Pensieve Header: Verifying the "2D Lie Algebra" formula for the Alexander polynomial. Also, YAAP - Yet Another Alexander Program.

Drawing Arrow Diagrams

```
Draw[expr_] := expr /. gd_GD :> Draw[gd];
Draw[gd_GD] := Module[
   {n = Length[gd], h, k = 0},
   Graphics[{
     Line[{{0, 0}, {2 n + 1, 0}}],
     Table[Text[i, {i, -0.3}], {i, 2 n}],
     (List @@ gd) /.  {
       Ar[i_, j_, s_] :> {
         h = Abs[i - j] / 2;
         BezierCurve[{
            {i, 0}, {i, h}, {(i + j) / 2, h}, {j, h}, {j, 0}
          }, SplineDegree -> 2],
         Text[s * (++k), {(i + j) / 2, h - 0.3}],
         Line[{{j - 0.2, 0.4}, {j, 0}, {j + 0.2, 0.4}}]
        }
      }
    }
   ]
  ];
```

Main

```
GD[K_] := GD @@ PD[K] /. X[i_, j_, k_, l_] :> If[PositiveQ[X[i, j, k, l]],
      Ar[l, i, +1],
      Ar[j, i, -1]
     ];
SL[K_] := Plus @@ (GD[K] /. Ar[t_, h_, s_] :> If[h > t, s DR, s DL]);
Arcs[gd_GD] := Module[
   {n, heads, arcs, k},
   n = Length[gd];
   heads = Sort[Cases[gd, Ar[_, h_, _] :> h]];
   arcs = Table[0, {2 n}];
   arcs[[heads]] = Range[n];
   k = 1;
   Do[If[arcs[[i]] == 0, arcs[[i]] = k, ++k], {i, 2 n}];
   arcs
  ];
R[gd_GD] := Module[
   {n, mat, arcs, k},
   n = Length[gd]; arcs = Arcs[gd];
   mat = Table[0, {n}, {n + 1}];
   k = 0; gd /. Ar[t_, h_, s_] :> {
      ++k;
      mat[[k, arcs[[h]]]] += -1;
      mat[[k, arcs[[t]]]] += 1 - X^s;
      mat[[k, arcs[[h]] + 1]] += X^s
     };
   mat
  ];
e[n_, i_] := ReplacePart[Table[0, {n}], i -> 1];
EZ[gd_GD] := Module[
   {n, r, arcs, j},
   n = Length[gd];
   r = R[gd];
   arcs = Arcs[gd];
   j = 0;
   Simplify[Plus @@ (
      gd /. Ar[t_, h_, s_] :> s * (
           Inverse[Append[r, e[n + 1, arcs[[h]]]]].Append[x e[n, ++j], DR]
          )[[arcs[[t]]]]
     )]
```

```
  ];
EZ1[gd_GD] := Module[
   {n, r, arcs, j},
   n = Length[gd];
   r = R[gd];
   arcs = Arcs[gd];
   j = 0;
   Simplify[List @@ (
      gd /. Ar[t_, h_, s_] :> s * (
           Inverse[Append[r, e[n + 1, arcs[[h]]]]].Append[x e[n, ++j], DR]
          )[[arcs[[t]]]]
     )]
  ];
EZ2[gd_GD] := Module[
   {n, r, arcs, j},
   n = Length[gd];
   r = R[gd];
   arcs = Arcs[gd];
   j = 0;
   Simplify[Plus @@ (
      gd /. Ar[t_, h_, s_] :>
        s * x * Inverse[Append[r, e[n + 1, arcs[[h]]]]][[
           arcs[[t]], ++j
          ]]
     )]
  ];
```