

The HOMFLY Braidor Algebra - Programs

```
(
sigma[n_, i_, j_] := sigma[n, i, j] = (Range[n] /. {i -> j, j -> i});
Normalize[n_Integer, expr_] := Expand[expr //. {
  P[lft___, s1_List, s2_List, rgt___] -> P[lft, s2[[s1]], rgt],
  P[lft___, i_Integer, s_List, rgt___] -> P[lft, s, s[[i]], rgt],
  P[lft___, i_Integer, j_Integer, rgt___] /; i > j -> (
    P[lft, j, i, rgt] -
    x P[lft, sigma[n, i, j], j, rgt] + x P[lft, sigma[n, i, j], i, rgt]
  )
}];
Normalize[ASeries[n_, ps___]] := Normalize[n, ASeries[n, ps]];
Normalize[p1_P, p2_P] :=
  Normalize[p1, p2] = Normalize[Length[First[p1]], Join[p1, p2]];

ASeries /: a_ * ASeries[n_, ps__] := ASeries[n, Sequence @@ (a * {ps})];
ASeries /: ASeries[n_, ps1__] + ASeries[n_, ps2__] := Module[
  {d},
  d = Min[Length[{ps1}], Length[{ps2}]];
  ASeries[n, Sequence @@ (Take[{ps1}, d] + Take[{ps2}, d])]
];
ASeries /: ASeries[n_, ps1__] ** ASeries[n_, ps2__] := Module[
  {d, k, i},
  d = Min[Length[{ps1}], Length[{ps2}]] - 1;
  ASeries[n, Sequence @@ Table[
    Sum[
      Normalize[n, {ps1}][[1 + i]] ** {ps2}][[1 + k - i]],
      {i, 0, k}
    ],
    {k, 0, d}
  ]]
];

Unprotect[NonCommutativeMultiply];
NonCommutativeMultiply[a_] := a;
0 ** _ = 0;
_ ** 0 = 0;
(a_Plus) ** b_ := (# ** b) & /@ a;
a_ ** (b_Plus) := (a ** #) & /@ b;
(a1_.* p1_P) ** (a2_.* p2_P) := Expand[a1 a2 Normalize[p1, p2]];

```

```

iota[n_, expr_] := expr /. (p_P => (p /. s_List => Append[s, n+1]));
iota[ASeries[n_, ps_]] := ASeries[n+1, Sequence @@ iota[n, {ps}]];
Delta[p_P] := Delta[p] = Module[
  {n = Length[First[p]],
  Fold[Expand[#1 ** #2] &, P[Range[n+1]], (
    (P /@ (List @@ p)) /. {
      P[i_Integer] => P[Range[n+1, i+1] + x P[sigma[n+1, 1, i+1]],
      P[s_List] => P[Prepend[1+s, 1]]
    }
  )
  ]
];
Delta[n_, expr_] := Normalize[n+1, expr /. p_P => Delta[p]];
Delta[ASeries[n_, ps_]] := ASeries[n+1, Sequence @@ Delta[n, {ps}]];

NonDecreasingSequences[0, _] = {P[]};
NonDecreasingSequences[1, n_] := P /@ Range[n];
NonDecreasingSequences[L_, n_] /; L > 1 := Flatten[
  NonDecreasingSequences[L-1, n] /. p_P => (
    Append[p, #] & /@ Range[Last[p], n]
  )
];
AllGenerators[d_, n_] := Flatten[Table[
  x^k * Flatten[Outer[
    Join, P /@ Permutations[Range[n]], NonDecreasingSequences[d-k, n]
  ]],
  {k, 0, d}
]];

B[0] = ASeries[2, P[{2, 1}]];
B[1] = ASeries[2, P[{2, 1}], x P[{1, 2}]];

R3[B_] := iota[B] ** Delta[B] ** iota[B] - Delta[B] ** iota[B] ** Delta[B];

d1[twist_] := Normalize[2,
  Delta[1, twist] ** P[{2, 1}] - 0 P[{2, 1}] ** iota[1, twist]
  - P[{2, 1}] ** Delta[1, twist] + 0 iota[1, twist] ** P[{2, 1}]
];

d2[eps_] := Normalize[3,
  iota[2, eps] ** P[{2, 3, 1}]
  + (P[{2, 1, 3}] ** Delta[2, eps]) ** P[{2, 1, 3}]
  + P[{3, 1, 2}] ** iota[2, eps]
  - Delta[2, eps] ** P[{3, 1, 2}]
  - (P[{1, 3, 2}] ** iota[2, eps]) ** P[{1, 3, 2}]
  - P[{2, 3, 1}] ** Delta[2, eps]
];

d2inv[d_, err_] := Module[
  {v, mat},
  v = Coefficient[err, #] & /@ AllGenerators[d, 3];

```

```

mat = Outer[
  Coefficient[#2, #1] &, AllGenerators[d, 3], d2 /@ AllGenerators[d, 2]
];
LinearSolve[mat, v] . AllGenerators[d, 2]
];
d2ker[d_] := Module[
  {mat},
  mat = Outer[
    Coefficient[#2, #1] &, AllGenerators[d, 3], d2 /@ AllGenerators[d, 2]
  ];
  NullSpace[mat] . AllGenerators[d, 2]
];

d3[err_] := Normalize[4,
  P[{1, 2, 4, 3}] ** Iota[3, err] ** P[{1, 3, 4, 2}]
  + P[{2, 3, 4, 1}] ** Delta[3, err]
  - Delta[3, err] ** P[{4, 1, 2, 3}]
  - P[{1, 4, 2, 3}] ** Iota[3, err] ** P[{1, 2, 4, 3}]
  - P[{2, 3, 1, 4}] ** Delta[3, err] ** P[{2, 1, 3, 4}]
  - Iota[3, err] ** P[{2, 3, 4, 1}]
  + P[{4, 1, 2, 3}] ** Iota[3, err]
  + P[{2, 1, 3, 4}] ** Delta[3, err] ** P[{3, 1, 2, 4}]
];

FormalPower[expr_, d_] := NonCommutativeMultiply @@ Table[expr, {d}];
FormalExp[d_, n_, expr_] := Module[
  {t = P[Range[n]]},
  ASeries[n, t, Sequence @@ Table[Expand[(t = t ** expr) / k!], {k, d}]]
];

GeneratingSeries[perm_List, expr_] := expr /.
  P[l_List, ts___] => If[l != perm, 0, Times @@ t /@ {ts}];
GeneratingSeries[perm_List, ASeries[_ , ps___]] :=
  GeneratingSeries[perm, Plus @@ {ps}];
)

```