Pensieve header: Demo for Talks/TrendsInLDT-2005.

```
In[•]:= SetDirectory["C:\\drorbn\\AcademicPensieve\\Talks\\TrendsInLDT-2005"];
        << KnotTheory` (* can be commented out with little loss *)
```

```
Loading KnotTheory` version of February 2, 2020, 10:53:45.2097.
Read more at http://katlas.org/wiki/KnotTheory.
```

```
In[•]:= SetAttributes[VD, Orderless]
```

```
In[•]:= bp[i_, j_] := σ_{i,j}; bm[i_, j_] := σ̄_{i,j};
```

```
In[•]:= Tidy[vd_VD] := Module[{ps = Union @@ (List @@@ vd)},
          Replace[vd, Thread[ps → Range@Length@ps], {2}] ]
```

```
In[•]:= R12Reduce1[vd_VD] := Tidy@Module[{R2s, R2}, Which[
            Length[R2s = Cases[vd, X_s_[i_, j_] :> X_{-s}[i + 1, j + 1]] ⋂ (List @@ vd)] > 0,
            Complement[vd, VD[R2 = First@R2s, R2 /. X_s_[i_, j_] :> X_{-s}[i - 1, j - 1]]],
            Length[R2s = Cases[vd, X_s_[i_, j_] :> X_{-s}[i + 1, j - 1]] ⋂ (List @@ vd)] > 0,
            Complement[vd, VD[R2 = First@R2s, R2 /. X_s_[i_, j_] :> X_{-s}[i - 1, j + 1]]],
            True, DeleteCases[vd, X_[i_, j_] /; Abs[i - j] == 1] ]]
```

```
In[•]:= R12Reduce[vd_VD] := FixedPoint[R12Reduce1, vd]
```

```
In[•]:= γ[vd_VD] := Module[{js, s1, i1, j1, s2, i2, j2},
          js = Cases[vd, X_[_, j_] :> j] ⋂ Cases[vd, X_[i_, _] :> i - 1];
          If[Length[js] == 0, vd,
           j1 = RandomChoice[js]; i2 = j1 + 1;
           Cases[vd, X_s_[i_, j1] :> (s1 = s; i1 = i)];
           Cases[vd, X_s_[i2, j_] :> (s2 = s; j2 = j)];
           Tidy@Join[Complement[vd, VD[X_{s1}[i1, j1], X_{s2}[i2, j2]]],
             VD[X_{s2}[j1, j2], X_{s1}[i1, i2], X_{s1 s2}[i1 - s1/3, j2 + s2/3], X_{-s1 s2}[i1 + s1/3, j2 - s2/3]]
           ] ]]
```

```
In[•]:= Γ[vd_VD] := FixedPoint[γ, vd, 2^16]
```

```
In[•]:= Γ[T_] /; Head[T] =!= VD := Γ[VD[T]]
```

```
In[•]:= Γ̄[vd_VD] := FixedPoint[γ@*R12Reduce, vd, 2^16]
```

```
In[•]:= Γ̄[T_] /; Head[T] =!= VD := Γ̄[VD[T]]
```

```
In[•]:= BR[3, {2, -1, -1, -1, -1}] // Γ̄
Out[•]= VD[BR[1, 2]]
```

```
In[•]:= VPB[n_, {σs___}] := VPB[n, σs];
```

```
In[•]:= VD /: vd1_VD ** vd2_VD := Module[{es1, es2, m2},
  es1 = Cases[vd1, EOS[i_] :> i];
  m2 = Max[es2 = Cases[vd2, EOS[i_] :> i]];
  Tidy[vd1 ⋃ Replace[DeleteCases[vd2, _EOS],
     i_ :> i / m2 - 1 + es1[[1 + Count[es2, e_ /; i > e]]], {2}]]
 ]
```

```
In[•]:= VD[VPB[n_]] := VD @@ (EOS /@ Range[n]);
VD[VPB[n_, σ_{i_,j_}]] := Tidy@Append[VD @@ (EOS /@ Range[n]), X_{+1}[i - 0.5, j - 0.5]];
VD[VPB[n_, σ̄_{i_,j_}]] := Tidy@Append[VD @@ (EOS /@ Range[n]), X_{-1}[i - 0.5, j - 0.5]];
VD[VPB[n_, σ_, σs__]] := VD[VPB[n, σ]] ** VD[VPB[n, σs]]
```

```
In[•]:= VPBGenerators[n_] :=
  VPBGenerators[n] = Flatten@Table[{σ_{i,j}, σ̄_{i,j}}, {i, n}, {j, DeleteCases[Range@n, i]}];
```

```
In[•]:= ProdFollowers[n_, σ_{i_,j_}] := ProdFollowers[n, σ_{i,j}] = Module[{p, q, s},
    Flatten@{σ_{i,j}, σ_{j,i}, σ̄_{j,i},
      Table[{σ_{p,q}, σ_{q,p}, σ̄_{p,q}, σ̄_{q,p}}, {p, {i, j}}, {q, Complement[Range[n], {i, j}]}],
      Table[{σ_{p,q}, σ̄_{p,q}},
       {p, Complement[Range[i + 1, n], {j}]}, {q, Complement[Range[n], {i, j, p}]}]
     }];
ProdFollowers[n_, σ̄_{i_,j_}] := ProdFollowers[n, σ̄_{i,j}] = ProdFollowers[n, σ_{i,j}] /. σ_{i,j} → σ̄_{i,j}
```

```
In[•]:= ProdFollowers[n_, i_Integer] :=
  DeleteCases[Range[Max[Abs[i] - 1, 1], n - 1] ⋃ (-Range[Max[Abs[i] - 1, 1], n - 1]), -i];
```

```
In[•]:= ProdFollowers[5, 3]
Out[•]= {-4, -2, 2, 3, 4}
```

```
In[•]:= ProdVPBs[n_, 0] := {VPB[n]};
ProdVPBs[n_, 1] := VPB[n, #] & /@ VPBGenerators[n];
ProdVPBs[n_, m_] /; m > 1 := Flatten[
  ProdVPBs[n, m - 1] /. VPB[n, σs___, σ_] :> (VPB[n, σs, σ, #] & /@ ProdFollowers[n, σ])]
```

```
In[•]:= CountOUForms[n_, m_] := Module[{k},
  Length@Union@Flatten@Table[Γ̄@vpb, {k, 0, m}, {vpb, ProdVPBs[n, k]}]]
```

```
In[•]:= ProdBs[n_, 0] := {BR[n, {}]};
ProdBs[n_, 1] := BR[n, {#}] & /@ (Range[n - 1] ⋃ (-Range[n - 1]));
ProdBs[n_, m_] /; m > 1 := Flatten[
  ProdBs[n, m - 1] /. BR[n, {σs___, σ_}] :> (BR[n, {σs, σ, #}] & /@ ProdFollowers[n, σ])]
```

In[•]:=
```
ProudBs[3, 3]
```

Out[•]=
```
{BR[3, {-2, -2, -2}], BR[3, {-2, -2, -1}], BR[3, {-2, -2, 1}], BR[3, {-2, -1, -2}],
 BR[3, {-2, -1, -1}], BR[3, {-2, -1, 2}], BR[3, {-2, 1, -2}], BR[3, {-2, 1, 1}],
 BR[3, {-2, 1, 2}], BR[3, {-1, -2, -2}], BR[3, {-1, -2, -1}], BR[3, {-1, -2, 1}],
 BR[3, {-1, -1, -2}], BR[3, {-1, -1, -1}], BR[3, {-1, -1, 2}], BR[3, {-1, 2, -1}],
 BR[3, {-1, 2, 1}], BR[3, {-1, 2, 2}], BR[3, {1, -2, -2}], BR[3, {1, -2, -1}],
 BR[3, {1, -2, 1}], BR[3, {1, 1, -2}], BR[3, {1, 1, 1}], BR[3, {1, 1, 2}],
 BR[3, {1, 2, -1}], BR[3, {1, 2, 1}], BR[3, {1, 2, 2}], BR[3, {2, -1, -2}],
 BR[3, {2, -1, -1}], BR[3, {2, -1, 2}], BR[3, {2, 1, -2}], BR[3, {2, 1, 1}],
 BR[3, {2, 1, 2}], BR[3, {2, 2, -1}], BR[3, {2, 2, 1}], BR[3, {2, 2, 2}]}
```

In[•]:=
```
CountBs[n_, m_] := Module[{k},
  Length@Union@Flatten@Table[Γ̄@b, {k, 0, m}, {b, ProudBs[n, k]}]]
```

In[•]:=
```
CountBs[3, 3]
```

Out[•]=
```
1
```

In[•]:=
```
AllBs[n_, m_] := DeleteDuplicatesBy[Γ̄]@Flatten@Table[b, {k, 0, m}, {b, ProudBs[n, k]}]
```

In[•]:=
```
Length /@ Table[ProudBs[3, k], {k, 0, 3}]
```

Out[•]=
```
{1, 4, 12, 36}
```

In[•]:=
```
AllOUs[n_, 0] := {VD @@ Flatten@Table[{BT[2 i - 1], EOS[2 i]}, {i, n}]};
AllOUs[n_, m_] /; m > 0 :=
 Sort@Flatten[AllOUs[n, m - 1] /. vd_VD :> Module[{BTs, EOSs, k, maxO, s},
       BTs = Sort@Cases[vd, BT[i_] :> i];
       EOSs = Sort@Cases[vd, EOS[i_] :> i];
       maxO = Max[1, Max[Cases[vd, X_[i_, _] :> i]]];
       Table[
        Tidy[Append[vd, Xs[p - 0.5, q + 0.5]]],
        {s, {-1, 1}}, {k, Length[BTs]},
        {q, BTs[[k]], EOSs[[k]] - 1}, {p, Select[BTs, (# ≥ maxO) &]}
       ]]]
```

In[•]:=
```
AllROUs[n_, m_] :=
 Select[AllOUs[n, m] /. vd_VD :> Tidy@DeleteCases[vd, _BT], (# === R12Reduce[#]) &]
```

In[•]:=
```
ξ[vd_VD] := Count[Γ̄[vd], X_[_, _]]
```

In[•]:=
```
VD /: (σ̄i_,j_ | vd_VD) := Switch[Order[ξ[vd], ξ[VD[VPB[Count[vd, _EOS], σ̄i,j]] ** vd]],
   0, Print["OMG, Trouble!"],
   1, False, -1, True];
VD /: (σ̄i_,j_ | vd_VD) := Switch[Order[ξ[vd], ξ[VD[VPB[Count[vd, _EOS], σi,j]] ** vd]],
   0, Print["OMG, Trouble!"],
   1, False, -1, True];
```

In[ ]:=
```
VD /: Divisors[vd_VD] := Select[VPBGenerators[Count[vd, _EOS]], (# | vd) &];
VD /: Quotients[vd_VD] :=
  Γ̄[VD[VPB[Count[vd, _EOS], # /. {σ → σ̄, σ̄ → σ}]] ** vd] & /@ Divisors[vd];
```

In[ ]:=
```
OUGraph[n_, m_] := Module[{gens, OUs, k, d, g, q, m1, m2},
  gens = VPBGenerators[n];
  OUs = Flatten@Table[AllROUs[n, k], {k, 0, m}];
  OURule = Dispatch@Thread[OUs → Range@Length@OUs];
  Graph[
   Range@Length@OUs,
   Union@Flatten@Table[
      m1 = Count[d, X_[_, _]];
      m2 = Count[q = Γ̄[VD[VPB[n, g]] ** d], X_[_, _]];
      If[m2 < m1, Labeled[(d ↦ q) /. OURule, g], Nothing],
      {d, OUs}, {g, gens}
     ]
  ]
 ]
```

In[ ]:=
```
ExtractVPB[vd_VD] := Module[{n, ds, d},
   n = Count[vd, _EOS];
   If[Length[ds = Divisors[vd]] == 0, VPB[n],
    d = First@Sort[ds];
    q = Γ̄[VD[VPB[n, d /. {σ → σ̄, σ̄ → σ}]] ** vd];
    Insert[ExtractVPB[q], d, 2]
   ]];
CF[vpb_VPB] := ExtractVPB[Γ̄[vpb]];
```

In[ ]:=
```
ExtractionGraph[O_] :=
 ExtractionGraph[O] = Module[{vd, n, gs, vs, es, p, m1, m2, g, q, k},
   gs = VPBGenerators[n = Count[vd = Γ̄[O], _EOS]];
   vs = {vd}; es = {}; p = 0;
   While[p < Length[vs],
    m1 = Count[vd = vs[[++p]], X_[_, _]];
    Do[
     m2 = Count[q = Γ̄[VD[VPB[n, g /. {σ → σ̄, σ̄ → σ}]] ** vd], X_[_, _]];
     If[m2 < m1,
      If[! MemberQ[vs, q], AppendTo[vs, q]];
      k = Position[vs, q][[1, 1]];
      AppendTo[es, Labeled[p ↦ k, g]]
     ],
     {g, gs}
    ]
   ];
   Graph[Table[Labeled[k, Length[vs[[k]]] - n], {k, p}], es]
  ]
```

In[•]:=
```
BR[3, {2, -1, -1, -1, -1}] // ExtractionGraph
```

Out[•]=

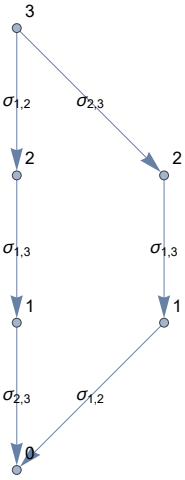
In[•]:=
```
VPB[BR[n_, is_List]] := VPB[n, Module[{π, i},
    π = Range[n];
    Sequence @@ Table[
      If[i > 0,
       π[[{i, i + 1}]] = π[[{i + 1, i}]]; σ_{π[[i+1]],π[[i]]},
       π[[{-i, -i + 1}]] = π[[{-i + 1, -i}]]; σ̄_{π[[-i]],π[[-i+1]]}
      ],
      {i, is}
     ]]];
VD[br_BR] := VD[VPB@br]
```

In[•]:=
```
RandomBraid[n_, m_] := BR[n, Table[RandomChoice[Range[n - 1] ⋃ (-Range[n - 1])], {m}]]
```

This is Demo.nb at http://drorbn.net/ap/Talks/TrendsInLDT-2005/.

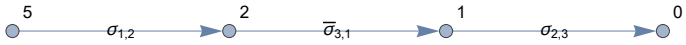
In[•]:= **BR[3, {1, 2, 1}] // ExtractionGraph**

Out[•]=



In[•]:= **BR[3, {1, -2, 1}] // ExtractionGraph**

Out[•]=



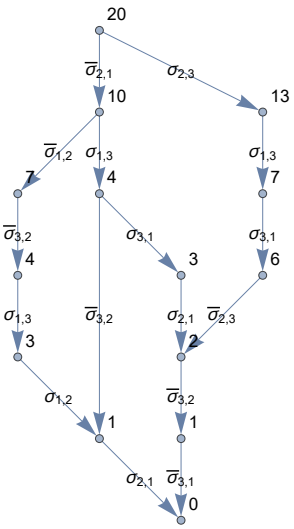
In[•]:= **BR[3, {1, -2, 1, -2}] // ExtractionGraph**

Out[•]=

In[•]:= **Knot[4, 1] // BR // Echo // ExtractionGraph**

KnotTheory: The minimum braids representing the knots with up to 10 crossings were provided by Thomas Gittings. See arXiv:math.GT/0401051.

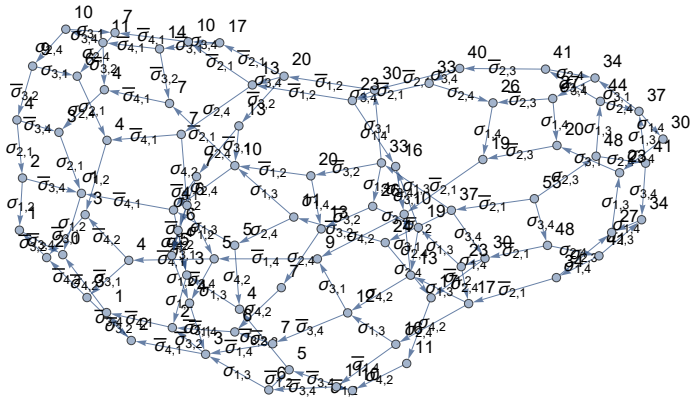
» BR[3, {-1, 2, -1, 2}]



In[•]:=
```
Knot[6, 1] // BR // Echo // ExtractionGraph
```

» `BR[4, {-1, -1, -2, 1, 3, -2, 3}]`

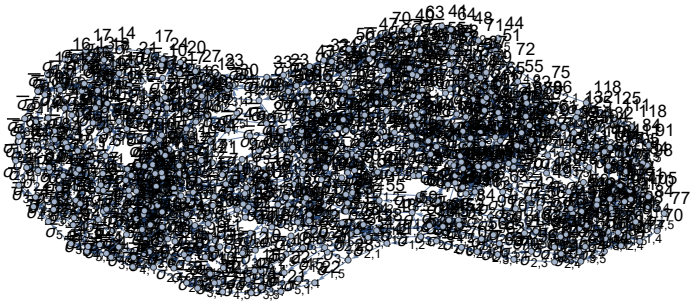
Out[•]=



In[•]:=
```
Knot[8, 1] // BR // Echo // ExtractionGraph
```

» `BR[5, {-1, -1, -2, 1, -2, -3, 2, 4, -3, 4}]`

Out[•]=



In[•]:=
```
Knot[8, 1] // BR // Echo // ExtractionGraph // VertexList // Length
```

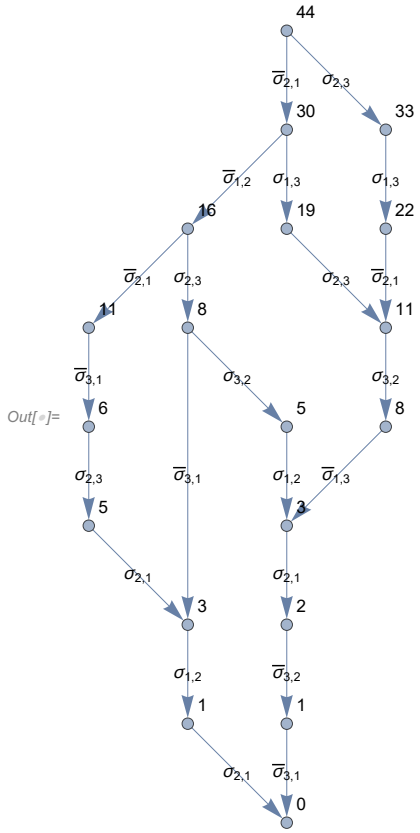
» `BR[5, {-1, -1, -2, 1, -2, -3, 2, 4, -3, 4}]`

Out[•]= 583

In[•]:=
```
Knot[6, 3] // BR // Echo // ExtractionGraph
```

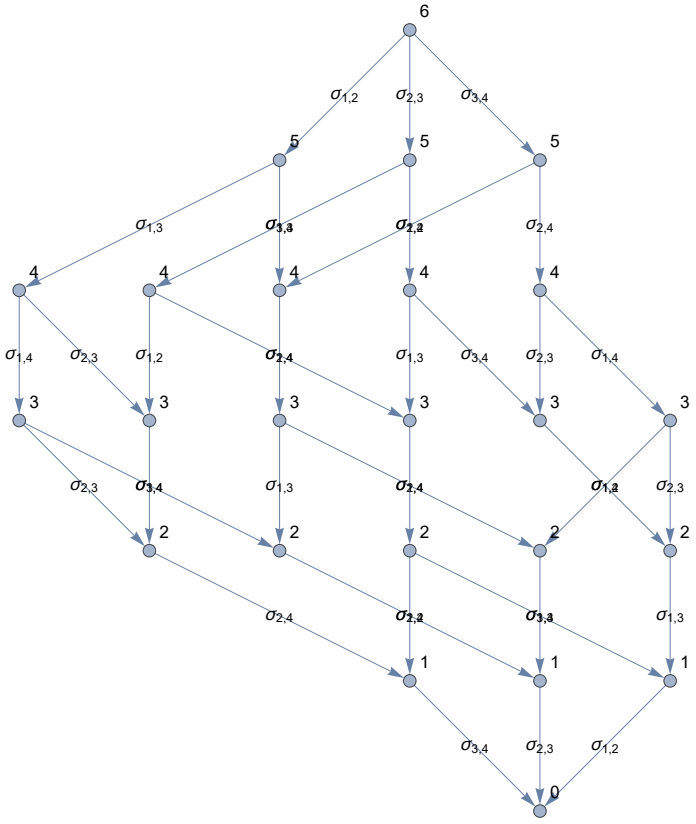
» BR[3, {-1, -1, 2, -1, 2, 2}]

Out[•]=



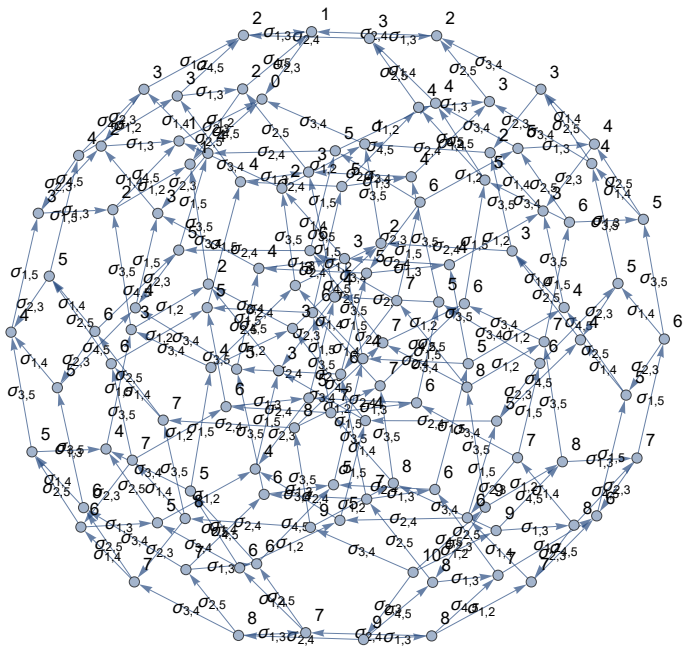
In[•]:=
```
BR[4, {1, 2, 3, 1, 2, 1}] // ExtractionGraph
```

Out[•]=



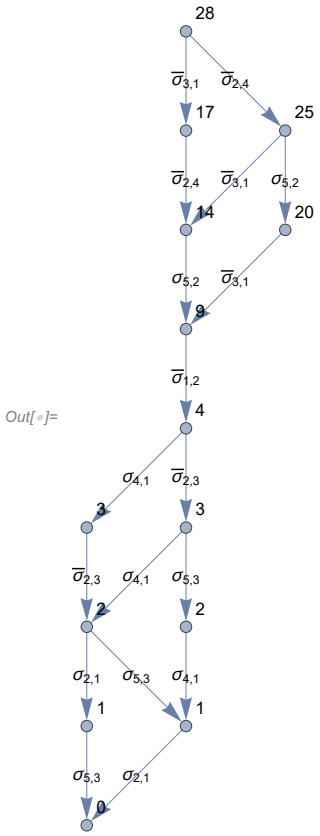
In[•]:=
```
BR[5, {1, 2, 3, 4, 1, 2, 3, 1, 2, 1}] // ExtractionGraph
```

Out[•]=



In[•]:=
```
VPB[5, σ̄3,1, σ̄2,4, σ5,2, σ̄1,2, σ4,1, σ̄2,3, σ2,1, σ5,3] // ExtractionGraph
```

Out[•]=



In[•]:=
```
BR[3, {2, -1, -1, -1, -1, -1, -1}] // ExtractionGraph
```

Out[•]=