Bocksteins for $SO(n)$

Agosto MA, Perez JJ

We were scooped!

> Martin Čadek, Mamoru Mimura, Jiří,Vanžura, *The cohomology rings of real Stiefel manifolds with integer coefficients*, J. Math. Kyoto Univ. **43** (2003), no. 2, 411-428. MR 2051031.

...and just in case one might have thought something was left, look at http://arxiv.org/pdf/0711.2541.

Bockstein homomorphisms are useful in computing the integer cohomology rings for the special orthogonal groups $SO(n)$ and related Stiefel manifolds. The following report describes some functions we have written in *Mathematica* to compute these homomorphisms. The notation we use is near that used by Hatcher, *Algebraic Topology*, available at http://www.math.cornell.edu/~hatcher/AT/ATpage.html.

In the code below, the input variable `maxdegree` is the highest class i of the β_i that will be used in the computation. The dimension of the defining representation space n for $SO(n)$ is defined by the input variable `dim`. The function `expand` expands monomials into vectors of generators, with the generator appearing as many times as its power in the monomial, thus

$$\texttt{expand}(\beta_1^2\beta_3^3) = \{\beta_1, \beta_1, \beta_3, \beta_3, \beta_3\}.$$

We use this because the Bockstein homomorphism satisfies a product rule

$$\beta(ab) = \beta(a)b + (-1)^{|a|}a\beta(b)$$

and it is convenient to thread it through a vector and then take the product of the elements of the result. That product function we use is called `crunch`. For example,

$$\texttt{crunch}(\{\beta_1, \beta_1, \beta_3, \beta_3, \beta_3\}) = \beta_1^2\beta_3^3$$

The function F_{gen} gives the value of the Bockstein homomorphism (usually denoted β, we call it F) on the monomials β_j . Thus $\beta(\beta_{2j-1}) = \beta_{2j}$, and $\beta(\beta_{2j}) = 0$, as in Hatcher, Example 3E.7, p 308. The notation in the code is `b[j]` for β_j so for example we have

$$\texttt{F(b[2j-1]) = b[2j]}, \text{and } \texttt{F(b[2j]) = 0}.$$

The function `F` computes the summands of the Bockstein homomorphism's values, taking into account the product rule. The summands in the product rule are arranged as rows of a matrix, the rows themselves representing the summand monomials. These

are given in a data type on which `expand` and `crunch` are defined. `F` acts on the k^{th} row in the k^{th} position. For example,

$$\texttt{F}(\{\beta_1,\beta_1,\beta_3,\beta_3\}) = \begin{pmatrix} F_{gen}(\beta_1) & \beta_1 & \beta_3 & \beta_3 \\ \beta_1 & \pm F_{gen}(\beta_1) & \beta_3 & \beta_3 \\ \beta_1 & \beta_1 & \pm F_{gen}(\beta_3) & \beta_3 \\ \beta_1 & \beta_1 & \beta_3 & \pm F_{gen}(\beta_3) \end{pmatrix}$$

Since *Mathematica*, by default, writes monomials in order of ascending subscripts, after the F_{gen} replacements are made, $F_{gen}(\beta_k)$ commutes with β_{k+1} and so `crunch` can be used to make monomials out of the rows of $\texttt{F}(\{\beta_1,\beta_1,\beta_3,\beta_3\})$ directly. The permutation signs therefore can be ignored.

The next step is to crunch and add the rows that F_{gen} generates, applying the relations (last paragraph of p 297, Hatcher) $\beta_i^2 = \beta_{2i}$ for $2i < n$, and $\beta_i^2 = 0$ for $2i \geq n$. These reduce the expressions to ones having minimal subscripts. This is effected in the `breduce` function by means of the *Mathematica* `ReplaceRepeated` command. The function `unbreduce` then returns each term to ones having maximal subscripts (and no exponents).

The function `bockstein` composes all these intermediate calculations and is defined on monomials in the β_i.

```
<< combinatorica‘

expand = Function[z, {
    y = {};
    Do[{
      Do[AppendTo[y, b[k]], {l, 1, Exponent[z, b[k]]}]
      },
     {k, 1, maxdegree}];
     Return[y, Function]
    }];

crunch = Function[z, {
    prod = 1;
    Do[{
      prod = prod z[[k]]
      },
     {k, 1, Dimensions[z][[1]]}];
     Return[prod, Function]
    }];

rank = Function[bbb, {
    index =
     Position[Append[Table[-b[j], {j, 1, maxdegree}], {0}], bbb][[1,
      1]];
```

```
    Return[index, Function]
    }];

F = Function[z, {
    Do[{
      Fmon[b[2 k - 1]] = b[2 k];
      Fmon[b[2 k]] = 0;
      },
     {k, 1, maxdegree/2}];
    accum = 1;
    z1 = {};
    Do[{
      temp = ReplacePart[z, Fmon[z[[k]]], k];
      z1 = Append[z1, temp];
      },
     {k, 1, Dimensions[z][[1]]}];
    Return[z1, Function]
    }];

breduce = Function[signedzz, {
    rules =
     Join[Table[b[2 n] -> b[n]^2, {n, 1, dim/2 + 1}],
      Table[b[n]^2 -> 0, {n, IntegerPart[dim/2 + .5], maxdegree}]];
    rules2 = Table[b[n] -> 0, {n, dim, maxdegree}];
    sum = 0;
    Do[{
      p1 = crunch[signedzz[[k]]] //. rules;
      p2 = unbreduce[p1] /. rules2;
      sum = sum + p2;
      },
     {k, 1, Dimensions[signedzz][[1]]}];
     Return[sum, Function]
    }];

unbreduce = Function[z, {
    z2 = z;
    If[z == 0, Return[0, Function]];
    Do[{
      newz = 1;
      Do[{
        term = b[k]^Exponent[z2, b[k]];
        slash = term //. b[k] -> b[2 k]^(1/2);
        slashex = Exponent[slash, b[2 k]];
        intex = IntegerPart[slashex];
        newterm1 = b[2 k]^intex;
        newterm2 = b[2 k]^(slashex - intex) //. b[2 k]^(1/2) -> b[k];
```

```
        newz = newz newterm1 newterm2;
        },
       {k, 1, maxdegree}];
      z2 = newz;
      },
     {iter, 1, maxdegree}];
    Return[newz, Function]
    }];

bockstein = Function[z, {
    answer = breduce[F[expand[z]]];
    Return[answer, Function]
    }];

vertprod = Function[z, {
    answer = 1;
    Do[{
      answer = answer*b[z[[k]]];
      },
     {k, 1, Dimensions[z][[1]]}] ;
    Return[answer, Function]
    }];

prodvert = Function[z, {
    If[z == 0, Return[0, Function]];
    answer = Table[rank[Part[z, i]], {i, 1, Dimensions[z][[1]]}];
    Return[answer, Function]
    }];

polytovert = Function[z, {
    zzz = ToString[z];
    If[StringCount[zzz, "+"] == 0, Return[{prodvert[z]}, Function]];
    zz = Table[Part[z, j], {j, 1, Dimensions[z][[1]]}];
    answer = Table[prodvert[zz[[k]]], {k, 1, Dimensions[z][[1]]}];
    Return[answer, Function]
    }];

vertbockvert = Function[z, {
    answer = polytovert[bockstein[vertprod[z]]];
    Return[answer, Function]
    }];

splitsumsbock = Function[z, {
    answer =
     Table[{z, vertbockvert[z][[k]]}, {k, 1,
       Dimensions[vertbockvert[z]][[1]]}];
```

```
    Return[answer, Function]
    }];

distpart = Function[{n, ceil}, {
    list2 = IntegerPartitions[n];
    l = Dimensions[list2][[1]];
    goodones = {};
    Do[{
      good = 1;
      Do[{
        If[
         list2[[k, j]] > ceil ||
          list2[[k, j]] == list2[[k, j - 1]], {good = 0, Break[]}]
        },
       {j, 1, Dimensions[list2[[k]]][[1]]}];
      If[
       good == 1,
       goodones = Append[goodones, Sort[list2[[k]]]];}, {k, 1, l}];
    Return[goodones, Function];
    }];

drawgraph = Function[dim, {
   dimension = dim (dim - 1)/2;
   ceiling = dim - 1;
   numrows =
    Dimensions[distpart[IntegerPart[dimension/2], ceiling]][[1]];

   (* make list of points (vertices for graph) *)
   Do[{
     labels[col] = distpart[col, ceiling];
     points[col] = {};
     Do[{
       points[col] = Append[points[col], {col, row}];
       },
      {row, 1, Dimensions[labels[col]][[1]]}];
     },
    {col, 0, IntegerPart[dimension/2]}];

   Do[{
     labels[col] = distpart[col, ceiling];
     points[col] = {};
     rowshift =
      Dimensions[labels[IntegerPart[dimension/2]]][[1]] -
       Dimensions[labels[col]][[1]];
     Do[{
       points[col] = Append[points[col], {col, row + rowshift}];
```

```
    },
   {row, 1, Dimensions[labels[col]][[1]]}];

  },
 {col, IntegerPart[dimension/2] + 1, dimension}];

masterlabels = Flatten[Table[labels[col], {col, 0, dimension}], 1];
masterpoints = Flatten[Table[points[col], {col, 0, dimension}], 1];

If[(adjmat = Import["adjmat" <> ToString[dim] <> ".csv"]) == $Failed,
 {
  mm = Flatten[Map[splitsumsbock, masterlabels], 1];

  (* mm is the Bockstein homomorphism written out as a list of
     pairs of connected vertices *)

  zeroes = Partition[Position[mm, 0][[All, 1]], 1];

  (* zeroes finds the positions in the mm table of the Bocksteins
     that take the value zero.*)

  lines = Delete[mm, zeroes];

  (* lines is the Bockstein table mm with the zero-image terms
     omitted. It is these pairs that will be represented with
     connecting lines in the graph.*)

  sa =
   SparseArray[
    Flatten[
     Table[{Flatten[Position[masterlabels, lines[[k, 1]]]][[1]],
        Flatten[Position[masterlabels, lines[[k, 2]]]][[1]]} ->
       1, {k, 1, Dimensions[lines][[1]]}], 0]];

  (* Position[masterlabels,lines[[k,1]]]] finds the origins of
     lines and Position[masterlabels,lines[[k,2]]]] finds the ends.
     At the point {Position[masterlabels,lines[[k,1]]]],
     Position[masterlabels,lines[[k,2]]]]} a one is placed in a
     sparse array. *)

  adjmat = Normal[sa];  (* Makes a matrix of the array. *)

  Export["adjmat" <> ToString[dim] <> ".csv", adjmat];
  }];
```

Sample calculations: The monomial is expanded to a vector.

```
maxdegree = 36;
dim = 7;
z=b[1]^4 b[3]^7 b[4]^3;
y=expand[z]
> {b[1],b[1],b[1],b[1],b[3],b[3],b[3],b[3],b[3],b[3],b[3],b[4],b[4],b[4]}
```

Here, `F` is applied to the vector immediately above. It has replaced the `b[1]` in the first position with a `b[2]` in the first row. In the second row, the `b[2]` is now in the second position, and so on. In the last three rows, `b[4]` is taken to zero by the Bockstein.

```
F[y]

> {{b[2],b[1],b[1],b[1],b[3],b[3],b[3],b[3],b[3],b[3],b[3],b[4],b[4],b[4]},
  {b[1],b[2],b[1],b[1],b[3],b[3],b[3],b[3],b[3],b[3],b[3],b[4],b[4],b[4]},
  {b[1],b[1],b[2],b[1],b[3],b[3],b[3],b[3],b[3],b[3],b[3],b[4],b[4],b[4]},
  {b[1],b[1],b[1],b[2],b[3],b[3],b[3],b[3],b[3],b[3],b[3],b[4],b[4],b[4]},
  {b[1],b[1],b[1],b[1],b[4],b[3],b[3],b[3],b[3],b[3],b[3],b[4],b[4],b[4]},
  {b[1],b[1],b[1],b[1],b[3],b[4],b[3],b[3],b[3],b[3],b[3],b[4],b[4],b[4]},
  {b[1],b[1],b[1],b[1],b[3],b[3],b[4],b[3],b[3],b[3],b[3],b[4],b[4],b[4]},
  {b[1],b[1],b[1],b[1],b[3],b[3],b[3],b[4],b[3],b[3],b[3],b[4],b[4],b[4]},
  {b[1],b[1],b[1],b[1],b[3],b[3],b[3],b[3],b[4],b[3],b[3],b[4],b[4],b[4]},
  {b[1],b[1],b[1],b[1],b[3],b[3],b[3],b[3],b[3],b[4],b[3],b[4],b[4],b[4]},
  {b[1],b[1],b[1],b[1],b[3],b[3],b[3],b[3],b[3],b[3],b[4],b[4],b[4],b[4]},
  {b[1],b[1],b[1],b[1],b[3],b[3],b[3],b[3],b[3],b[3],b[3], 0, b[4],b[4]},
  {b[1],b[1],b[1],b[1],b[3],b[3],b[3],b[3],b[3],b[3],b[3],b[4], 0, b[4]},
  {b[1],b[1],b[1],b[1],b[3],b[3],b[3],b[3],b[3],b[3],b[3],b[4],b[4], 0 }}
```

Here we do some sample calculations for $SO(7)$. Refer to Hatcher p 309.

```
bockstein[b[1]]
> b[2]

bockstein[b[1]b[3]]
> b[2]b[3] + b[1]b[4]

bockstein[b[1]b[3]b[5]]
> b[2]b[3]b[5] + b[1]b[4]b[5] + b[1]b[3]b[6]

bockstein[b[1]b[2]b[3]b[4]]
> 0
```

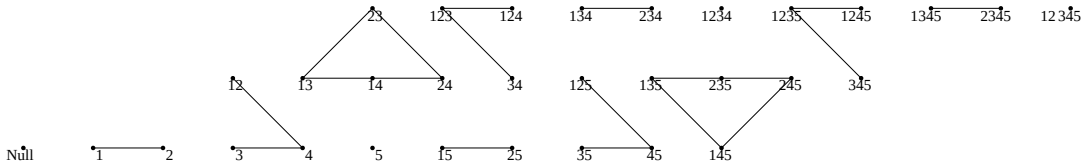
Below is the first application of our code. We draw the graphs as in Hatcher, using *Combinatorica*.

```
adjmatrix = drawgraph[dim];
```

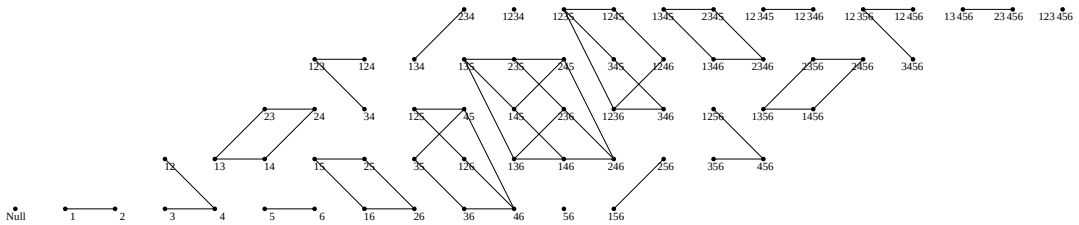
```
g = SetGraphOptions[
      FromAdjacencyMatrix[adjmatrix, masterpoints],
      VertexLabelPosition -> {0.02, -0.02},
      ImageSize -> 50 dimension,
      VertexStyle -> Disk[0.001],
      EdgeStyle -> Thickness[0.0002]
      ];

labelg = ShowLabeledGraph[
      g,
      Table[Style[ToExpression[StringReplace[
            ToString[masterlabels[[n]]],
            {"," -> "", "{" -> "", "}" -> "", " " -> ""}
            ]], Small],
            {n, 1, Dimensions[masterlabels][[1]]}
            ]
      ]
```

Agosto and Perez thank Prof Hatcher for helpful emails.

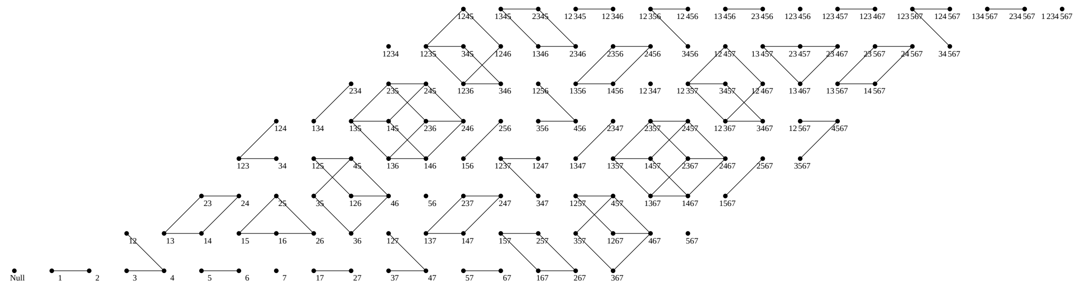
We include some sample graphs below:



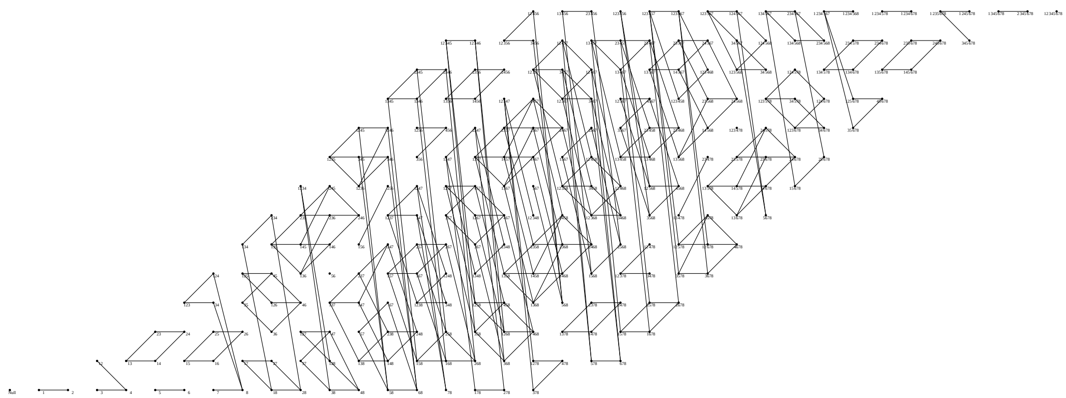
$SO(6)$



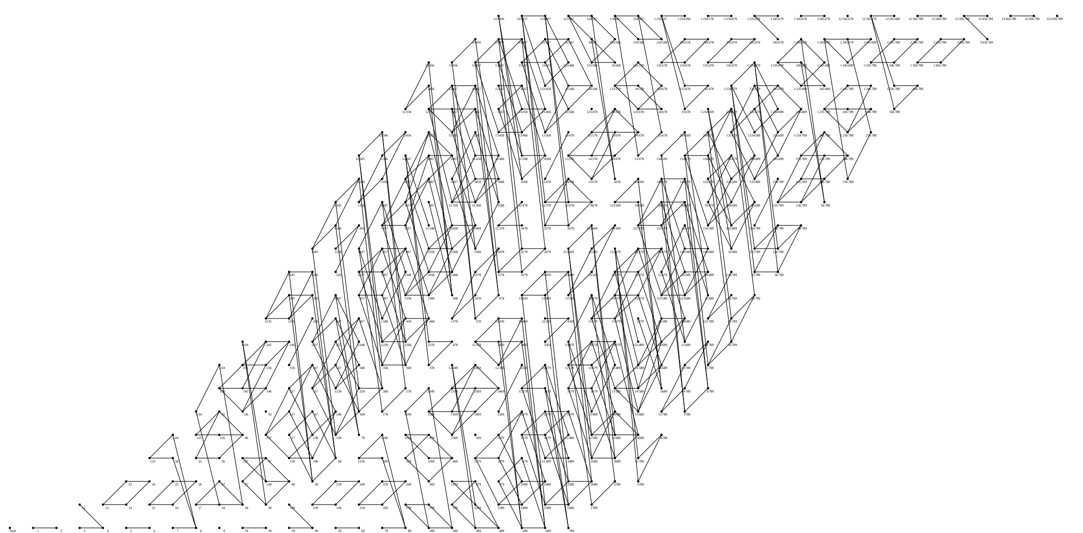
$SO(7)$


1245 1345 2345 12345 12346 12356 12456 13456 23456 123456 123457 123467 123567 124567 134567 234567 1234567
1234 1235 345 1246 1346 2346 2356 2456 3456 12457 13457 23457 23467 23567 24567 34567
234 235 245 1236 346 1256 1356 1456 12347 12357 3457 12467 13467 13567 14567
124 134 135 145 236 246 256 356 456 2347 2357 2457 12367 3467 12567 4567
123 34 125 45 136 146 156 1237 1247 1347 1357 1457 2367 2467 2567 3567
23 24 25 35 126 46 56 237 247 347 1257 457 1367 1467 1567
12 13 14 15 16 26 36 127 137 147 157 257 357 1267 467 567
Null 1 2 3 4 5 6 7 17 27 37 47 57 67 167 267 367

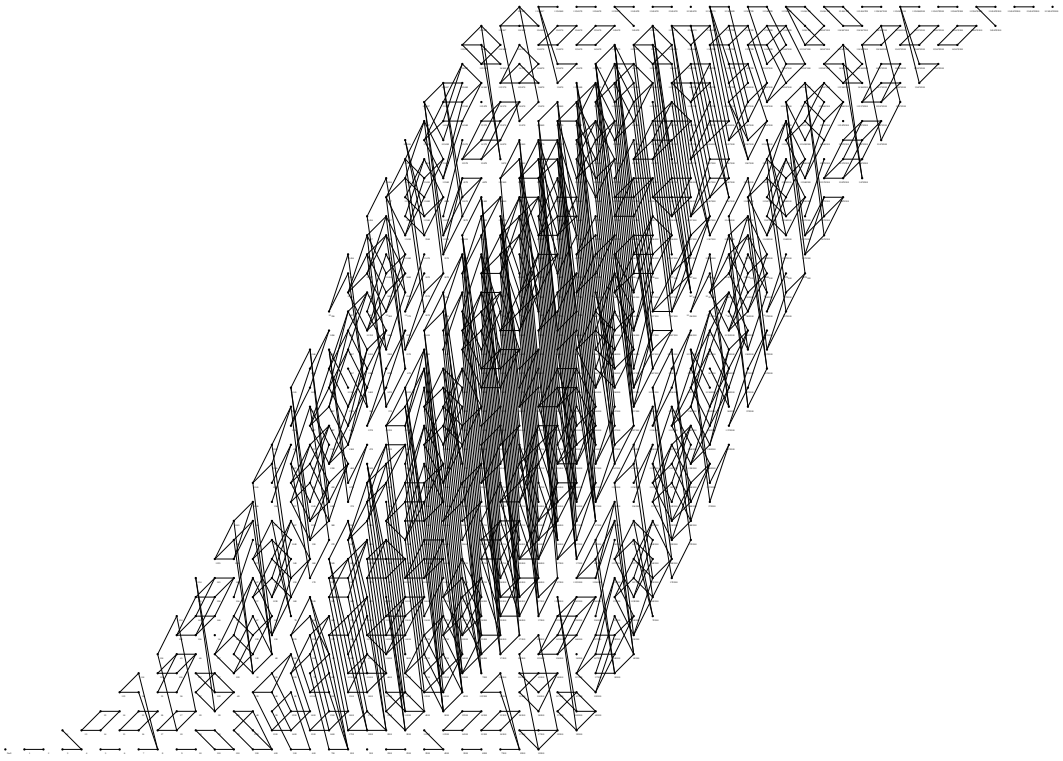
$SO(8)$



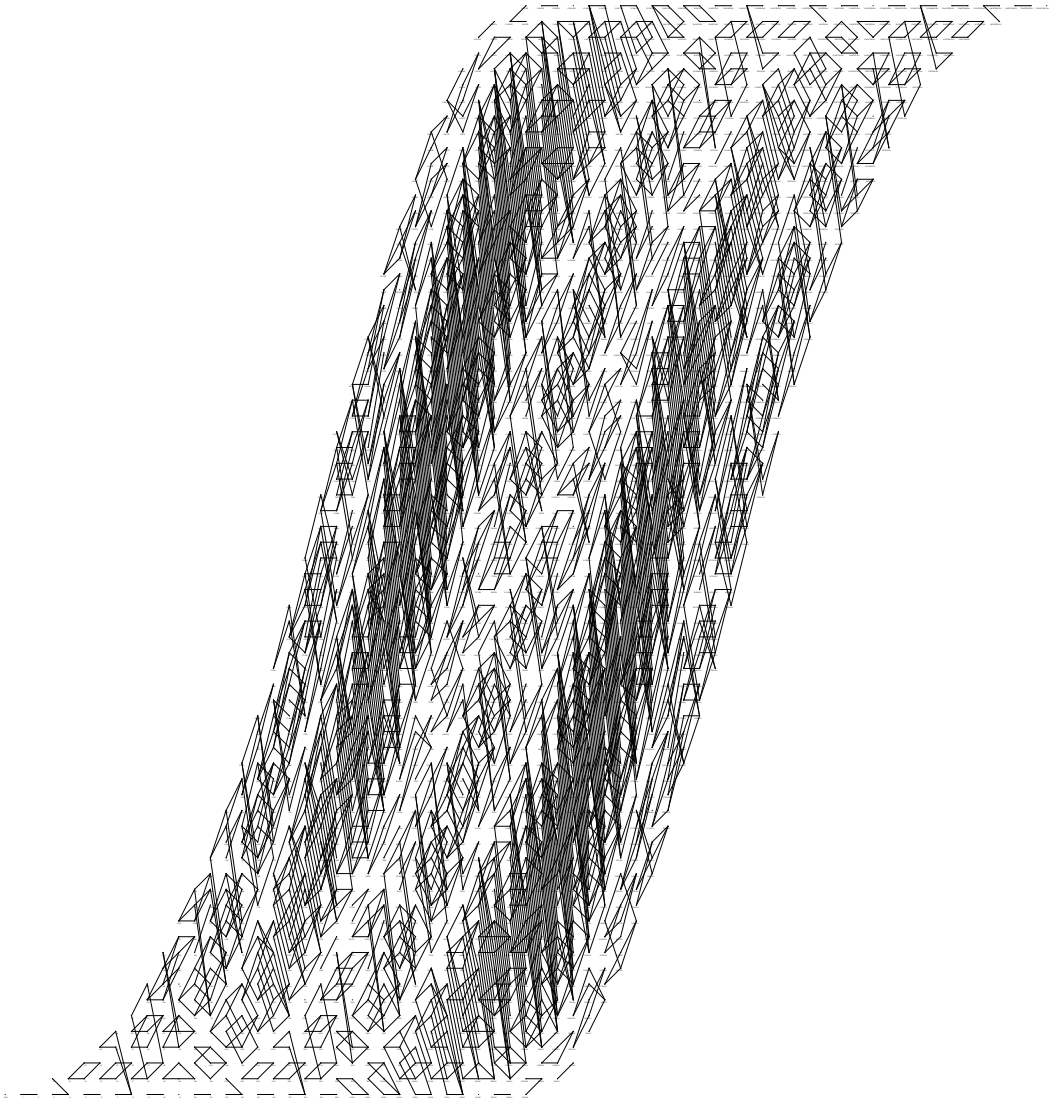
$SO(9)$



$SO(10)$



$SO(11)$



$SO(12)$