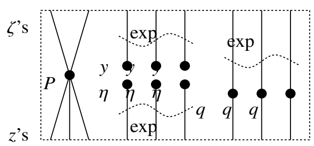
A Partial To Do List.

- 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

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

The "Speedy" Engine

ωεβ/engine

Internal Utilities

Canonical Form:

```
CCF[ℰ_] :=
  PP_CCF@ExpandDenominator@
    ExpandNumerator@PP_Together@Together[PP_Exp[
        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_] := PP_CF@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[𝔼_sp___[ℰs___]] := CF /@ 𝔼_sp[ℰs];
```

The Kronecker δ:

```
Kδ /: Kδ_{i_,j_} := If[i === j, 1, 0];
```

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

```
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, 𝔷};
{τ*, β*, η*, α*, ξ*, 𝔷*} = {t, b, y, a, x, z};
(u_i_)* := (u*)_i;
```

Upper to lower and lower to Upper:

```
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^ℰ_ :→ e^Expand@ℰ};
```

Derivatives in the presence of exponentiated variables:

```
D_b[f_] := ∂_b f - ħ γ B ∂_B f;  D_b_i_[f_] := ∂_b_i f - ħ γ B_i ∂_B_i f;
D_t[f_] := ∂_t f + ħ T ∂_T f;  D_t_i_[f_] := ∂_t_i f + ħ T_i ∂_T_i f;
D_α[f_] := ∂_α f + γ 𝒜 ∂_𝒜 f;  D_α_i_[f_] := ∂_α_i f + γ 𝒜_i ∂_𝒜_i f;
D_v_[f_] := ∂_v f;  D_{v_,0}[f_] := f;  D_{}[f_] := f;
D_{v_,n_Integer}[f_] := D_v[D_{v,n-1}[f]];
D_{l_List,ls___}[f_] := D_{ls}[D_l[f]];
```

Finite Zips:

```
collect[sd_SeriesData, ζ_] :=
  MapAt[collect[#, ζ] &, sd, 3];
collect[ℰ_, ζ_] := PP_Collect@Collect[ℰ, ζ];
Zip_{}[P_] := P;
Zip_ζs_[Ps_List] := Zip_ζs /@ Ps;
Zip_{ζ_,ζs___}[P_] := PP_Zip[
  (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 $\mathbb{E}(L, Q, P) = e^{L+Q}P(\epsilon)$. Such zips regard the L variables as scalars.