```
Once[<< KnotTheory`];
Once[<< "../Projects/Profile/Profile.m"]
```

Rotational Virtual Knots

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

Z

```
ul_ = nl_ = rot[_, 0] = 𝔼[1, 0, 0, 0];
rot[i_, 1] := ur_i;
rot[i_, n_Integer] /; n > 1 := Module[{y}, rot[i, n - 1] rot[y, 1] // m_{i,y→i}];
rot[i_, -1] := nr_i;
rot[i_, n_Integer] /; n < -1 := Module[{y}, rot[i, n + 1] rot[y, -1] // m_{i,y→i}];
```

```
t_ = t;
Z[K_] := Z[RVK@K];
Z[rvk_RVK] := PPz@Module[{todo, n, rots, ℰ, done, st, x, ℰ1, i, j, k, k1, k2, k3},
   {todo, rots} = List@@rvk;
   AppendTo[rots, 0];
   n = Length[todo];
   ℰ = 𝔼[1, 0, 0, 0];
   done = {0};
   st = Range[0, 2 n + 1];
   While[todo =!= {},
    {x} = MaximalBy[todo, Length[done ⋂ {#[[1]], #[[2]], #[[1]] - 1, #[[2]] - 1}] &, 1];
    Z$todo = todo; Z$x = x;
    {i, j} = List@@x;
    ℰ1 = Switch[Head[x],
       Xp, mj,k→j[R+i,j (R-k3,k nrk1 ulk2 // mk,k1→k // mk,k2→k // mk,k3→k)],
       Xm, mj,k→j[R-i,j (R+k,k3 nrk1 ulk2 // mk,k1→k // mk,k2→k // mk,k3→k)]
     ];
    ℰ1 = rot[k, rots[[i]]] ℰ1 // mk,i→i; rots[[i]] = 0;
    ℰ1 = ℰ1 rot[k, rots[[i + 1]]] // mi,k→i; rots[[i + 1]] = 0;
    ℰ1 = rot[k, rots[[j]]] ℰ1 // mk,j→j; rots[[j]] = 0;
    ℰ1 = ℰ1 rot[k, rots[[j + 1]]] // mj,k→j; rots[[j + 1]] = 0;
    ℰ *= ℰ1;
    If[MemberQ[done, i], ℰ = ℰ // mi,i+1→i; st = st /. st[[i + 2]] → st[[i + 1]]];
    If[MemberQ[done, i - 1], ℰ = ℰ // mst[[i]],i→st[[i]]; st = st /. st[[i + 1]] → st[[i]]];
    If[MemberQ[done, j], ℰ = ℰ // mj,j+1→j; st = st /. st[[j + 2]] → st[[j + 1]]];
    If[MemberQ[done, j - 1], ℰ = ℰ // mst[[j]],j→st[[j]]; st = st /. st[[j + 1]] → st[[j]]];
    done = done ⋃ {i - 1, i, j - 1, j};
    todo = DeleteCases[todo, x]
   ];
   ℰ /. {e0 → e, l0 → l, f0 → f}
  ]
```