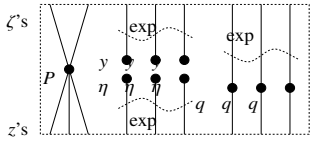
A Partial To Do List.

- Understand tr and links.
- Implement Φ, J. Determine the appropriate wt-0 ground ring.
- Implement the "dequantizators".
- Understand denominators and get rid of them.
- Implement zipping at the log-level.
- Clean the program and make it efficient.
- Run it for all small knots and links, at $k = 3, 4$.
- Understand the centre and figure out how to read the output.
- Is the "+" really necessary in sl^{ϵ}_{2+}? Why?
- Extend to sl_3 and beyond.
- Describe a genus bound and a Seifert formula.
- Obtain "Gauss-Gassner formulas" (ωεβ/NCSU).
- Relate with the representation theory dogma, with Melvin-Morton-Rozansky and with Rozansky-Overbay.
- Understand the braid group representations that arise.
- Relate with finite-type (Vassiliev) invariants.
- Find a topological interpretation/foundation. The Garoufalidis - Rozansky "loop expansion" [GR]?
- Figure out the action of the Cartan automorphism.
- Understand "the subspace of classical knots / tangles".
- Disprove the ribbon-slice conjecture!
- Figure out the action of the Weyl group.
- Use to study "Ševera quantization".
- Do everything at the "arrow diagram" level of finite-type invariants of (rotational) virtual tangles.
- Find "internal" proofs of consistency.
- What else can you do with the "solvable approximations"?
- And with the "Gaussian compositions" technology?

Warning. Some implementation details match earlier versions of the theory.

The Zipping Theorem. If P has a finite ζ-degree and $\tilde{q}$ is the inverse matrix of $1-q$: $(\delta^i_j - q^i_j)\tilde{q}^j_k = \delta^i_k$, then



$$\left\langle P(z_i,\zeta^j)\mathfrak{e}^{c+\eta^i z_i + y_j\zeta^j + q^i_j z_i \zeta^j}\right\rangle$$
$$= |\tilde{q}|\mathfrak{e}^{c+\eta^i\tilde{q}^k_i y_k}\left\langle P\left(\tilde{q}^k_i(z_k+y_k),\zeta^j+\eta^i\tilde{q}^j_i\right)\right\rangle.$$

The "Speedy" Engine

ωεβ/engine

Internal Utilities

Canonical Form:

```
CCF[ℰ_] :=
  PPCCF@ExpandDenominator@
    ExpandNumerator@PPTogether@Together[PPExp[
        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_] := PPCF@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[𝔼sp___[ℰs___]] := CF /@ 𝔼sp[ℰs];
```

The Kronecker δ:

```
Kδ /: Kδi_,j_ := If[i === j, 1, 0];
```

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

```
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}]];
```

Zip and Bind

Variables and their duals:

```
{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;
```

Upper to lower and lower to Upper:

```
U2l = {B^p_i_· :> e^(-p ħ γ bi), B^p_· :> e^(-p ħ γ b), T^p_i_· :> e^(p ħ ti),
    T^p_· :> e^(p ħ t), 𝒜^p_i_· :> e^(p γ αi), 𝒜^p_· :> e^(p γ α)};
12U = {e^(c_· bi_ + d_·) :> Bi^(c/(ħ γ)) e^d, e^(c_· b + d_·) :> B^(c/(ħ γ)) e^d,
    e^(c_· ti_ + d_·) :> Ti^(c/ħ) e^d, e^(c_· t + d_·) :> T^(c/ħ) e^d,
    e^(c_· αi_ + d_·) :> 𝒜i^(c/γ) e^d, e^(c_· α + d_·) :> 𝒜^(c/γ) e^d,
    e^ℰ_ :> e^Expand@ℰ};
```

Derivatives in the presence of exponentiated variables:

```
Db[f_] := ∂b f - ħ γ B ∂B f; Dbi_[f_] := ∂bi f - ħ γ Bi ∂Bi f;
Dt[f_] := ∂t f + ħ T ∂T f; Dti_[f_] := ∂ti f + ħ Ti ∂Ti f;
Dα[f_] := ∂α f + γ 𝒜 ∂𝒜 f; Dαi_[f_] := ∂αi f + γ 𝒜i ∂𝒜i f;
Dv_[f_] := ∂v f; D{v_,0}[f_] := f; D{}[f_] := f;
D{v_,n_Integer}[f_] := Dv[D{v,n-1}[f]];
D{l_List,ls___}[f_] := D{ls}[Dl[f]];
```

Finite Zips:

```
collect[sd_SeriesData, ζ_] :=
  MapAt[collect[#, ζ] &, sd, 3];
collect[ℰ_, ζ_] := PPCollect@Collect[ℰ, ζ];
Zip{}[P_] := P;
Zip{ζs_}[Ps_List] := Zip{ζs} /@ Ps;
Zip{ζ_,ζs___}[P_] := PPZip[
  (collect[P // Zip{ζs}, ζ] /. f_. ζ^d_· :> (D{ζ*,d}[f])) /.
    ζ* → 0 /. ((ζ* /. {b → B, t → T, α → 𝒜}) → 1)]
```

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