Implementation.

ωεβ/SL2Portfolio

```
QZip[ξS_List,simp_][E[L_, Q_, P_]] :=
  Module[{ξ, z, zs, c, ys, ηs, qt, zrule, Q1, Q2},
   zs = Table[ξ*, {ξ, ξS}];
   c = Q /. Alternatives@@(ξS ⋃ zs) → 0;
   ys = Table[∂ξ (Q /. Alternatives@@zs → 0), {ξ, ξS}];
   ηs = Table[∂z (Q /. Alternatives@@ξS → 0), {z, zs}];
   qt = Inverse@Table[Kδz,ξ* - ∂z,ξQ, {ξ, ξS}, {z, zs}];
   zrule = Thread[zs → qt.(zs + ys)];
   Q2 = (Q1 = c + ηs.zs /. zrule) /. Alternatives@@zs → 0;
   simp /@ E[L, Q2, Det[qt]] e^-Q2 Zip_ξS[e^Q1 (P /. zrule)]];
QZip_ξS_List := QZip_ξS,CF;
```

```
LZip[ξS_List,simp_][E[L_, Q_, P_]] :=
  Module[{ξ, z, zs, c, ys, ηs, lt, zrule, L1, L2, Q1, Q2},
   zs = Table[ξ*, {ξ, ξS}];
   c = L /. Alternatives@@(ξS ⋃ zs) → 0;
   ys = Table[∂ξ (L /. Alternatives@@zs → 0), {ξ, ξS}];
   ηs = Table[∂z (L /. Alternatives@@ξS → 0), {z, zs}];
   lt = Inverse@Table[Kδz,ξ* - ∂z,ξL, {ξ, ξS}, {z, zs}];
   zrule = Thread[zs → lt.(zs + ys)];
   L2 = (L1 = c + ηs.zs /. zrule) /. Alternatives@@zs → 0;
   Q2 = (Q1 = Q /. T2t /. zrule) /. Alternatives@@zs → 0;
   simp /@
     E[L2, Q2, Det[lt] e^-L2-Q2
       Zip_ξS[e^L1+Q1 (P /. T2t /. zrule)]] //. t2T];
LZip_ξS_List := LZip_ξS,CF;
```

```
Bind{}[L_, R_] := L R;
Bind{is__}[L_E, R_E] := Module[{n},
   Times[
      L /. Table[(v : T | t | a | x | y)i → vn@i, {i, {is}}],
      R /. Table[(v : τ | α | ξ | η)i → vn@i, {i, {is}}]
     ] // LZip_Flatten@Table[{τn@i,an@i},{i,{is}}] //
    QZip_Flatten@Table[{ξn@i,yn@i},{i,{is}}]];
B_i_List := Bind_i; B_is___ := Bind_{is};
Bind[E_E] := E;
Bind[Ls__, ξS_List, R_] := Bind_ξS[Bind[Ls], R];
```

A Partial To Do List.

- Complete all "docility" arguments by identifying a "contained" docile substructure.
- Understand denominators and get rid of them.
- See if much can be gained by including P in the exponential: $\mathbb{e}^{L+Q}P \rightsquigarrow \mathbb{e}^{L+Q+P}$?
- Clean the program and make it efficient.
- Run it for all small knots and links, at $k = 2, 3$.
- Understand the centre and figure out how to read the output.
- Execute the Drinfel'd double procedure at $\mathbb{E}$-level (and thus get rid of `DeclareAlgebra` and all that is around it!).
- Extend to sl_3 and beyond.
- Do everything with `Zip` and `Bind` as the fundamentals, without ever referring back to (quantized) Lie algebras.
- Prove a genus bound and a Seifert formula.
- Obtain "Gauss-Gassner formulas" (ωεβ/NCSU).
- Relate with Melvin-Morton-Rozansky and with Rozansky-Overbay.
- Understand the braid group representations that arise.
- Find a topological interpretation. The Garoufalidis-Rozansky "loop expansion" [GR]?
- Figure out the action of the Cartan automorphism.
- Disprove the ribbon-slice conjecture!
- Figure out the action of the Weyl group.
- Do everything at the "arrow diagram" level of finite-type invariants of (rotational) virtual tangles.
- What else can you do with the "solvable approximations"?
- And with the "Gaussian zip and bind" technology?

References.

[BN1] D. Bar-Natan, *Balloons and Hoops and their Universal Finite Type Invariant, BF Theory, and an Ultimate Alexander Invariant,* ωεβ/KBH, arXiv:1308.1721.

[BN2] D. Bar-Natan, *Polynomial Time Knot Polynomial,* research proposal for the 2017 Killam Fellowship, ωεβ/K17.

[BNG] D. Bar-Natan and S. Garoufalidis, *On the Melvin-Morton-Rozansky conjecture,* Invent. Math. **125** (1996) 103–133.

[BNS] D. Bar-Natan and S. Selmani, *Meta-Monoids, Meta-Bicrossed Products, and the Alexander Polynomial,* J. of Knot Theory and its Ramifications **22-10** (2013), arXiv:1302.5689.

[BV] D. Bar-Natan and R. van der Veen, *A Polynomial Time Knot Polynomial,* Proc. Amer. Math. Soc., to appear, arXiv:1708.04853.

[Fa] L. Faddeev, *Modular Double of a Quantum Group,* arXiv:math/9912078.

[GR] S. Garoufalidis and L. Rozansky, *The Loop Exapnsion of the Kontsevich Integral, the Null-Move, and S -Equivalence,* arXiv:math.GT/0003187.

[GST] R. E. Gompf, M. Scharlemann, and A. Thompson, *Fibered Knots and Potential Counterexamples to the Property 2R and Slice-Ribbon Conjectures,* Geom. and Top. **14** (2010) 2305–2347, arXiv:1103.1601.

[MM] P. M. Melvin and H. R. Morton, *The coloured Jones function,* Commun. Math. Phys. **169** (1995) 501–520.

[Ov] A. Overbay, *Perturbative Expansion of the Colored Jones Polynomial,* University of North Carolina PhD thesis, ωεβ/Ov.

[Qu] C. Quesne, *Jackson's q-Exponential as the Exponential of a Series,* arXiv:math-ph/0305003.

[Ro1] L. Rozansky, *A contribution of the trivial flat connection to the Jones polynomial and Witten's invariant of 3d manifolds, I,* Comm. Math. Phys. **175-2** (1996) 275–296, arXiv:hep-th/9401061.

[Ro2] L. Rozansky, *The Universal R-Matrix, Burau Representation and the Melvin-Morton Expansion of the Colored Jones Polynomial,* Adv. Math. **134-1** (1998) 1–31, arXiv:q-alg/9604005.

[Ro3] L. Rozansky, *A Universal U(1)-RCC Invariant of Links and Rationality Conjecture,* arXiv:math/0201139.

[Vo] H. Vo, *Alexander Invariants of Tangles via Expansions,* University of Toronto Ph.D. thesis, in preparation.

[Za] D. Zagier, *The Dilogarithm Function,* in Cartier, Moussa, Julia, and Vanhove (eds) *Frontiers in Number Theory, Physics, and Geometry II.* Springer, Berlin, Heidelberg, and ωεβ/Za.

The Complete Implementation.

ωεβ/SL2Portfolio

An even fuller implementation is at ωεβ/FullImp.

Initialization / Utilities

```
$p = 2; $k = 1; $U = QU; $E := {$k, $p};
$trim := {ħ^p_- · /; p > $p → 0, ϵ^k_- · /; k > $k → 0};
q_ħ = e^γ ϵ ħ;
T2t = {T_i^p_- · → e^p ħ t_i, T^p_- · → e^p ħ t};
t2T = {e^c_- · t_i_+b_- · :> T_i^c/ħ e^b, e^c_- · t+b_- · :> T^c/ħ e^b, e^ℰ_- :> e^Expand@ℰ};
SetAttributes[SS, HoldAll];
SS[ℰ_, op_] := Collect[
   Normal@Series[If[$p > 0, ℰ, ℰ /. T2t], {ħ, 0, $p}],
   ħ, op];
SS[ℰ_] := SS[ℰ, Together];
Simp[ℰ_, op_] := Collect[ℰ, _CU | _QU, op];
Simp[ℰ_] := Simp[ℰ, SS[#, Expand] &];
Kδ /: Kδ_i_,j_ := If[i === j, 1, 0];
c_Integer_k_Integer := c + O[ϵ]^k+1;
```

```
CF[ℰ_] := ExpandDenominator@
    ExpandNumerator@
     Together[Expand[ℰ] //. e^x_- e^y_- :> e^x+y /. e^x_- :> e^CF[x]];
Unprotect[SeriesData];
SeriesData /: CF[sd_SeriesData] := MapAt[CF, sd, 3];
SeriesData /: Expand[sd_SeriesData] :=
  MapAt[Expand, sd, 3];
SeriesData /: Simplify[sd_SeriesData] :=
  MapAt[Simplify, sd, 3];
SeriesData /: Together[sd_SeriesData] :=
  MapAt[Together, sd, 3];
SeriesData /: Collect[sd_SeriesData, specs__] :=
  MapAt[Collect[#, specs] &, sd, 3];
Protect[SeriesData];
```

DeclareAlgebra

Video and more at http://www.math.toronto.edu/~drorbn/Talks/Matemale-1804/