# Arrow Diagrams and gl(N) - Program

Joint project of Louis Leung and Dror Bar - Natan.

```
SetAttributes[Diag, Orderless];
Place[{ar}, {i_, j_}] := {Diag[ar[i, j]], Diag[ar[j, i]]};
Place[{ar, objs__}, {i_, rest__}] := Flatten[Table[
    Outer[Join,
     Place[{ar}, {i, {rest}[[k]]}],
     Place[{objs}, Delete[{rest}, k]]
    ],
    {k, Length[{rest}]}
   ]];
Place[{R6T}, {i_, j_, k_}] :=
  Permutations[{i, j, k}] /. {i1_, j1_, k1_} :> Diag[R6T[i1, j1, k1]];
Place[{R6T, objs__}, {i_, rest__}] := Flatten[Table[
    Outer[Join,
     Place[{R6T}, {i, {rest}[[j]], {rest}[[k]]}],
     Place[{objs}, Delete[{rest}, {{j}, {k}}]]
    ],
    {k, 2, Length[{rest}]}, {j, 1, k - 1}
   ]];
Diagrams[k_. * ar] := Place[Table[ar, {k}], Range[2 k]];
Diagrams[R6T] := Place[{R6T}, {1, 2, 3}];
Diagrams[R6T + k_. * ar] /; k > 0 := Flatten[
   Place[#, Range[2 k + 3]] & /@ Permutations[Table[ar, {k}] ~Append~ R6T]
  ];
Diagrams[R6T + k_. * ar] /; k < 0 := {};
NormalizeDiag[diag_Diag] := Module[
   {indices = Union @@ (List @@ diag /. ar → List)},
   diag /. Thread[indices → Range[Length[indices]]]
  ];
R[Diag[lft___, R6T[i_, j_, k_], rgt___]] := (
   + NormalizeDiag[Diag[lft, ar[i, j], ar[i + 0.5, k], rgt]]
    + NormalizeDiag[Diag[lft, ar[i, j], ar[j + 0.5, k], rgt]]
    + NormalizeDiag[Diag[lft, ar[i, k], ar[j, k + 0.5], rgt]]
    - NormalizeDiag[Diag[lft, ar[i, k], ar[i + 0.5, j], rgt]]
    - NormalizeDiag[Diag[lft, ar[i, j + 0.5], ar[j, k], rgt]]
    - NormalizeDiag[Diag[lft, ar[i, k + 0.5], ar[j, k], rgt]]
  );
DimAArrow[m_] /; m < 2 := Length[Diagrams[m ar]];
DimAArrow[m_] /; m ≥ 2 := Module[
  {diags, rels, mat, rel, i},
  diags = Diagrams[m ar];
  rels = R /@ Diagrams[R6T + (m - 2) ar];
  mat = SparseArray[
    Join @@ Table[
      rel = rels[[i]];
      {i, Position[diags, #][[1, 1]]} → Coefficient[rel, #] & /@
       Cases[rel, diag_Diag, Infinity],
      {i, Length[rels]}
     ],
    {Length[rels], Length[diags]}
   ];
  Length[diags] - MatrixRank[mat]
 ]
```

Next challenge: given a diag generate all the gl(N)-inspired equalities and innequalities cooresponding to it, and count the solutions by counting permutations.

```
OrderTypes[0] = {{}};
OrderTypes[1] = {{1}};
OrderTypes[l_List] := Module[{nl, snl},
   (
      snl = Union[nl = Append[l, #]];
      nl /. Thread[snl → Range[Length[snl]]]
     ) & /@ Range[1 / 2, 1 / 2 + Max[l], 1 / 2]
  ];
OrderTypes[n_Integer] := OrderTypes[n] = Join @@ (OrderTypes /@ OrderTypes[n - 1]);
Wgl[diag_Diag] := Wgl[diag] = Module[
   {p, eq, ltheq, rule, l, indices, ineqs},
   p = Times @@ (diag /. ar[i_, j_] :> eq[i, j - 1] eq[i - 1, j] ltheq[i - 1, i]);
   While[! FreeQ[p, eq],
    p = (p
         /. {rule = Cases[p, eq[i_, j_] :> (i → j), Infinity, 1];
          eq[ij__] :> (eq[ij] /. rule),
          ltheq[ij__] :> (ltheq[ij] /. rule)
         }
        /. {
         eq[i_, i_] → 1
        }
      )
   ];
   l = Length[indices = Union @@ Cases[{p}, ltheq[i_, j_] :> {i, j}, Infinity]];
   ineqs = p /. e_ltheq :> (e /. Thread[indices → Range[l]]);
   Expand[
    Plus @@ (Times[
          ineqs /. ltheq[i_, j_] :> Switch[
             Order[#[[i]], #[[j]]],
             1, 1,
             0, 1 / 2,
             -1, 0
            ],
          Binomial[n, Length[Union[#]]]
         ] & /@ OrderTypes[l])
   ]
  ]
```