Tutorial: visualizing root system

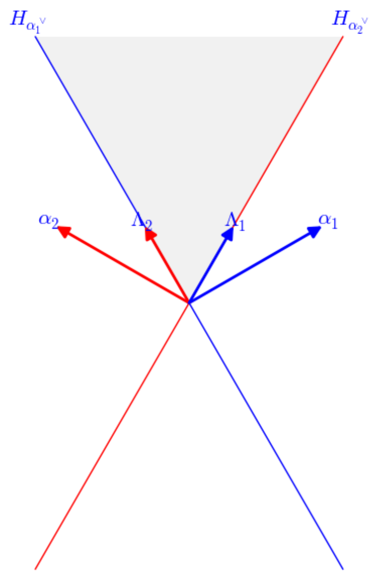
Root systems encode the positions of collections of hyperplanes in space, and form the fundamental combinatorial data underlying Coxeter and Weyl groups, Lie algebras and groups, etc. The theory can be a bit intimidating at first because of the many technical gadgets (roots, coroots, weights, ...). Vizualizing them goes a long way toward building a geometric intuition.

This tutorial starts from simple plots and guides you all the way to advanced plots with your own combinatorial data drawn on top of it.

## First plots

In this first plot, we draw the root system for type $A_2$ in the ambient space. It is generated from two hyperplanes at a 120 degree angle:

```
L = RootSystem(["A",2]).ambient_space()
L.plot()
```



Each of those hyperplane $H_{alpha^vee_i}$ is described by a linear form $alpha_i^vee$ called simple coroot. To each such hyperplane corresponds a reflection along a vector called root. In this picture, the reflections are orthogonal and the two simple roots $alpha_1$ and $alpha_2$ are vectors which are normal to the reflection hyperplanes. The same color code is used uniformly: blue for 1, red for 2, green for 3, ... (see :meth:`CartanType.color). The fundamental weights, `\Lambda_1` and $Lambda_2$ form the dual basis of the coroots.

System Message: ERROR/3 (`<string>`, line 30); *backlink*

Unknown interpreted text role "meth".

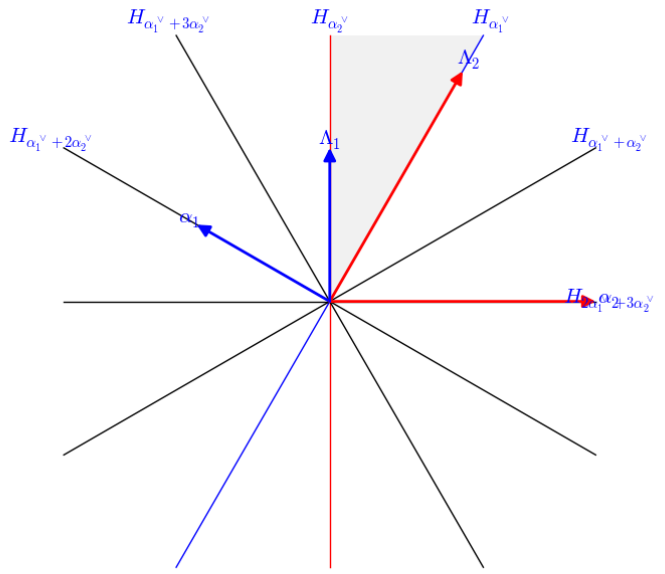
The two reflections generate a group of order six which is nothing but the usual symmetric group $S_3$, in its natural action by permutations of the coordinates of the ambient space. Wait, but the ambient space should be of dimension 3 then? That's perfectly right. Here is the full picture in 3D:

```
L = RootSystem(["A",2]).ambient_space()
L.plot(projection=False)
```

Toggle Advanced Controls | Help for Jmol 3-D viewer

However in this space, the line $(1,1,1)$ is fixed by the action of the group. Therefore, the so called barycentric projection orthogonal to $(1,1,1)$ gives a convenient 2D picture which contains all the essential information. The same projection is used by default in type $G_2$:

```
L = RootSystem(["G",2]).ambient_space()
L.plot(reflection_hyperplanes="all")
```


$H_{\alpha_1^\vee+3\alpha_2^\vee}$
$H_{\alpha_2^\vee}$
$H_{\alpha_1^\vee}$
$\Lambda_2$
$H_{\alpha_1^\vee+2\alpha_2^\vee}$
$\Lambda_1$
$H_{\alpha_1^\vee+\alpha_2^\vee}$
$\alpha_1$
$H_{2\alpha_1^\vee+3\alpha_2^\vee}$
$\alpha_2$

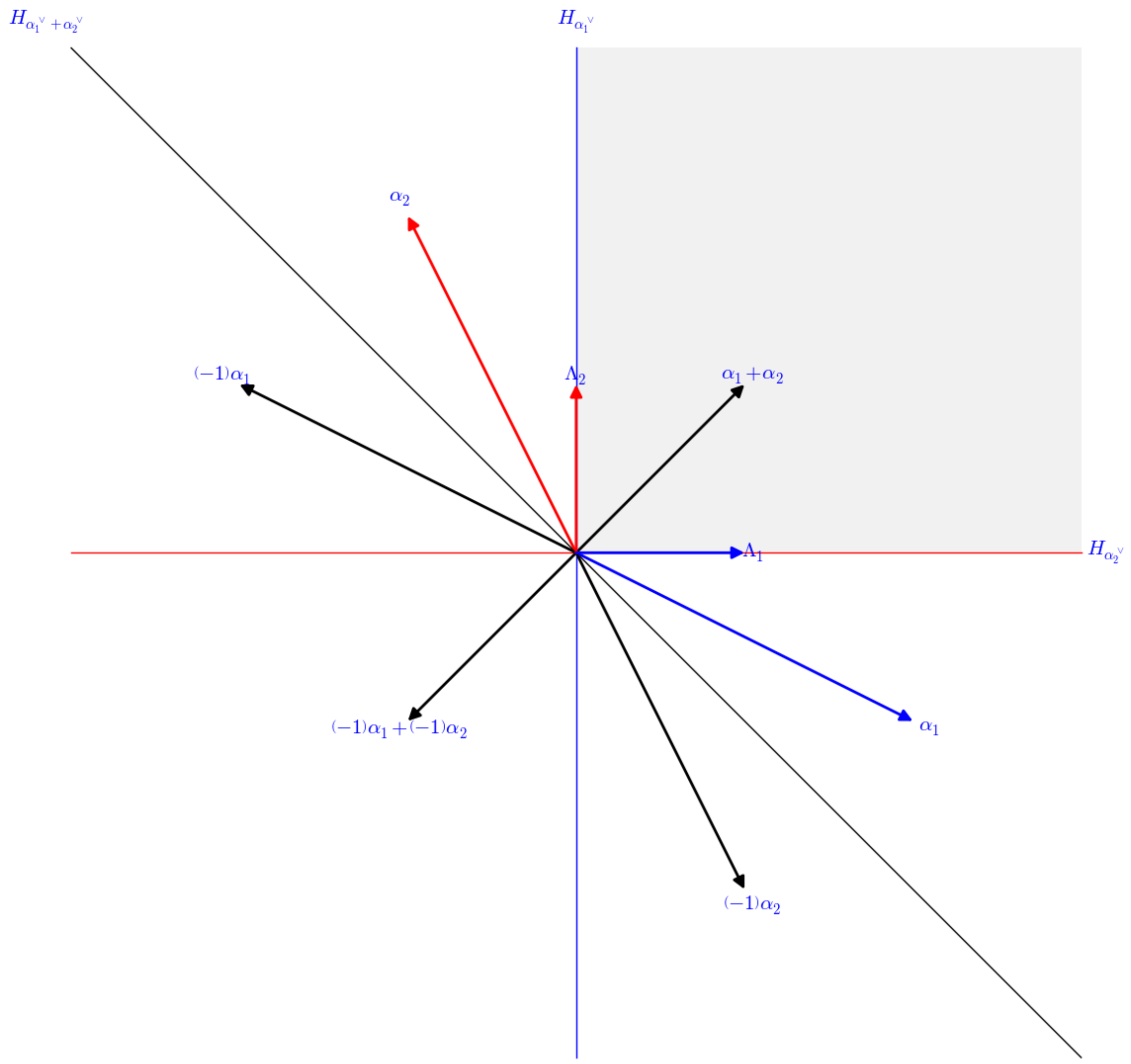
The group is now the dihedral group of order 12, generated by the two reflections $s_1$ and $s_2$. The picture displays the hyperplanes for all 12 reflections of the group. Those reflections delimit 12 chambers which are in one to one correspondance with the elements of the group. The fundamental chamber, which is grayed out, is associated with the identity of the group.

Warning

The fundamental chamber is currently plotted as the cone generated by the fundamental weights; as can be seen on the previous 3D picture this is not quite correct if the fundamental weights do not span the space.

Coming back to the symmetric group, here is the picture in the weight space, with all roots and all reflection hyperplanes; remark that, unlike in the ambient space, a root is not necessarily orthogonal to its corresponding reflection hyperplane:

```
L = RootSystem(["A",2]).weight_space()
L.plot(roots = "all", reflection_hyperplanes="all").show(figsize=15)
```



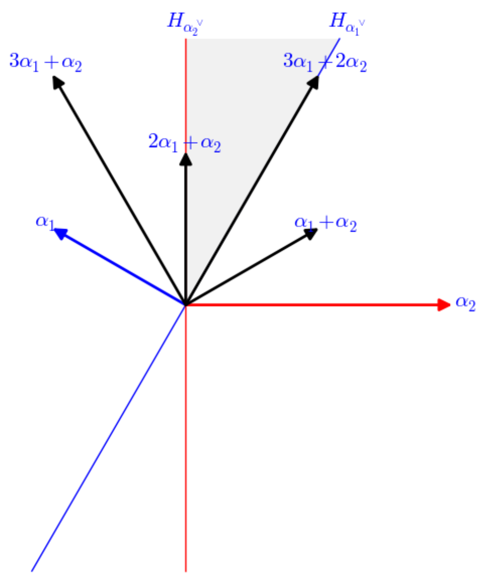
Note

Setting a larger figure size as above can help reduce the overlap between the text labels when the figure gets crowded.

One can further customize which roots to display, as in the following example showing the positive roots in the weight space for type ['G',2], labelled by their coordinates in the root lattice:

```
Q = RootSystem(["G",2]).root_space()
L = RootSystem(["G",2]).ambient_space()
L.plot(roots=list(Q.positive_roots()), fundamental_weights=False)
```

evaluate



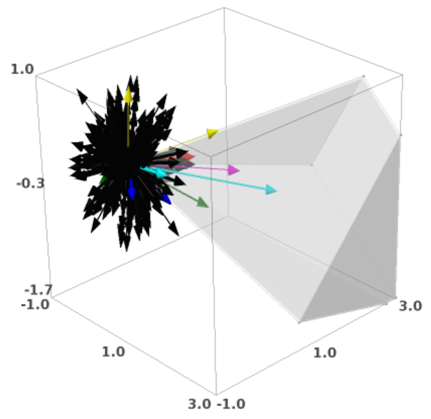
One can also customize the projection by specifying a function. Here, we display all the roots for type $E_8$ using the projection from its eight dimensional ambient space onto 3D described on :wikipedia:`Wikipedia's E8 3D picture <File:E8_3D.png>`:

```
M = matrix([[0., -0.556793440452, 0.19694925177, -0.19694925177, 0.0805477263944, -0.385290876171, 0., 0.385290876171],
              [0.180913155536, 0., 0.160212955043, 0.160212955043, 0., 0.0990170516545, 0.766360424875, 0.0990170516545],
              [0.338261212718, 0, 0, -0.338261212718, 0.672816364803, 0.171502564281, 0, -0.171502564281]])
L = RootSystem(["E",8]).ambient_space()
L.dimension()
```

```
    8
```

```
L.plot(roots="all", reflection_hyperplanes=False, projection=lambda v: M*vector(v), labels=False)
```

Sleeping... Make Interactive



The projection function should be linear or affine.

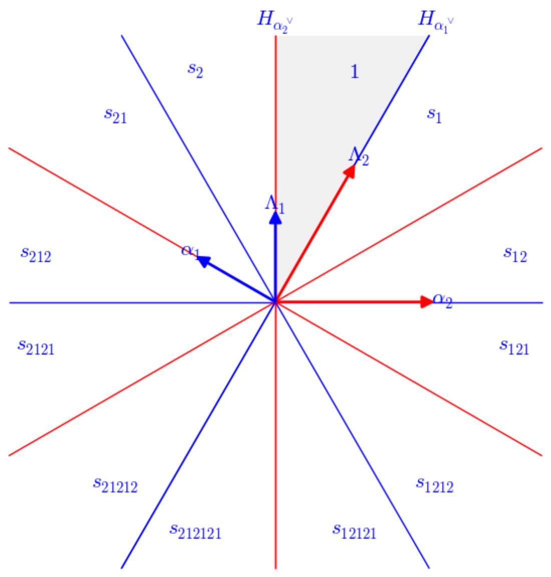
Exercise

Draw all finite root systems in 2D, using the canonical projection onto their Coxeter plane. See Stembridge's page.

## Alcoves and chambers

We now draw the root system for type $G_2$, with its alcoves (in finite type, those really are the chambers) and the corresponding elements of the Weyl group. We enlarge a bit the bounding box to make sure everything fits in the picturex:

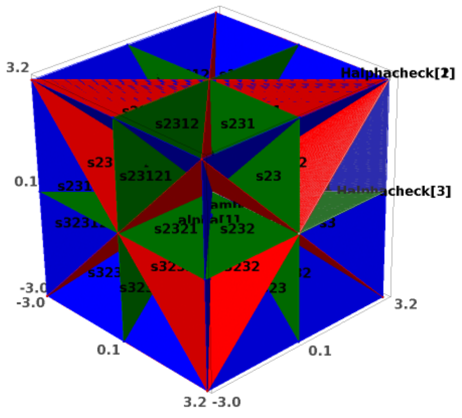
```
RootSystem(["G",2]).ambient_space().plot(alcoves=True, alcove_labels=True, bounding_box=5)
```



The same picture in 3D, for type $B_3$:

```
RootSystem(["B",3]).ambient_space().plot(alcoves=True, alcove_labels=True)
```

Sleeping... Make Interactive



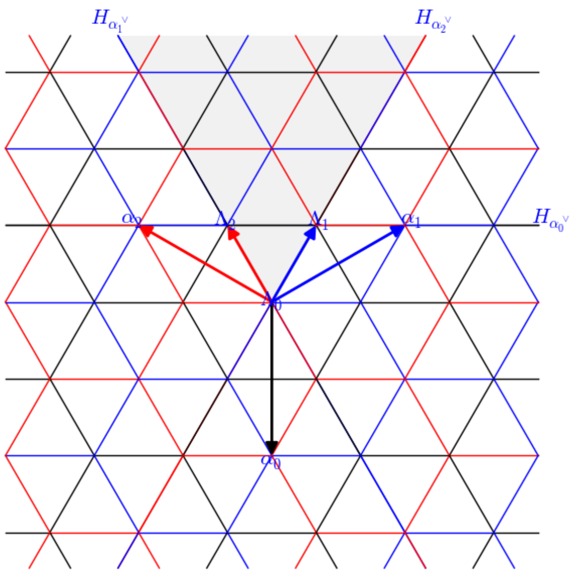
Exercise

Can you spot the fundamental chamber? The fundamental weights? The simple roots? The longest element of the Weyl group?

## Alcove pictures for affine types

We now draw the usual alcove picture for affine type $A_2^1$:

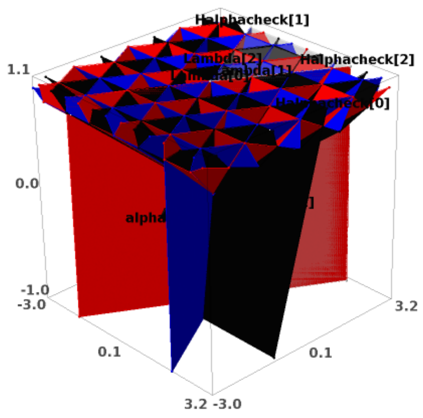
```
L = RootSystem(["A",2,1]).ambient_space()
L.plot()
```



This picture is convenient because it is low dimensional and contains most of the relevant information. Beside, by choosing the ambient space, the elements of the Weyl group act as orthogonal affine maps. In particular, reflections are usual (affine) orthogonal reflections. However this is in fact only a slice of the real picture: the Weyl group actually acts by linear maps on the full ambient space. Those maps stabilize the so-called level $l$ hyperplanes, and we are visualizing here what's happening at level 1. Here is the full picture in 3D:

```
L.plot(bounding_box=[[-3,3],[-3,3],[-1,1]], affine=False)
```

Sleeping... Make Interactive

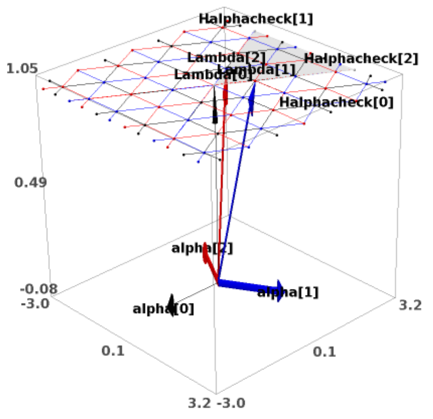


In fact, in type $A$, this really is a picture in 4D, but as usual the barycentric projection kills the boring extra dimension for us.

It's usually more readable to only draw the intersection of the reflection hyperplanes with the level 1 hyperplane:

```
L.plot(bounding_box=[[-3,3],[-3,3],[-1,1]], affine=False, level=1)
```

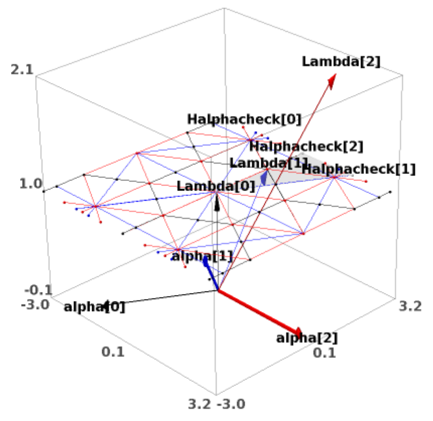
Sleeping... Make Interactive



Such 3D pictures are useful to better understand technicalities, like the fact that the fundamental weights do not necessarily all live at level 1:

```
L = RootSystem(["G",2,1]).ambient_space()
L.plot(affine=False, level=1)
```

Sleeping... Make Interactive

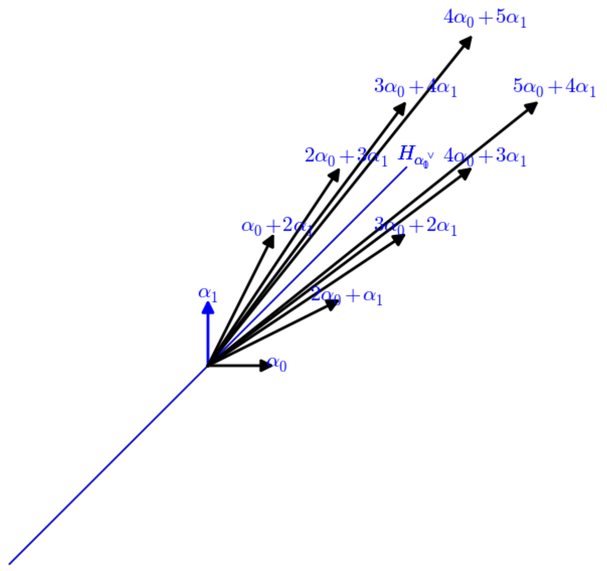


Exercise

Draw the alcove picture at level 1, and compare the position of the fundamental weights and the vertices of the fundamental alcove.

Even 2D pictures of the rank 1 one cases can give some food for thought. Here, we draw the root lattice, with the positive roots of small height in the root poset:

```
L = RootSystem(["A",1,1]).root_lattice()
positive_roots = TransitiveIdealGraded(attrcall("pred"), L.simple_roots())
it = iter(positive_roots)
first_positive_roots = [it.next() for i in range(10)]
L.plot(roots=first_positive_roots, affine=False, alcoves=False)
```



Exercises

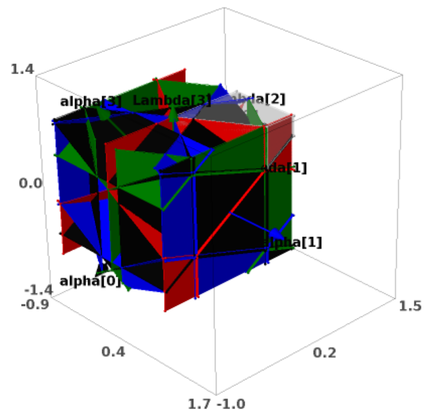
1. Use the same trick to draw the reflection hyperplanes in the weight lattice for the coroots of small height.
2. Draw the positive roots in the weight lattice and in the extended weight lattice
3. Draw the reflection hyperplanes in the root lattice

## Higher dimension affine pictures

We now do some plots for rank 4 affine types, at level 1. The space is tiled by the alcoves, each of which is a 3D simplex:

```
L = RootSystem(["A",3,1]).ambient_space()
L.plot(reflection_hyperplanes=False, bounding_box=85/100)
```

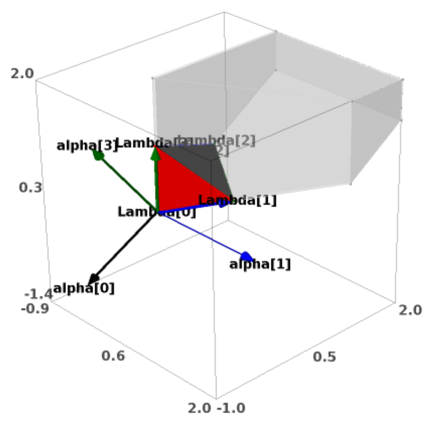
Sleeping... Make Interactive



It is recommended to use a small bounding box here, for otherwise the number of simplices grows quickly what Sage can handle smoothly. Beside we would need some sort of transparency or wireframe drawing to actually see something useful. An alternative is to specify explicitly which alcoves to visualize. Here is the fundamental alcove, specified by an element of the Weyl group:

```
W = L.weyl_group()
L.plot(reflection_hyperplanes=False, alcoves=[W.one()], bounding_box=2)
```

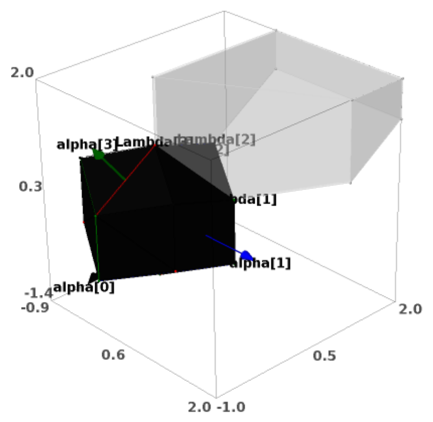
Sleeping... Make Interactive



and the fundamental polygon, specified by the coordinates of its center in the root lattice:

```
W = L.weyl_group()
L.plot(reflection_hyperplanes=False, alcoves=[[0,0]], bounding_box=2)
```

Sleeping... Make Interactive



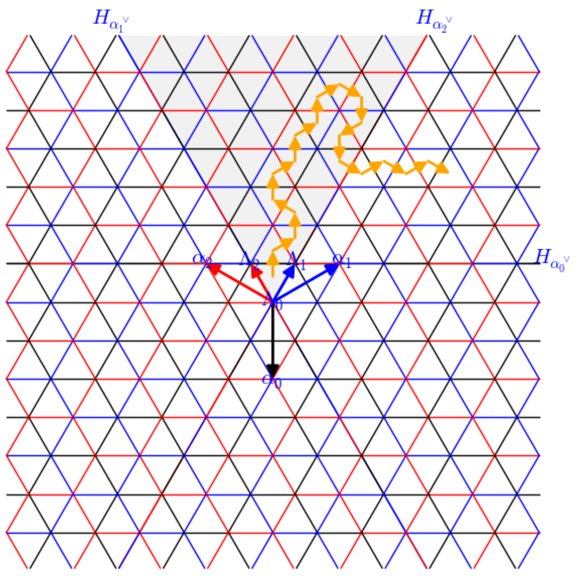
Exercise

1. Draw the fundamental alcove in the ambient space, just by itself (no reflection hyperplane, root, ...). The automorphism group of the Dynkin diagram for $A_3^1$ (a cycle of length 4) is the dihedral group. Visualize the corresponding symmetries of the fundamental alcove.
2. Draw the fundamental alcoves for the other rank 4 affine types, and recover the automorphism groups of their Dynkin diagram from the pictures.

## Drawing on top of a root system plot

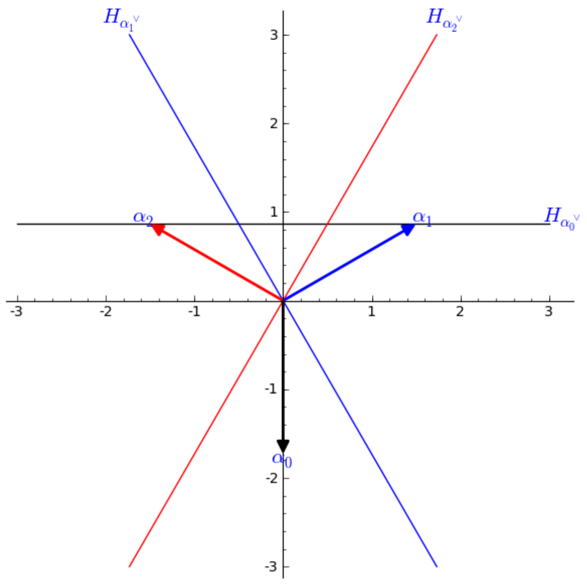
The root system plots have been designed to be used as wallpaper on top of which to draw more information. In the following example, we draw an alcove walk, specified by a word of indices of simple reflections, on top of the weight lattice in affine type $A_{2,1}$:

```
L = RootSystem(["A",2,1]).ambient_space()
w1 = [0,2,1,2,0,2,1,0,2,1,2,1,2,0,2,0,1,2,0]
L.plot(alcove_walk=w1, bounding_box=6)
```



Now, what about drawing several alcove walks, and specifying some colors? A single do-it-all plot method would be cumbersome; so instead, it is actually built on top of many methods (see the list below) that can be called independently and combined at will:

```
L.plot_roots() + L.plot_reflection_hyperplanes()
```



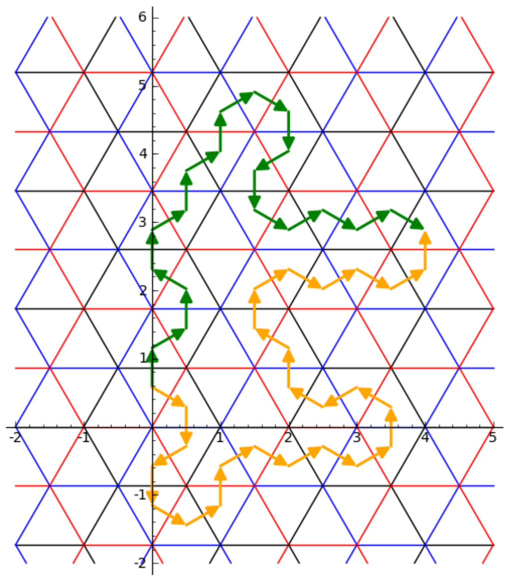
Note

By default the axes are disabled in root system plots since they tend to polute the picture. Annoyingly they come back when combining them. Here is a workaround:

```
sage: p = L.plot_roots() + L.plot_reflection_hyperplanes()
sage: p.axes(False)
sage: p
```

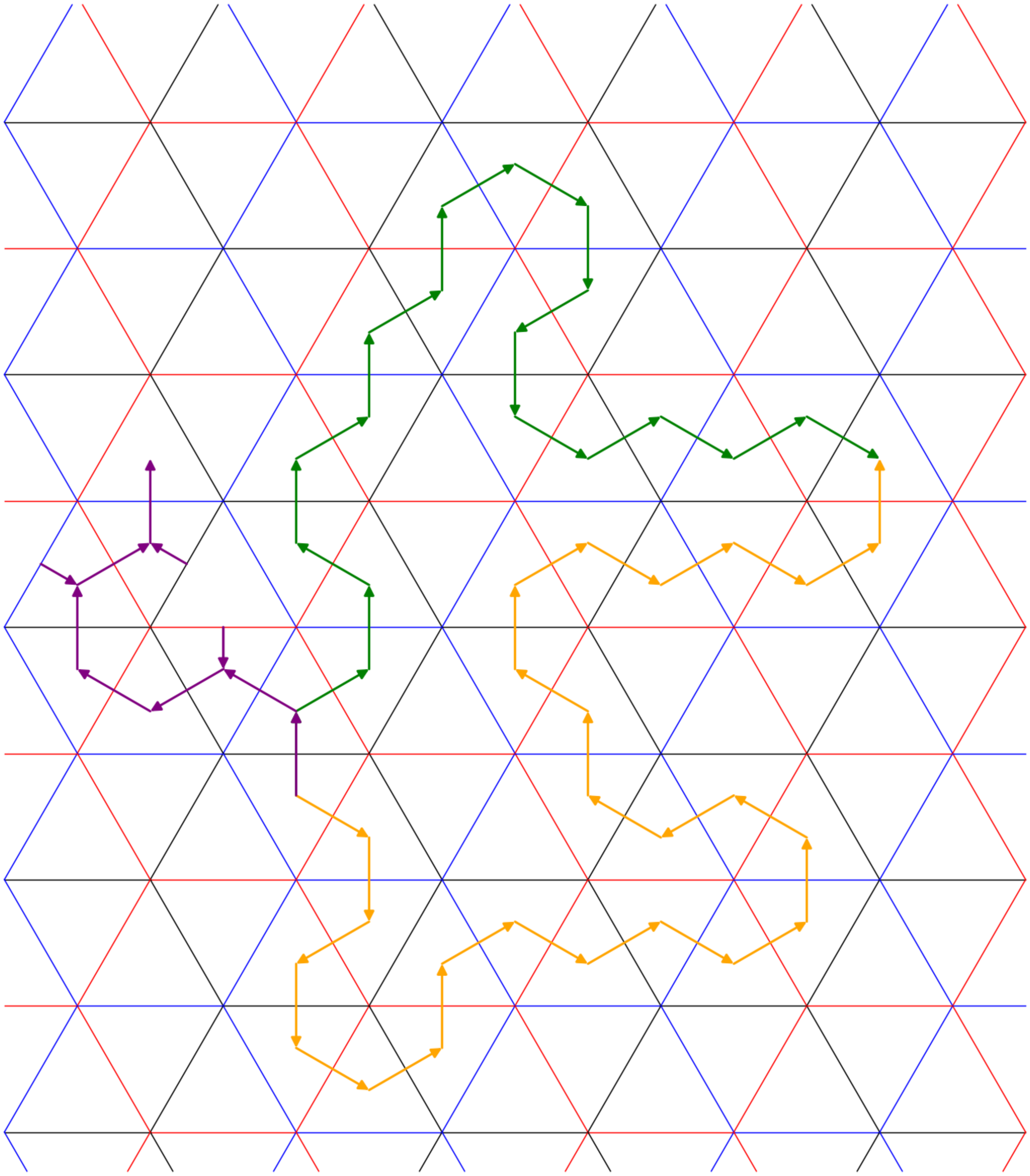
In order to specify common information for all the pieces of a root system plot (choice of projection, bounding box, color code for the index set, ...), the easiest is to create an option object using :meth:`plot_parse_options`, and pass it down to each piece. We use this to plot our two walks:

```
plot_options = L.plot_parse_options(bounding_box=[[-2,5],[-2,6]])
w2 = [2,1,2,0,2,0,2,1,2,0,1,2,1,2,1,0,1,2,0,2,0,1,2,0,2]
p = L.plot_alcoves(plot_options=plot_options)
p += L.plot_alcove_walk(w1, color="green", plot_options=plot_options)
p += L.plot_alcove_walk(w2, color="orange", plot_options=plot_options)
p
```



And another with some foldings:

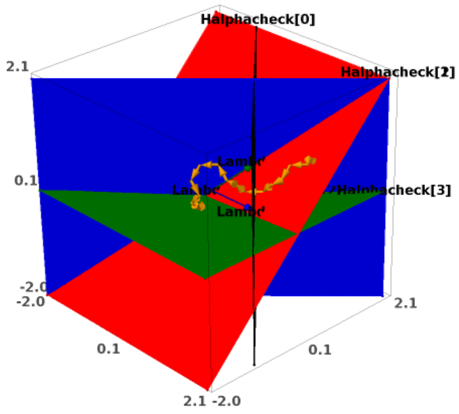
```
p += L.plot_alcove_walk([0,1,2,0,2,0,1,2,0,1],
                          foldings= [False, False, True, False, False, False, True, False, True, False],
                          color="purple")
p.axes(False)
p.show(figsize=20)
```



And finaly a rank 4 alcove walk:

```
L = RootSystem(["B",3,1]).ambient_space()
w3 = [0,2,1,3,2,0,2,1,0,2,3,1,2,1,3,2,0,2,0,1,2,0]
L.plot_fundamental_weights() + L.plot_reflection_hyperplanes(bounding_box=2)+L.plot_alcove_walk(w3)
```

Sleeping... Make Interactive



Paragraph Font Family Font Size

**Hand drawing on top of a root system plot (aka Coxeter graph paper)**

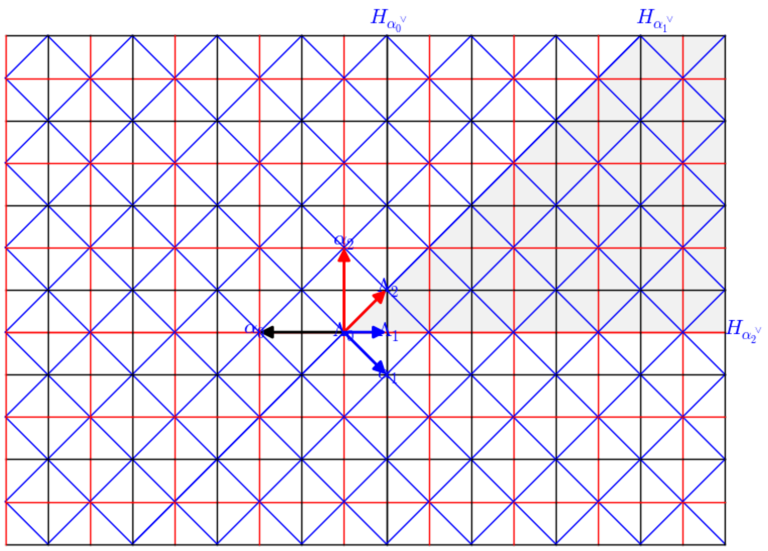
Taken from John Stembridge's excellent data archive:

"If you've ever worked with affine reflection groups, you've probably wasted lots of time drawing the reflecting hyperplanes of the rank 2 groups on scraps of paper. You may also have wished you had pads of graph paper with these lines drawn in for you. If so, you've come to the right place. Behold! Coxeter graph paper!".

Path: p » a

Save changes Cancel changes

```
L = RootSystem(["C",2,1]).ambient_space()
p = L.plot(bounding_box=[[-8,9],[-5,7]])     # long time (10 s)
p
```



By default Sage's plot are bitmap pictures whic would come out ugly if printed on paper. Instead, we recommend saving the picture in postscript or svg before printing it:

```
p.save("C21paper.eps")        # not tested
```

C21paper.eps

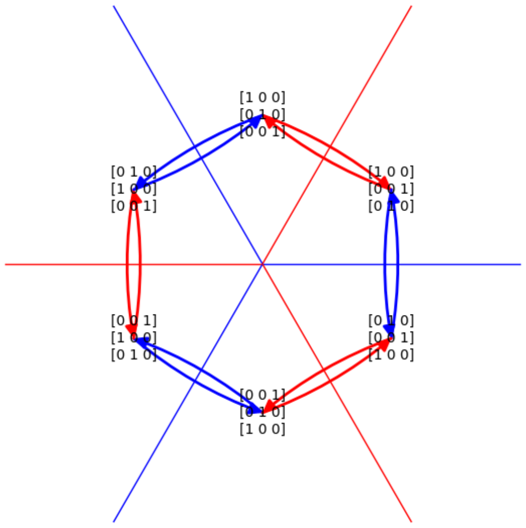
Note

Drawing pictures with a large number of alcoves is currently somewhat ridiculously slow. This is due to the use of generic code that works uniformly in all dimension rather than taylor-made code for 2D. Things should improve with the upcoming fast interface to the PPL library (see e.g. :trac:`12553`).

## Drawing custom objects on top of a root system plot

So far so good. Now, what if one wants to draw, on top of a root system plot, some object for which there is no preexisting plot method? Again, the `plot_options` object come in handy, as it can be used to compute appropriate coordinates. Here we draw the permutohedron, that is the Cayley graph of the symmetric group $W$, by positioning each element $w$ at $w(rho)$, where $rho$ is in the fundamental alcove:

```
L = RootSystem(["A",2]).ambient_space()
rho = L.rho()
plot_options = L.plot_parse_options()
W = L.weyl_group()
g = W.cayley_graph(side="right")
positions = {w: plot_options.projection(w.action(rho)) for w in W}
p = L.plot_alcoves()
p += g.plot(pos = positions, vertex_size=0,
              color_by_label=plot_options.color)
p.axes(False)
p
```

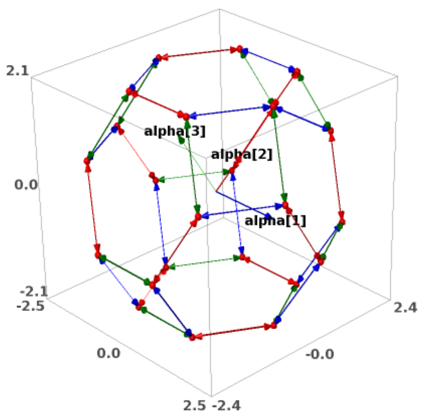


.. TODO:: Could we have nice LaTeX labels in this graph?

The same picture for $A_3$ gives a nice 3D permutohedron:

```
L = RootSystem(["A",3]).ambient_space()
rho = L.rho()
plot_options = L.plot_parse_options()
W = L.weyl_group()
g = W.cayley_graph(side="right")
positions = {w: plot_options.projection(w.action(rho)) for w in W}
p = L.plot_roots()
p += g.plot3d(pos3d = positions, color_by_label=plot_options.color)
p
```

Sleeping... Make Interactive



Exercises

1. Locate the identity element of $W$ in the previous picture
2. Rotate the picture appropriately to highlight the various symmetries of the permutohedron.
3. Make a function out of the previous example, and explore the Cayley graphs of all rank 2 and 3 Weyl groups.
4. Draw the root poset for type B_2 and B_3
5. Draw the root poset for type E_8 to recover the picture from :wikipedia:`File:E8_3D.png`

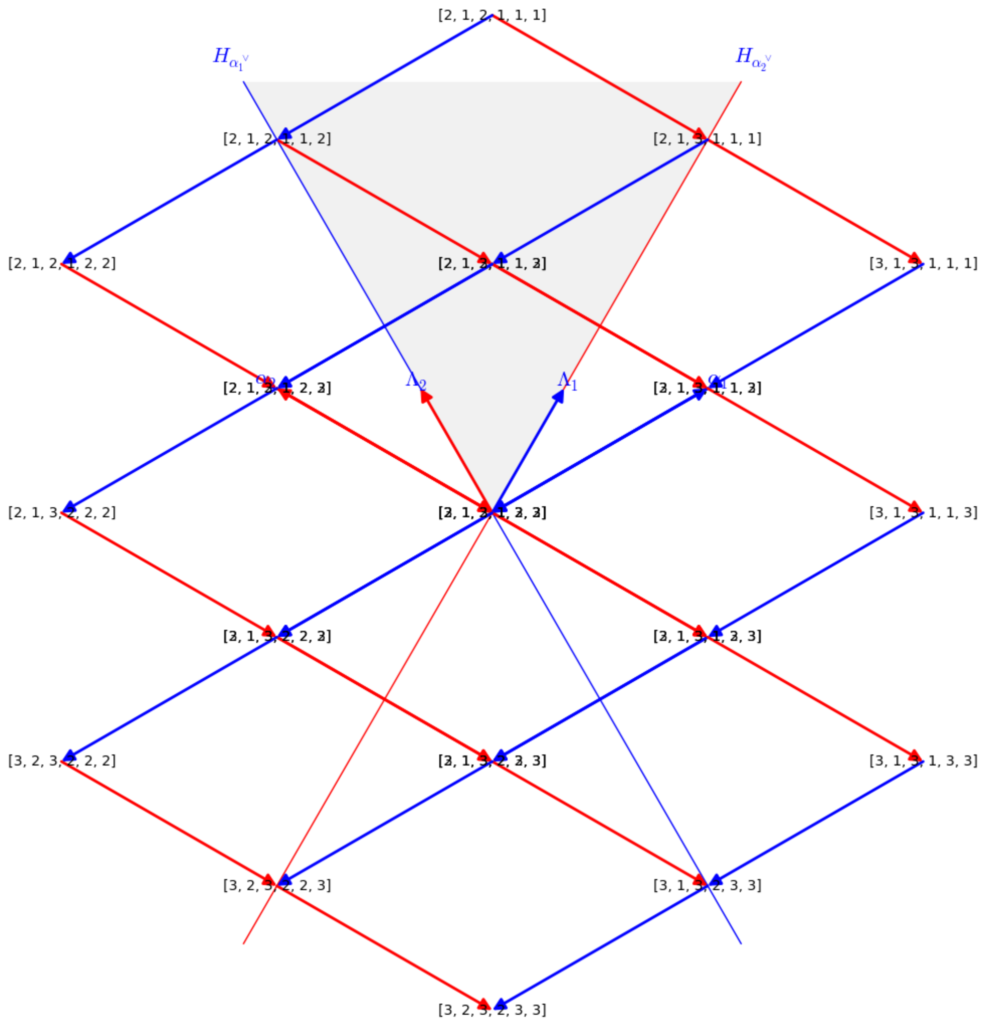
System Message: ERROR/3 (<string>, line 373); *backlink*

Unknown interpreted text role "wikipedia".

Similarly, we display a crystal graph by positioning each element according to its weight:

```
C = CrystalOfTableaux(["A",2], shape=[4,2])
L = C.weight_lattice_realization()
plot_options = L.plot_parse_options()
```

```
g = C.digraph()
positions = {x: plot_options.projection(x.weight()) for x in C}
p = L.plot()
p += g.plot(pos = positions,
              color_by_label=plot_options.color, vertex_size=0)
p.axes(False)
p.show(figsize=15)
```



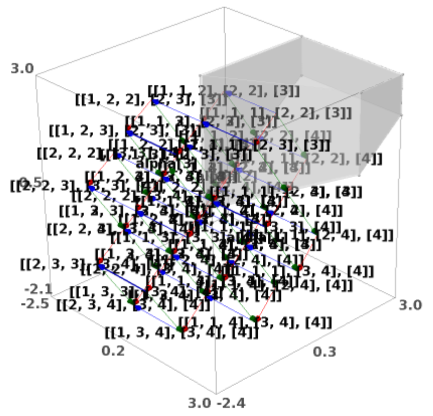
Note

In the above picture, many pairs of tableaux have the same weight and are thus superposed (look for example near the center). Some more layout logic would be needed to separate those nodes properly, but the foundations are laid firmly and uniformly accross all types of root systems for writing such extensions.

Here is an analogue picture in 3D:

```
C = CrystalOfTableaux(["A",3], shape=[3,2,1])
L = C.weight_lattice_realization()
plot_options = L.plot_parse_options()
g = C.digraph()
positions = {x:plot_options.projection(x.weight()) for x in C}
p = L.plot(reflection_hyperplanes=False, fundamental_weights=False)
p += g.plot3d(pos3d = positions, vertex_labels=True,
                color_by_label=plot_options.color, edge_labels=True)
p
```

Sleeping... Make Interactive



Exercise

Explore the previous picture and notice how the edges of the crystal graph are parallel to the simple roots.

Enjoy and please post your best pictures on the Sage-Combinat wiki.