In[•]:=
```
SetDirectory[NotebookDirectory[]]
```

In[•]:=
```
(* In the modules below,
we have cosidered the notation that {l1, l2} = {l2, l1}^{-1} *)

(*List which site is connected to which other sites via the spanning tree*)
getConnections[nSites_, nDim_, tree_] := Module[{treeConnections, sites},
   treeConnections = <||>;
   sites = Range[0, nSites^nDim - 1];
   Do[
    AppendTo[treeConnections, site → {}];
    Do[
     If[ IntersectingQ[{Sort[{site, site1}]}, tree],
      AppendTo[treeConnections[site], site1]
     ]
     , {site1, sites}]
    , {site, sites}];
   treeConnections];

(* We follow the norm that the 0th site is set to I. Starting from this,
we traverse the tree and obtain the gauge elements
 on each site that gives rise to the said spanning tree *)
getGaugeTransformation[nSites_, nDim_, tree_] := Module[
   {GaugeTransformation, sites, treeConnections, fixedSites, newFixedSites},
   sites = Range[0, nSites^nDim - 1];
   GaugeTransformation = <|sites[[1]] → {}|>;
   (* Group elements on each site, site → element*)

   treeConnections = getConnections[ nSites, nDim, tree];
   fixedSites = {{sites[[1]]}}; (*Starting from 0*)
   While[Not[Equal[Sort[Flatten[fixedSites]], sites]],
    (* While all sites are not fixed*)
    newFixedSites = {};
    (*Start from 0,
    and fix all the other end points of links starting from 0. These new end points
     that are fixed go to the newFixedSites, on which the next loop will run *)
    Do[
     Do[
      If[Not[KeyExistsQ[GaugeTransformation, connectedSite]],
       (*If site is not already fixed*)
       AppendTo[GaugeTransformation, connectedSite →
         Join[GaugeTransformation[site], {{site, connectedSite}}]]; (*on the left,
       the gauge transformation on 'site' will be acting. To set the link to I,
       we need to set the gauge transformation on the 'connectedSite' to '
        GaugeTransformation[site] x link' *)AppendTo[newFixedSites, connectedSite]]
      , {connectedSite, treeConnections[site]}
     ],
     {site, Last[fixedSites]}];
    AppendTo[fixedSites, newFixedSites];
   ];
   GaugeTransformation];
```

```
(* Given two trees,
equalityConstraints[site] gives the sites to be modified in order to preserve
 the unchanged linkes while setting the value of the lattice site 'site' *)
getEqualSites[nSites_, nDim_, tree1_, tree2_] :=
  Module[{sites, unchanged, equalityConstraints},
   sites = Range[0, nSites^nDim - 1];
   unchanged = Intersection[tree1, tree2];
   equalityConstraints = <||>;
   Do[
    AppendTo[equalityConstraints, site → {}];
    Do[
     If[ IntersectingQ[{Sort[{site, site1}]}, unchanged],
      AppendTo[equalityConstraints[site], site1]
     ]
     , {site1, sites}]
    , {site, sites}];
   equalityConstraints];

(* Returns all the links (unidirectional) for a given lattice *)
getLinks [nSites_, nDim_, neighbour_] := Module[{sites, links},
   sites = Range[0, nSites^nDim - 1];
   links = <||>;
   (* Association of all links → values. Please mind the abuse of notations. I
     am using (l1, l2) for denoting both the connection between l1 and l2,
   and also the value of the link element between l1 and l2*)
   Do[
    Do[
     If [Not[KeyExistsQ[links, Sort[{site, siteNeighbour}]]]
      , AppendTo[links, {site, siteNeighbour} → {site, siteNeighbour} ]
     ]
     , {siteNeighbour, neighbour[site]}]
    , {site, sites}];
   links];

(* Given a spanning tree,
obtain the gauge transformation using getGaugeTransformation,
and act upon the links by the gauge transformation*)
getModifiedLinks[nSites_, nDim_, links_, tree_] :=
  Module[{modifiedLinks, gaugeTransformation},
   modifiedLinks = <||>;
   gaugeTransformation = getGaugeTransformation[ nSites, nDim, tree];
   Do[
    AppendTo[modifiedLinks, link → Join[gaugeTransformation[link[[1]]],
       {link}, Reverse[gaugeTransformation[link[[2]]], {1, 2}]] ]
    (*The inverse of a product of gauge elements reverses the order
      of multiplication. At the same time we also reverse the
      order in the links since {l1, l2}^{-1} = {l2, l1}*)
    , {link, links}];
   modifiedLinks];
```

```
(* Given two spanning trees,
this obtains the gauge transformation relating one to another *)
gaugeTransFormationBetweenTwoTrees[nSites_, nDim_, tree1_, tree2_, neighbour_] :=
  Module[{links, modifiedLinks, toAdd, gaugeTransformation1to2,
    equalityConstraints, addedSites, added, newAdded, equalSites},

   links = getLinks[nSites, nDim, neighbour];
   modifiedLinks = getModifiedLinks[ nSites, nDim, links, tree1];
   toAdd = Complement[tree2, tree1];
   (*The links that are to be added to the first tree*)
   gaugeTransformation1to2 = <||>;
   Do[AppendTo[gaugeTransformation1to2, site → {}],
    {site, Range[0, nSites^nDim - 1]}];
   (* We start off with an identity element,
   i.e. for each site there is no gauge element *)

   equalityConstraints = getEqualSites[ nSites, nDim, tree1, tree2];

   Do[(*For each link to be added*)

    (* for a link {l1, l2},
    we set G[l1] = I and G[l2] = {l1, l2} to enforce the gauge transformation *)

    (* here we obtain first, all the sites that should
     be set equal to G[l2] according to the equality constraints *)
    addedSites = {{link[[2]]}};
    added = True;
    While[added,
     added = False;
     newAdded = {};
     Do[
      Do[
       If[Not@MemberQ[Flatten@addedSites, connected],
        added = True; AppendTo[newAdded, connected]]
       , {connected, equalityConstraints[site]}]
      , {site, Last[addedSites]}];
     AppendTo[addedSites, newAdded];
    ]; (* The above while loop, for every site added,
    checks the equality constraints and adds the other required sites also *)
    equalSites = Flatten@addedSites;

    Do[(* For each site to be set for the given link,
     i.e. for each site in equalSites*)
     gaugeTransformation1to2[site] =
       Join[gaugeTransformation1to2[[site]], modifiedLinks[link] ];
     , {site, equalSites}];

    , {link, toAdd}];
   gaugeTransformation1to2
  ];
```

```
In[•]:= nSites = 3;
        nDim = 2;
        Get[StringJoin[{"neighbours_", ToString[nSites] , "_" , ToString[nDim], ".mx"}]]
        (*Get the object "neighbour"*)
        Get[StringJoin[{"Trees_", ToString[nSites] , "_" , ToString[nDim], ".mx"}]]
        (*Get the object "Trees"*)

In[•]:= tree1 = Map[Sort, Trees[[11, 1]]]
        tree2 = Map[Sort, Trees[[11664, 1]]]
Out[•]=
        {{0, 1}, {1, 2}, {2, 5}, {3, 4}, {3, 6}, {3, 5}, {6, 7}, {6, 8}}
Out[•]=
        {{0, 1}, {0, 2}, {0, 3}, {0, 6}, {4, 5}, {6, 7}, {6, 8}, {5, 8}}

In[•]:= gaugeTransFormationBetweenTwoTrees[ nSites, nDim, tree1, tree2, neighbour]
Out[•]=
        ⟨| 0 → {}, 1 → {}, 2 → {{0, 2}, {2, 1}, {1, 0}},
         3 → {{0, 2}, {2, 1}, {1, 0}, {0, 3}, {3, 5}, {5, 2}, {2, 1}, {1, 0}}, 4 → {},
         5 → {{0, 1}, {1, 2}, {2, 5}, {5, 3}, {3, 4}, {4, 5}, {5, 2}, {2, 1}, {1, 0}},
         6 → {{0, 1}, {1, 2}, {2, 5}, {5, 3}, {3, 4}, {4, 5}, {5, 2}, {2, 1}, {1, 0},
            {0, 1}, {1, 2}, {2, 5}, {5, 8}, {8, 6}, {6, 3}, {3, 5}, {5, 2}, {2, 1}, {1, 0}},
         7 → {{0, 1}, {1, 2}, {2, 5}, {5, 3}, {3, 4}, {4, 5}, {5, 2}, {2, 1}, {1, 0},
            {0, 1}, {1, 2}, {2, 5}, {5, 8}, {8, 6}, {6, 3}, {3, 5}, {5, 2}, {2, 1}, {1, 0},
            {0, 1}, {1, 2}, {2, 5}, {5, 8}, {8, 6}, {6, 3}, {3, 5}, {5, 2}, {2, 1}, {1, 0}},
         8 → {{0, 6}, {6, 3}, {3, 5}, {5, 2}, {2, 1}, {1, 0}, {0, 6}, {6, 3}, {3, 5}, {5, 2}, {2, 1},
            {1, 0}, {0, 1}, {1, 2}, {2, 5}, {5, 8}, {8, 6}, {6, 3}, {3, 5}, {5, 2}, {2, 1}, {1, 0}} |⟩
```