

getyCoef[i][gdo][b[i]] returns the linear coefficient of y[i] as a function of b[i].

Out[ ]=

getbCoef[i][gdo] returns the linear coefficient of b[i].

Out[ ]=

getPCoef[i][gdo] returns the perturbation P of a GDO as a function of b[i].

Out[ ]=

getaCoef[i][gdo] returns the linear coefficient of a[i].

Out[ ]=

getxCoef[i][gdo][b[i]] returns the linear coefficient of x[i] as a function of b[i].

Out[ ]=

getabCoef[i][gdo] returns the linear coefficient of a[i]b[i].

Out[ ]=

getxyCoef[i][gdo][b[i]] returns the linear coefficient of x[i]y[i] as a function of b[i].

The trace

In[ ]:=
```
safeEval[f_][x_] := Module[{fx, x0},
  If[(fx = Quiet[f[x]]) === Indeterminate, Series[f[x0], {x0, x, 0}] // Normal, fx]]

closeComponent[i_][gdo_GDO] :=
 gdo // setCO[Complement[gdo // getCO, {i}]] // setCC[Union[gdo // getCC, {i}]]

tr::usage =
 "tr[i] computes the trace of a GDO element on component i. Current implementation
   assumes the Subscript[a, i] Subscript[b, i] term vanishes and $k=0."
tr::nonzeroSigma = "tr[`1`]: Component `1` has writhe: `2`, expected: 0."
tr[i_][gdo_GDO] :=
 Module[{cL = getConstLCoef[i][gdo], cQ = getConstQCoef[i][gdo], βP = getPCoef[i][gdo],
     ηη = getyCoef[i][gdo], ββ = getbCoef[i][gdo], αα = getaCoef[i][gdo],
     ξξ = getxCoef[i][gdo], λ = getxyCoef[i][gdo], ta}, ta = (1 - Exp[-αα]) t[i];
    expL = cL + αα a[i] + ββ ta;
    expQ = safeEval[cQ[#] + t[i] ηη[#] ξξ[#] / (1 - t[i] λ[#]) &][ta];
    expP = safeEval[βP[#] / (1 - t[i] λ[#]) &][ta];
    CF[(gdo // closeComponent[i] // setL[expL] // setQ[expQ] // setP[expP]) //. l2U]] /;
   Module[{σ = getabCoef[i][gdo]},
    If[σ == 0, True, Message[tr::nonzeroSigma, i, ToString[σ]];
     False]]
```

Out[ ]=
```
tr[i] computes the trace of a GDO element on component i. Current implementation
  assumes the Subscript[a, i] Subscript[b, i] term vanishes and $k=0.
```

Out[ ]=
```
tr[`1`]: Component `1` has writhe: `2`, expected: 0.
```

Z invariant

In[ ]:=
```
CCn[i_][n_Integer] := Module[{j}, If[n == 0, GDO["co" → {i}],
    If[n > 0, If[n == 1, CC[i], CC[j] // CCn[i][n - 1] // cm[i, j, i]],
     If[n == -1, CCi[i], CCi[j] // CCn[i][n + 1] // cm[i, j, i]]]]]]

cm[{}, j_] := cη[j]
cm[{i_}, j_] := cσ[i, j]
cm[{i_, j_}, k_] := cm[i, j, k]
cm[ii_List, k_] := Module[{i = First[ii], is = Rest[ii], j, js, l}, j = First[is];
  js = Rest[is];
  cm[i, j, l] // cm[Prepend[js, l], k]]

toGDO[Xp[i_, j_]] := cR[i, j]
toGDO[Xm[i_, j_]] := cRi[i, j]
```

```
toGDO[{i_, n_}] := CCn[i][n]
toGDO[xs_Strand] := cm[List @@ xs, First[xs]]
toGDO[xs_Loop] := Module[{x = First[xs]}, cm[List @@ xs, x] // tr[x]]

toList[RVT[cs_List, xs_List, rs_List]] :=
 Flatten[#, 1] &@ ((toGDO /@ # &) /@ {xs, rs, cs})

getIndices[RVT[cs_List, _List, _List]] := Sort@Flatten[#, 1] &@ (List @@@ cs)

ZFramed[rvt_RVT] := Fold[#2[#1] &, GDO["co" → getIndices@rvt], toList@rvt]

combineBySecond[l_List] := mergeWith[Total, #] & /@ GatherBy[l, First];
combineBySecond[lis___] := combineBySecond[Join[lis]]

mergeWith[f_, l_] := {l⟦1, 1⟧, f@(#⟦2⟧ & /@ l)}

Reindex[RVT[cs_, xs_, rs_]] :=
 Module[{sf, cs2, xs2, rs2, repl, repl2}, sf = Flatten[List @@ # & /@ cs];
  repl = (Thread[sf → Range[Length[sf]]]);
  repl2 = repl /. {(a_ → b_) → ({a, i_} → {b, i})};
  cs2 = cs /. repl;
  xs2 = xs /. repl;
  rs2 = rs /. repl2;
  RVT[cs2, xs2, rs2]]

Unwrithe[RVT[cs_List, xs_List, rs_List]] := Module[{lw},
  lw = Table[{l, Plus @@ xs /. {Xp[i_, j_] :→ If[MemberQ[l, i] ∧ MemberQ[l, j], 1, 0],
       Xm[i_, j_] :→ If[MemberQ[l, i] ∧ MemberQ[l, j], -1, 0]}}, {l, cs}];
  addLoops[l_, n_] := Join[l, Head[l] @@ Table[Subscript[Last[l], i], {i, 2 Abs[n]}]];
  Xn[n_] := If[n ≥ 0, Xm, Xp];
  (*Loops to counteract the writhe.*) addXings[l_, n_] := If[n == 0, {},
    Table[Xn[n][Subscript[Last[l], 2 i - 1], Subscript[Last[l], 2 i]], {i, Abs[n]}]];
  addRots[l_, n_] := {First@l, n};
  (*Print["lw: ",lw];*) Reindex@RVT[addLoops @@@ lw,
    Join[xs, Flatten[addXings @@@ lw]], combineBySecond[rs, addRots @@@ lw]]]

Z[L_RVT] := ZFramed[Unwrithe[L]]
```

Partial Trace

In[ ]:=
```
ptr[L_] := Module[{ZL = Z[L], cod}, cod = getCO@ZL;
  Table[(Composition @@ Table[tr[j], {j, Complement[cod, {i}]}])[ZL], {i, cod}]]
```

Reindexing of GDO's

In[ ]:=
```
getGDOIndices[gdo_GDO] := Sort@Catenate@Through[{getDO, getDC, getCO, getCC}@gdo]

isolateVarIndices[i_ → j_] := (v : y | b | t | a | x | η | β | α | ξ | A | B | T)[i] → v[j];

ReindexBy[f_][gdo_GDO] :=
 Module[{replacementRules, varIndexFunc, repFunc, indices = getGDOIndices[gdo]},
  replacementRules = Thread[indices → (f /@ indices)];
  repFunc = ReplaceAll[replacementRules];
  varIndexFunc = ReplaceAll[Thread[isolateVarIndices[replacementRules]]];
  gdo // applyToPG[varIndexFunc] // applyToCO[repFunc] // applyToDO[repFunc] //
    applyToDC[repFunc] // applyToCC[repFunc]]

fromAssoc[ass_] := Association[ass][#] &

ReindexToInteger[gdos_List] :=
 Module[{is = getGDOIndices@gdos⟦1⟧, f}, f = fromAssoc@Thread[is → Range[Length[is]]];
  ReindexBy[f] /@ gdos]

getReindications[gdos_List] :=
 Module[{gdosInt = ReindexToInteger[gdos], is, fs, ls}, is = getGDOIndices[gdosInt⟦1⟧];
  fs = (fromAssoc@*Association@*Thread) /@ (is → # & /@ Permutations[is]);
  ls = CF@ReindexBy[#] /@ gdosInt & /@ fs;
  Sort[Sort /@ ls]]

getCanonicalIndex[gdo_] := First@getReindications@gdo
```

Where the MVA is not stronger than ptr

In[ ]:=
```
getCanonicalIndex@*ptr@*toRVT /@ {Link[5, Alternating, 1], Link[7, NonAlternating, 2]}
```

Out[ ]=

$$\left\{\left\{\mathbb{E}^{\{\},\{\}}_{\{1\},\{2\}}\left[0,0,\frac{B_1}{B_1+t_2-2B_1t_2+B_1^2t_2}\right],\mathbb{E}^{\{\},\{\}}_{\{2\},\{1\}}\left[0,0,\frac{B_2^{3/2}}{B_2+t_1-2B_2t_1+B_2^2t_1}\right]\right\},\left\{\mathbb{E}^{\{\},\{\}}_{\{1\},\{2\}}\left[0,0,\frac{B_1}{B_1+t_2-2B_1t_2+B_1^2t_2}\right],\mathbb{E}^{\{\},\{\}}_{\{2\},\{1\}}\left[0,0,\frac{B_2^{5/2}}{1-B_2+B_2^2+t_1-2B_2t_1+B_2^2t_1}\right]\right\}\right\}$$

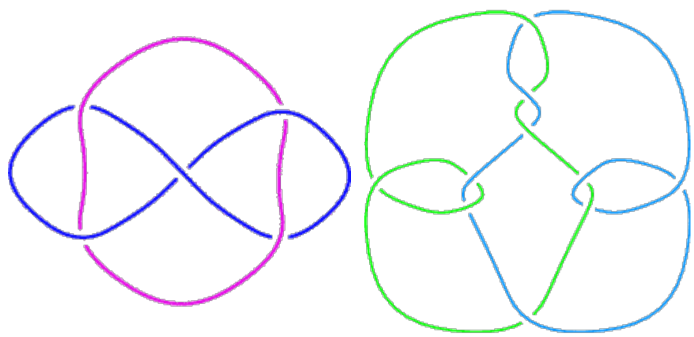
In[ ]:=
```
MultivariableAlexander[#][B] & /@ {Link[5, Alternating, 1], Link[7, NonAlternating, 2]}
```

KnotTheory: Loading precomputed data in MultivariableAlexander4Links`.

Out[ ]=

$$\left\{\frac{(-1+B_1)(-1+B_2)}{\sqrt{B_1}\sqrt{B_2}},\frac{(-1+B_1)(-1+B_2)}{\sqrt{B_1}\sqrt{B_2}}\right\}$$

Where ptr is not stronger than the MVA



L5a1 L10a95

In[∘]:=
```
getCanonicalIndex@*ptr@*toRVT /@ {Link[5, Alternating, 1], Link[10, Alternating, 95]}
```

Out[∘]=

$$\left\{\left\{\mathbb{E}^{\{\},\{\}}_{\{1\},\{2\}}\left[0, 0, \frac{B_1}{B_1 + t_2 - 2 B_1 t_2 + B_1^2 t_2}\right], \mathbb{E}^{\{\},\{\}}_{\{2\},\{1\}}\left[0, 0, \frac{B_2^{3/2}}{B_2 + t_1 - 2 B_2 t_1 + B_2^2 t_1}\right]\right\}, \left\{\mathbb{E}^{\{\},\{\}}_{\{1\},\{2\}}\left[0, 0, \frac{B_1}{B_1 + t_2 - 2 B_1 t_2 + B_1^2 t_2}\right], \mathbb{E}^{\{\},\{\}}_{\{2\},\{1\}}\left[0, 0, \frac{B_2^{3/2}}{B_2 + t_1 - 2 B_2 t_1 + B_2^2 t_1}\right]\right\}\right\}$$

In[∘]:=
```
MultivariableAlexander[#][B] & /@ {Link[5, Alternating, 1], Link[10, Alternating, 95]}
```

Out[∘]=

$$\left\{\frac{(-1 + B_1)(-1 + B_2)}{\sqrt{B_1}\sqrt{B_2}}, -\frac{(-1 + B_1)(-1 + B_2)(-1 + B_1 + B_2)(-B_1 - B_2 + B_1 B_2)}{B_1^{3/2} B_2^{3/2}}\right\}$$

Where the MVA+A+A is not stronger than ptr?

In[∘]:=
```
toRVT[Link[2, Alternating, 1]]
```

Out[∘]=
```
RVT[{Strand[1, 2], Strand[3, 4]}, {Xm[1, 4], Xm[3, 2]}, {{1, 0}, {2, 0}, {3, 0}, {4, 1}}]
```

In[∘]:=
```
ptr[ℰ_] /; Head[ℰ] =!= RVT := ptr[toRVT[ℰ]]
```

In[ ]:= `ptr /@ AllLinks[{2, 5}]`

Out[ ]=

$$\left\{\left\{\mathbb{E}^{\{\},\{\}}_{\{1\},\{3\}}\left[-a_3 b_1+\frac{a_1 (1-B_1) t_3}{B_1}, \frac{T_3 x_1 y_1 - T_3^{\frac{1}{B_1}} x_1 y_1}{b_1 T_3}, T_3^{\frac{1}{2}-\frac{1}{2 B_1}}\right],\right.\right.$$

$$\left.\mathbb{E}^{\{\},\{\}}_{\{3\},\{1\}}\left[-a_1 b_3+\frac{a_3 (1-B_3) t_1}{B_3}, \frac{T_1 x_3 y_3 - T_1^{\frac{1}{B_3}} x_3 y_3}{b_3 T_1}, \sqrt{B_3}\right]\right\},$$

$$\left\{\mathbb{E}^{\{\},\{\}}_{\{1\},\{5\}}\left[-2 a_5 b_1+\frac{a_1 \left(2-2 B_1^2\right) t_5}{B_1^2}, \frac{T_5^2 x_1 y_1 - T_5^{\frac{2}{B_1^2}} x_1 y_1}{b_1 T_5^2}, \frac{T_5^{3/2}+B_1 T_5^{3/2}}{T_5^{1+\frac{1}{2 B_1^2}}+B_1 T_5^{\frac{3}{2 B_1^2}}}\right],\right.$$

$$\left.\mathbb{E}^{\{\},\{\}}_{\{5\},\{1\}}\left[-2 a_1 b_5+\frac{a_5 \left(2-2 B_5^2\right) t_1}{B_5^2}, \frac{T_1^2 x_5 y_5 - T_1^{\frac{2}{B_5^2}} x_5 y_5}{b_5 T_1^2}, \frac{\sqrt{B_5}\, T_1+B_5^{3/2} T_1}{T_1+B_5 T_1^{\frac{1}{B_5^2}}}\right]\right\},$$

$$\left.\left\{\mathbb{E}^{\{\},\{\}}_{\{1\},\{5\}}\left[0, 0, \frac{B_1}{B_1+t_5-2 B_1 t_5+B_1^2 t_5}\right], \mathbb{E}^{\{\},\{\}}_{\{5\},\{1\}}\left[0, 0, \frac{B_5^{3/2}}{B_5+t_1-2 B_5 t_1+B_5^2 t_1}\right]\right\}\right\}$$

Talk title: “Link invariants from the 2D Lie algebra: Some mysteries”

Abstract. A standard construction associates a tangle invariant Z with the 2D Lie algebra L; it is fast to compute (poly time!), it is well-behaved under standard operations (strand stitching, strand doubling, etc.), and when restricted to knots, it yields the Alexander polynomial. In this talk we explain how a study of the co-invariants of the double of L leads to an invariant Y of links which we understand a lot less well: Y is still well behaved under knot operations, and when we can compute Y, the computation is easy. But we don’t know how to compute Y algebraically for links with more than two components, and we don’t know where Y fits in the bigger Alexander world: we give counterexamples to show that it is not a simple variant of the Alexander or multi-variable Alexander invariants.