Pensieve header: RVK and Z.

RVK and Z

RVK, rot, Z from 2016-09/OneSmidgen.nb. See also in AP/Projects/SL2Invariant/.

Some details of the code below are at http://drorbn.net/bbs/show?shot=Dror-160920-151350.jpg.

In[ ]:=
```
RVK::usage =
  "RVK[xs, rots] represents a Rotational Virtual Knot with a list of n Xp/Xm crossings
    xs and a length 2n list of rotation numbers rots. Crossing
    sites are indexed 1 through 2n, and rots[[k]] is the rotation
    between site k-1 and site k. RVK is also a casting operator
    converting to the RVK presentation from other knot presentations.";
```

In[ ]:=
```
RVK[pd_PD] := PP_RVK@Module[{n, xs, x, rots, front = {0}, k},
    n = Length@pd; rots = Table[0, {2 n}];
    xs = Cases[pd, x_X :> { Xp[x[[4]], x[[1]]]  PositiveQ@x
                            Xm[x[[2]], x[[1]]]      True   ];
    For[k = 0, k < 2 n, ++k, If[k == 0 ∨ FreeQ[front, -k],
      front = Flatten[front /. k → (xs /. {
              Xp[k + 1, l_] | Xm[l_, k + 1] :> {l, k + 1, 1 - l},
              Xp[l_, k + 1] | Xm[k + 1, l_] :> (++rots[[l]]; {1 - l, k + 1, l})
            })],
      Cases[front, k | -k] /. {k, -k} :> --rots[[k + 1]];
     ]];
    RVK[xs, rots] ];
RVK[K_] := RVK[PD[K]];
```

In[ ]:=
```
rot[i_, 0] := 𝔼_{}→{i}[0, 0, 1];
rot[i_, n_] := Module[{j},
   rot[i, n] = If[n > 0, rot[i, n - 1] kC_j, rot[i, n + 1] k$\overline{C}$_j] // km_{i,j→i}];
```

```
Z[K_] := Z[RVK@K];
Z[rvk_RVK] := (*Z[rvk] =*)
 Monitor[ PP"Z"@Module[{todo, n, rots, ξ, done, st, cx, ξ1, i, j, k, k1, k2, k3},
    {todo, rots} = List@@rvk;
    AppendTo[rots, 0];
    n = Length[todo];
    ξ = 𝔼{}→{0}[0, 0, 1];
    done = {0};
    st = Range[0, 2 n + 1];
    While[{} =!= ($M = todo),
     {cx} = MaximalBy[todo, Length[done ⋂ {#[[1]], #[[2]], #[[1]] - 1, #[[2]] - 1}] &, 1];
     {i, j} = List@@cx;
     ξ1 = Switch[Head[cx],
        Xp, (kRi,j k̄K̄ink̄k) // kmj,k→j,
        Xm, (k̄R̄i,j kKinkk) // kmj,k→j
       ];
     ξ1 = (rot[k, rots[[i]]] ξ1) // kmk,i→i; rots[[i]] = 0;
     ξ1 = (ξ1 rot[k, rots[[i + 1]]]) // kmi,k→i; rots[[i + 1]] = 0;
     ξ1 = (rot[k, rots[[j]]] ξ1) // kmk,j→j; rots[[j]] = 0;
     ξ1 = (ξ1 rot[k, rots[[j + 1]]]) // kmj,k→j; rots[[j + 1]] = 0;
     ξ *= ξ1;
     If[MemberQ[done, i], ξ = ξ // kmi,i+1→i; st = st /. st[[i + 2]] → st[[i + 1]];
     If[MemberQ[done, i - 1], ξ = ξ // kmst[[i]],i→st[[i]]; st = st /. st[[i + 1]] → st[[i]];
     If[MemberQ[done, j], ξ = ξ // kmj,j+1→j; st = st /. st[[j + 2]] → st[[j + 1]];
     If[MemberQ[done, j - 1], ξ = ξ // kmst[[j]],j→st[[j]]; st = st /. st[[j + 1]] → st[[j]];
     done = done ⋃ {i - 1, i, j - 1, j};
     todo = DeleteCases[todo, cx]
    ];
    CF /@ (ξ /. {x0 → x, y0 → y, a0 → a})
   ], $M]
```