

In [1]:

```
##### This is some SAGEmath code used during the preparation of the paper
##### "A brute force computer aided proof of an existence result about ex
tremal hyperbolic surfaces"
##### by Ernesto Girondo and Cristian Reyes

##### THE CASE N=7
```

In [2]:

```
##### CELL 1: SOME FUNCTIONS
```

```
##### FUNCTION poli2pi3. Edges, vertices and midpoints of a regular polyg
on of n edges
def poli2pi3(n):
    u'''This is a function of an integer n. The output is a triplet consisting
of the n edges, the n vertices
    and the n edge midpoints of a regular hyperbolic polygon of angle 2Pi/3 ce
ntered at the origin. Points and vertices
    are listed in counterclockwise order. The first midpoint, which lies in th
e negative imaginary axis,
    corresponds to the last edge.
    '''
    H=N(arccosh(1/(tan((pi)/n)*tan((pi)/3))))
    h=N(tanh(H/2))
    R=N(arccosh((cos((pi)/3))/(sin((pi/n)))))
    r=N(tanh(R/2))
    puntosmedios=[N(r*(cos(3*(pi)/2+2*k*pi/n))+I*r*(sin(3*(pi)/2+2*k*pi/n))) f
or k in [0..n-1]
    vertices=[N(h*(cos(3*(pi)/2+(1+2*k)*pi/n))+h*(sin(3*(pi)/2+(1+2*k)*pi/n))*
I) for k in [0..n]]
    lados = [PD.get_geodesic(vertices[k], vertices[k+1]) for k in [0..n-1]]
    return lados, vertices, puntosmedios
####
```

```
##### FUNCTION moverpol returns a plot of the image of the list of edges "la
dos", which is a global variable
def moverpol(x, col):
    u'''This is a function of a transformation x and a color col. The output i
s a plot in color col of the image of the
    set "lados", which is a global variable.
    '''
    global lados
    bpol=Graphics()
    for j in [0..M-1]:
        movido=x*lados[j]
        bpol+=movido.show(color=col)
    return bpol
####
```

In [3]:

```
##### CELL2: SOME CODE DEFINING THE GENERATORS OF A TRIANGLE GROUP (2,3,M)
) and related things
```

```
PD=HyperbolicPlane().PD() #The Disc model of the hyperbolic plane
M=7 #The number of edges of the polygons. This is called N in the paper
lados, vertices, puntosmedios = poli2pi3(M) #

#Now computing the points A,B,C and the three generators a,b,c of the extended
triangle group (2,3,M)
H=N(arccosh(1/(tan((pi)/M)*tan((pi)/3))))
h=N(tanh(H/2))
R=N(arccosh((cos((pi)/3))/(sin((pi)/M))))
r=N(tanh(R/2))
LM=N(arccosh((cos((pi)/M))/(sin((pi)/3))))
L=2*LM
l=N(tanh(L/2))

B=PD.get_point(0+0*I)
C=PD.get_point(N(r*cos((pi)*3/2)+r*sin((pi)*3/2)*I))
A=PD.get_point(N(h*cos((pi)/M+3*(pi)/2)+h*sin((pi)/M+3*(pi)/2)*I))
lado_c, lado_b, lado_a =PD.get_geodesic(A,B),PD.get_geodesic(A,C), PD.get_geod
esic(B,C)
a,b,c=lado_a.reflection_involution(), lado_b.reflection_involution(), lado_c.r
eflection_involution()
```

In [4]:

```
##### CELL3: OTHER USEFUL FUNCTIONS
```

```
def Rota(t):
    return (c*a)^t*a*b

def rotacionpi(p,q): #THE ORDER 2 ELLIPTIC ELEMENT PERMUTING TWO GIVEN POINTS
p AND q
    ek1=PD.get_point(p)
    ek2=PD.get_point(q)
    geod=PD.get_geodesic(ek1,ek2)
    ek3=geod.midpoint().coordinates()
    ma1=matrix([[I,(ek3)*(-I)],[((ek3.conjugate())*(I),-I)]])
    Rot=PD.get_isometry(ma1)
    rotpii=matrix([[I,0],[0,-I]])
    rotpi=PD.get_isometry(rotpii)
    fuu=Rot*rotpi*Rot^-1
    return fuu

def G0(i):
    return a*b*(c*a)^i*a*b
def G1(i):
    return (c*a)*a*b*(c*a)^i*a*b*(c*a)^-1
def G2(i):
    return (c*a)^2*a*b*(c*a)^i*a*b*(c*a)^-2
def G3(i):
    return (c*a)^3*a*b*(c*a)^i*a*b*(c*a)^-3
def G4(i):
    return (c*a)^4*a*b*(c*a)^i*a*b*(c*a)^-4
def G5(i):
    return (c*a)^5*a*b*(c*a)^i*a*b*(c*a)^-5
```

In [5]:

```
##### CELL4: plotdom IS THE FUNDAMENTAL DOMAIN F DESCRIBED IN THE TEXT AND sidepairings ARE THE SET
##### OF SIDE PAIRING TRANSFORMATIONS GENERATING THE GROUP K

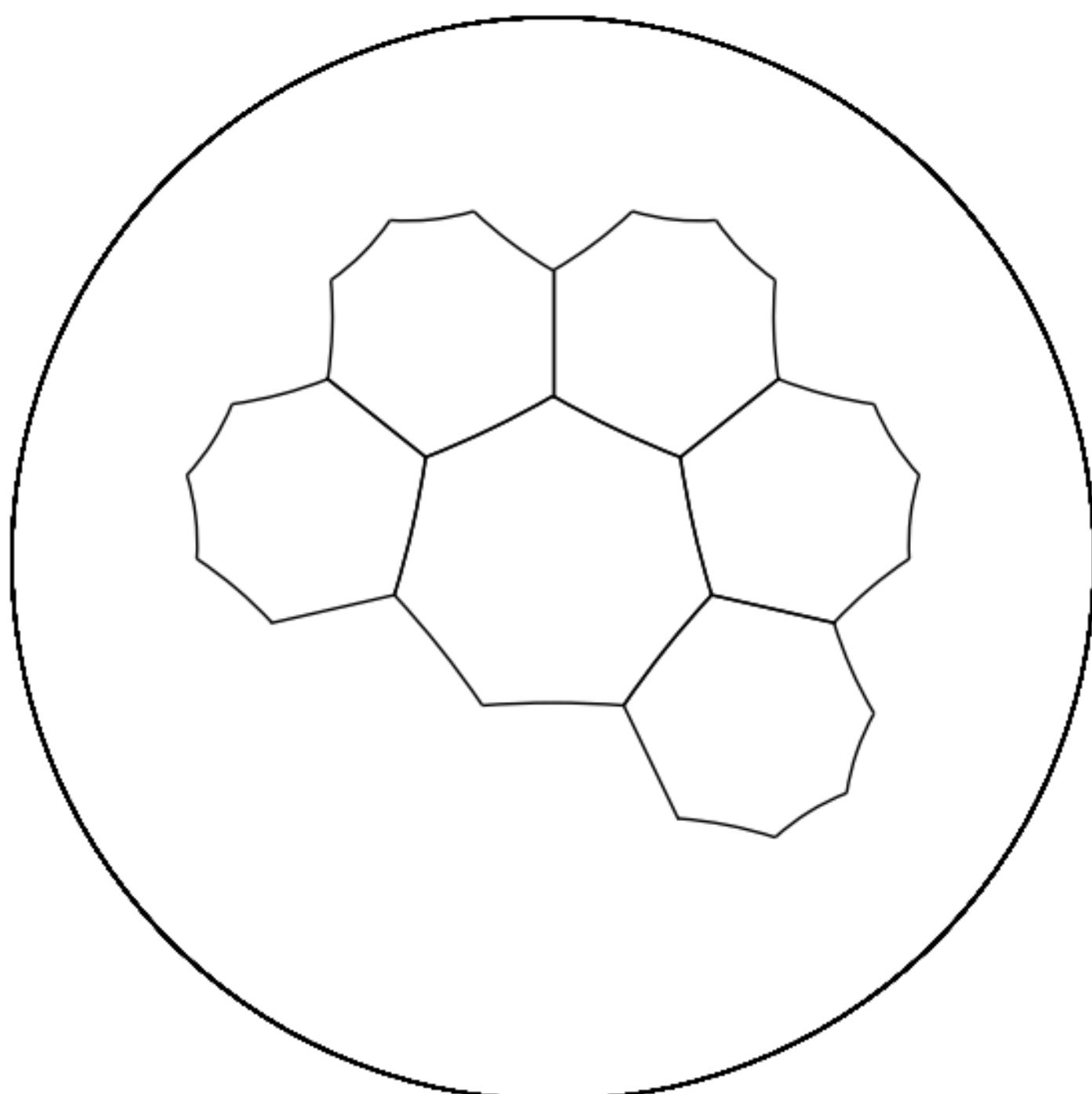
plotdom=moverpol(a*a^-1,'black')
for k in [1..5]:
    plotdom=plotdom+moverpol((c*a)^k*a*b*(c*a)^-k,'black')

show(plotdom)

sidepairings=['comodin']
sidepairingscode=['cero']

sidepairings.append(G1(4)*(c*a)*b*(c*a)^-1*G1(4)) #1 Side pairing (-, 3_3, 3_4)
sidepairingscode.append('(-, 3_3, 3_4)')
sidepairings.append(G1(5)*(c*a)*a*b*(c*a)^-1*G1(5)) #2 Side pairing (+, 3_2, 3_5)
sidepairingscode.append('(+, 3_2, 3_5)')
sidepairings.append(G2(2)*(c*a)^2*a*b*(c*a)^-1*G1(-1)) #3 Side pairing (+, 3_1, 4_2)
sidepairingscode.append('(+, 3_1, 4_2)')
sidepairings.append(G2(3)*(c*a)^2) #4 Side pairing (+, 1_0, 4_3)
sidepairingscode.append('(+, 1_0, 4_3)')
sidepairings.append(G3(-2)*(c*a)^3*b*(c*a)^-2*G2(3)) #5 Side pairing (-, 4_4, 5_5)
sidepairingscode.append('(-, 4_4, 5_5)')
sidepairings.append(G4(2)*(c*a)^4*b*(c*a)^-2*G2(2)) #6 Side pairing (-, 4_5, 6_2)
sidepairingscode.append('(-, 4_5, 6_2)')
sidepairings.append(G4(3)*(c*a)^4*b*(c*a)^-3*G3(-2)) #7 Side pairing (-, 5_2, 6_3)
sidepairingscode.append('(-, 5_2, 6_3)')
sidepairings.append(G5(-1)*(c*a)^5*b*(c*a)^-3*G3(-3)) #8 Side pairing (-, 5_3, 7_6)
sidepairingscode.append('(-, 5_3, 7_6)')
sidepairings.append((c*a)^6*a*(c*a)^-3*G3(3)) #9 Side pairing (-, 5_4, 1_6)
sidepairingscode.append('(-, 5_4, 1_6)')
sidepairings.append(G5(4)*G4(2)) #10 Side pairing (+, 6_4, 7_5)
sidepairingscode.append('(+, 6_4, 7_5)')
sidepairings.append(G5(2)*G4(-1)*(c*a)^4*a*(c*a)^-4*G4(2)) #11 Side pairing (-, 6_5, 7_3)
sidepairingscode.append('(-, 6_5, 7_3)')
sidepairings.append(G5(4)*(c*a)^5*b*(c*a)^-5*G5(-2)) #12 Side pairing (-, 7_2, 7_4)
sidepairingscode.append('(-, 7_2, 7_4)')
```

/Applications/SageMath-8.2.app/Contents/Resources/sage/local/lib/python2.7/site-packages/sage/repl/ipython\_kernel/\_\_main\_\_.py:31: DeprecationWarning: show is deprecated. Please use plot instead.  
See <http://trac.sagemath.org/20530> for details.

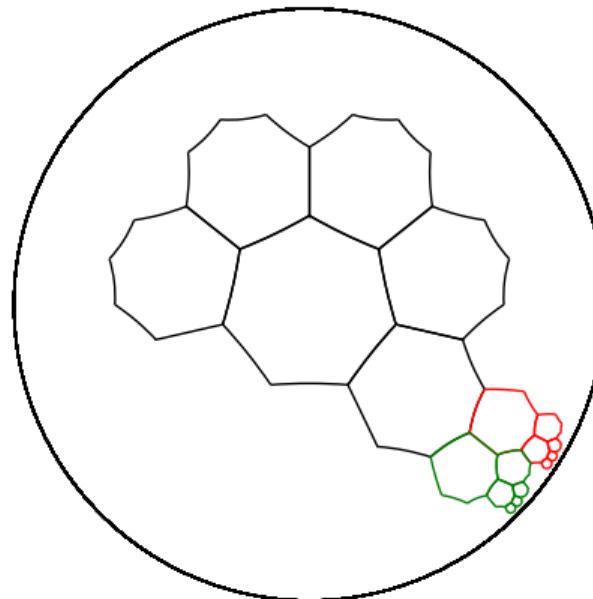


In [6]:

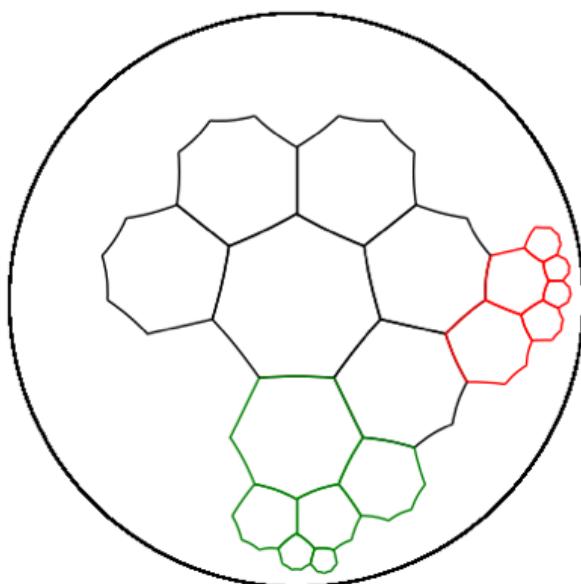
```
##### CELL 5: USE IT IF YOU WANT TO CHECK GRAPHICALLY HOW OUR SIDE PAIRINGS ACT.  
##### THE IMAGE OF plotdom (THE FUNDAMENTAL DOMAIN F) BY THE SIDE PAIRING IS DEPICTED IN RED.  
##### THE IMAGE BY THE INVERSE IS DEPICTED IN GREEN.  
##### THE SIDE PAIRING MAPS THE GREEN EDGE(S) OF plotdom TO THE RED ONE(S).  
##### THE CONFORMALITY/ANTICONFORMALITY OF THE SIDE PAIRING IS DETERMINED BY THE PARITY OF ITS LENGTH AS A WORD IN a,b,c
```

```
total=plotdom;  
for j in [1..len(sidepairings)-1]:  
  
    pairing=sidepairings[j]  
    resultado=plotdom+moverpol(pairing,'red')  
    total=total+moverpol(pairing,'black')  
    for k in [1..5]:  
        resultado=resultado+moverpol(pairing*(c*a)^k*a*b*(c*a)^-k,'red')  
        total=total+moverpol(pairing*(c*a)^k*a*b*(c*a)^-k,'black')  
  
    pairing_inv=sidepairings[j]^(-1)  
    resultado=resultado+moverpol(pairing_inv,'green')  
    total=total+moverpol(pairing_inv,'black')  
    for k in [1..5]:  
        resultado=resultado+moverpol(pairing_inv*(c*a)^k*a*b*(c*a)^-k,'green')  
        total=total+moverpol(pairing_inv*(c*a)^k*a*b*(c*a)^-k,'black')  
    show(resultado, title='Side pairing '+sidepairingscode[j]+' maps the black domain to the red one, its inverse maps the black domain to the green one', title_pos=(0.35,-0.1))  
  
show(total, title='The tesselation induced by the group generated by these side pairings', title_pos=(0.4,-0.05))
```

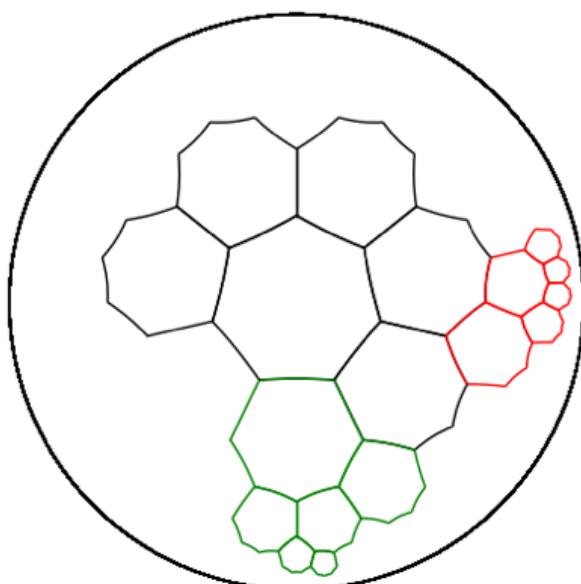
```
/Applications/SageMath-8.2.app/Contents/Resources/sage/local/lib/python2.7/site-packages/sage/repl/ipython_kernel/__main__.py:31: DeprecationWarning: show is deprecated. Please use plot instead.  
See http://trac.sagemath.org/20530 for details.
```



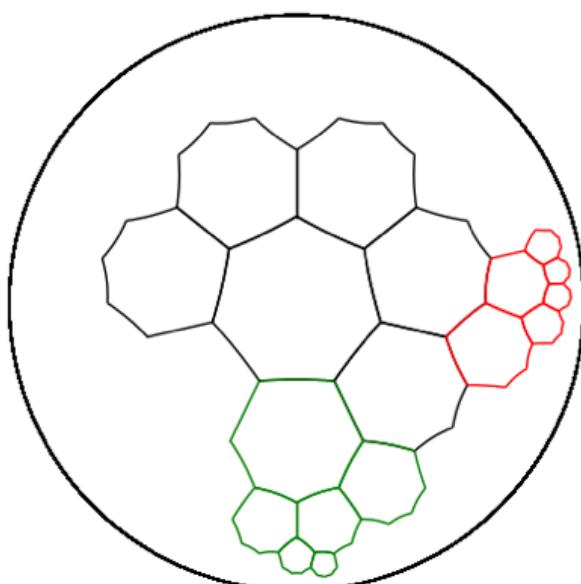
Side pairing (-, 3\_3, 3\_4) maps the black domain to the red one, its inverse maps the black domain to the green one



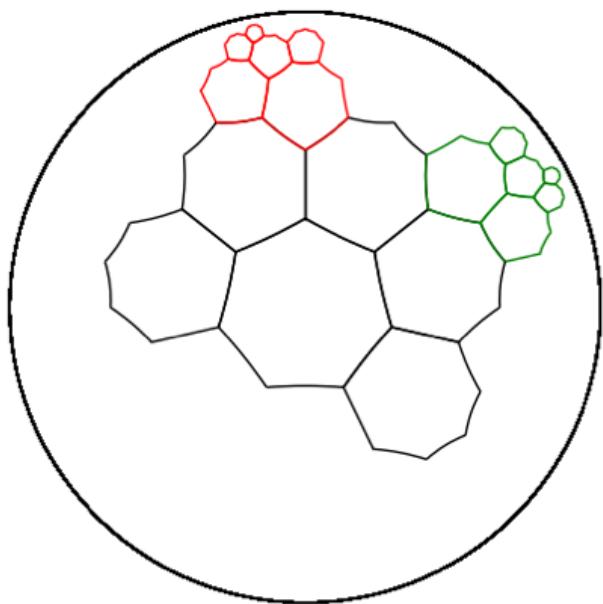
Side pairing  $(+, 3_2, 3_5)$  maps the black domain to the red one, its inverse maps the black domain to the green one



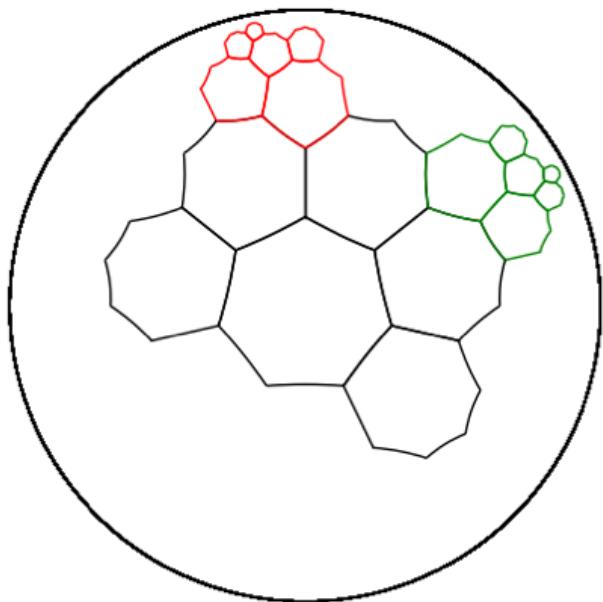
Side pairing  $(+, 3_1, 4_2)$  maps the black domain to the red one, its inverse maps the black domain to the green one



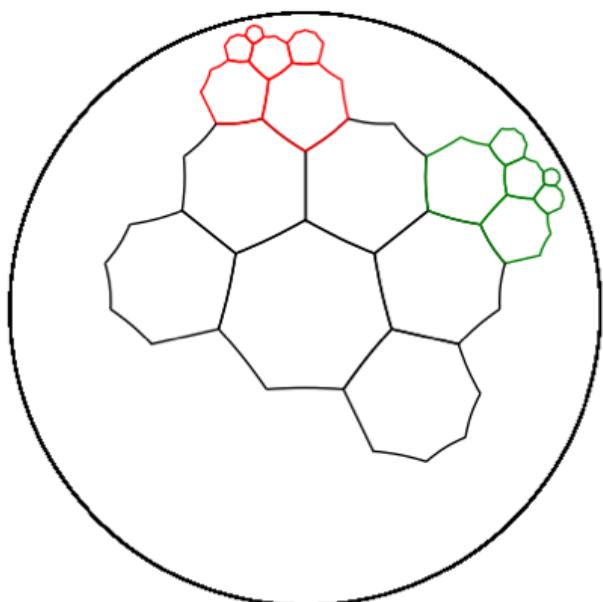
Side pairing  $(+, 1_0, 4_3)$  maps the black domain to the red one, its inverse maps the black domain to the green one



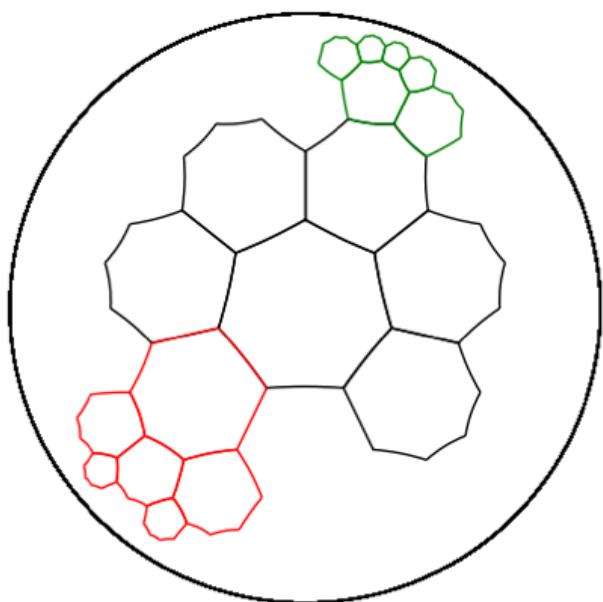
Side pairing (-, 4\_4, 5\_5) maps the black domain to the red one, its inverse maps the black domain to the green one



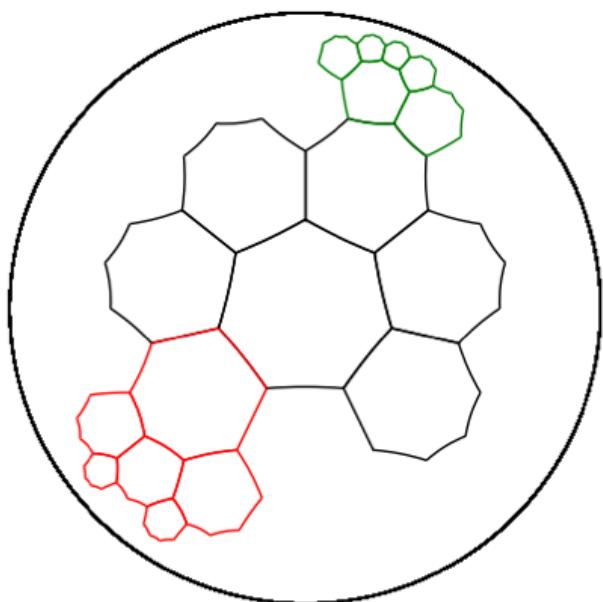
Side pairing (-, 4\_5, 6\_2) maps the black domain to the red one, its inverse maps the black domain to the green one



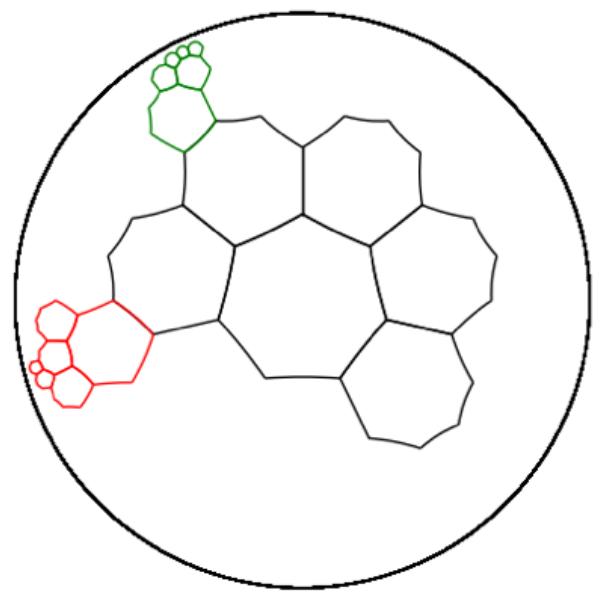
Side pairing (-, 5\_2, 6\_3) maps the black domain to the red one, its inverse maps the black domain to the green one



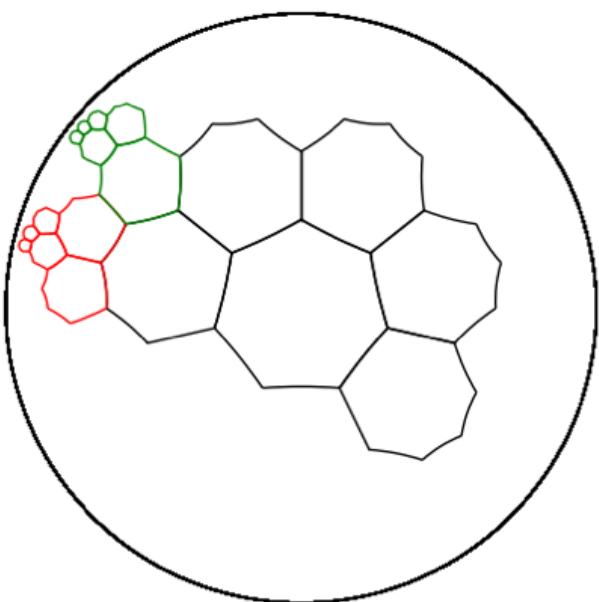
Side pairing  $(-, 5_3, 7_6)$  maps the black domain to the red one, its inverse maps the black domain to the green one



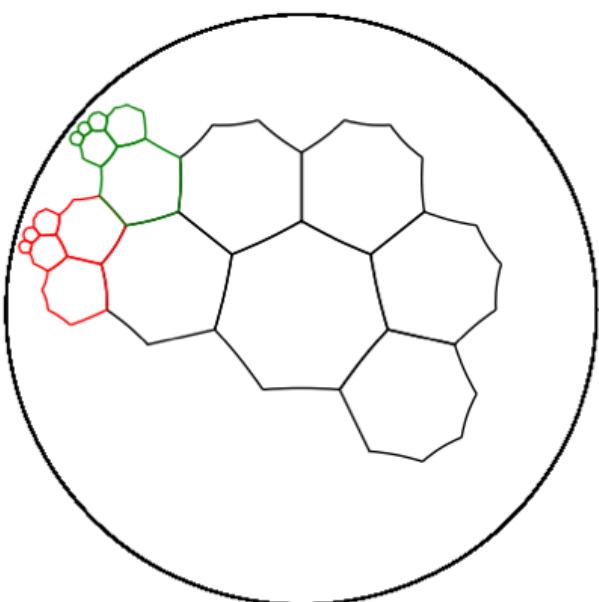
Side pairing  $(-, 5_4, 1_6)$  maps the black domain to the red one, its inverse maps the black domain to the green one



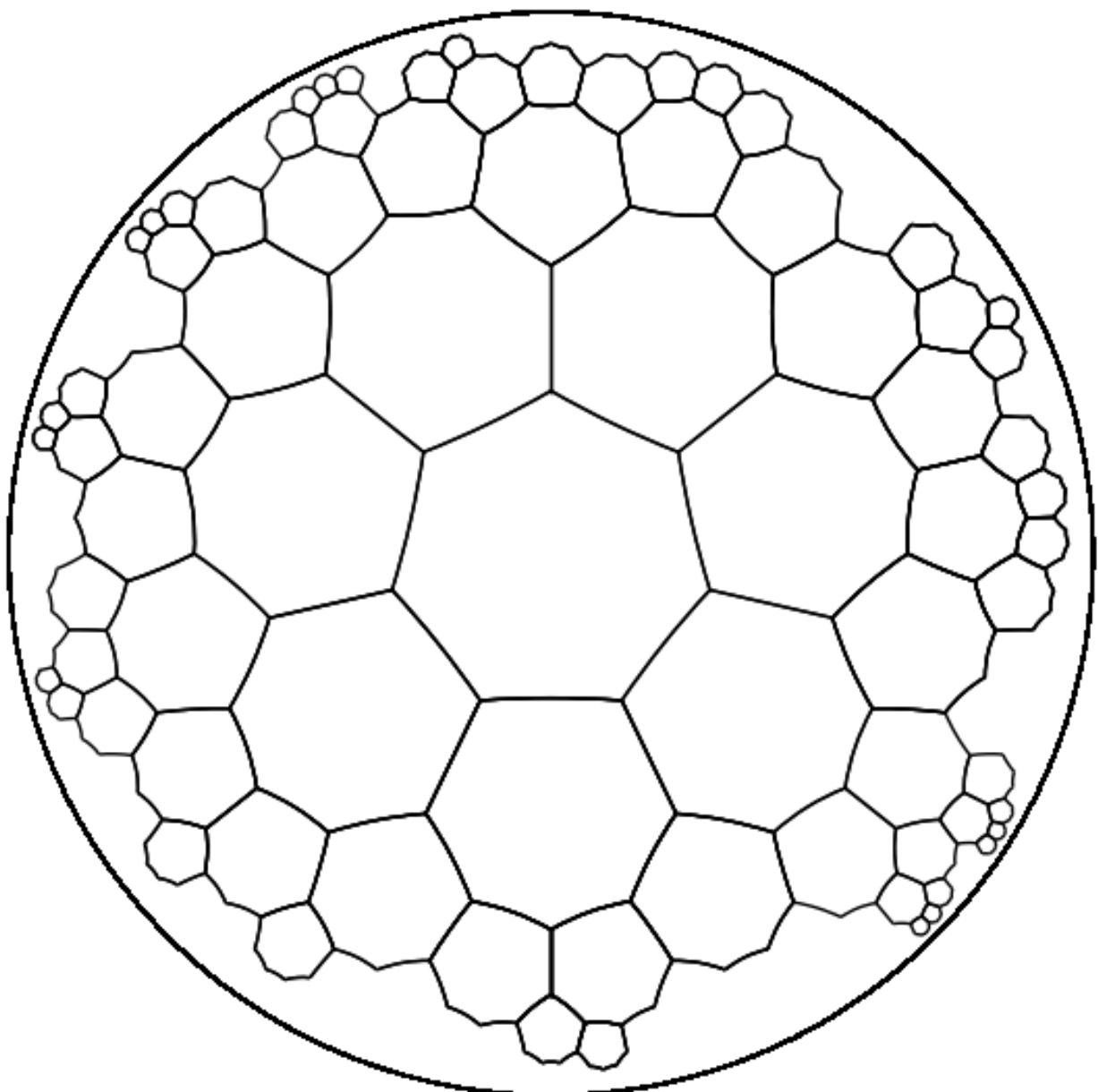
Side pairing  $(+, 6_4, 7_5)$  maps the black domain to the red one, its inverse maps the black domain to the green one



Side pairing  $(-, 6_5, 7_3)$  maps the black domain to the red one, its inverse maps the black domain to the green one



Side pairing  $(-, 7_2, 7_4)$  maps the black domain to the red one, its inverse maps the black domain to the green one



The tessellation induced by the group generated by these side pairings

In [7]:

```
##### CELL 6: DEFINING AN ELLIPTIC ELEMENT tau OF ORDER TWO THAT NORMALIZES THE GROUP K
##### candidate IS A POSSIBLE CENTER OF ONE OF THE DISCS OF A HIDDEN EXTR EMAL PACKING, OBTAINED BY THE BRUTE FORCE PROCEDURE
##### conjugates IS A LIST OF CERTAIN ELEMENTS OF THE GROUP K
##### WE SHOW HERE NUMERICALLY THAT conjugates[j]*tau*sidepairing[j]*tau^(-1) IS THE IDENTITY

candidate=0.0353613040941117 + 0.403413208727238*I
tau=rotasionpi(((c*a)^3*b*B).coordinates(),candidate)

conjugates=['cero',
sidepairings[9]^(-1)*sidepairings[1]*sidepairings[9],
(sidepairings[6]*sidepairings[9])^(-1),
(sidepairings[6]*sidepairings[9])^(-1),
(sidepairings[6]*sidepairings[9])^(-1),
sidepairings[6],
sidepairings[6],
sidepairings[6],
sidepairings[9],
sidepairings[9],
sidepairings[7]^(-1)*sidepairings[9],
sidepairings[7]^(-1)*sidepairings[11]^(-1)*sidepairings[9],
sidepairings[7]^(-1)*sidepairings[11]^(-1)*sidepairings[9]]

for j in [1..len(conjugates)-1]:
    print(conjugates[j]*tau*sidepairings[j]*tau^(-1))
```

```
Isometry in PD
[-1.00000000000000 - 3.96627175547337e-13*I -3.9196423884391
2e-13 + 5.07371922253697e-14*I]
[-3.91964238843912e-13 - 5.07371922253697e-14*I      -1.000000000
0000 + 3.96627175547337e-13*I]

Isometry in PD
[-1.00000000000001 + 1.44759204623313e-13*I 1.39568911983190e
-13 - 4.26880752968373e-14*I]
[1.39568911983190e-13 + 4.26880752968373e-14*I      -1.00000000000
001 - 1.44759204623313e-13*I]

Isometry in PD
[1.00000000000000 - 1.47701295638569e-13*I -1.4284406990583
4e-13 + 4.06896738525120e-14*I]
[-1.42844069905834e-13 - 4.06896738525120e-14*I      1.000000000
0000 + 1.47701295638569e-13*I]

Isometry in PD
[1.00000000000000 - 1.38736244714721e-13*I -1.3415657473814
2e-13 + 3.91353616180368e-14*I]
[-1.34156574738142e-13 - 3.91353616180368e-14*I      1.000000000
0000 + 1.38736244714721e-13*I]

Isometry in PD
[0.999999999999998 - 4.69069227904129e-15*I -6.5225602696727
9e-15 - 3.77475828372553e-15*I]
[-6.52256026967279e-15 + 3.77475828372553e-15*I      0.9999999999
9998 + 4.69069227904129e-15*I]

Isometry in PD
```

```

[ -0.999999999999999 + 1.38083988687754e-14*I 1.47243328640911e
-14 + 6.88338275267597e-15*I]
[1.47243328640911e-14 - 6.88338275267597e-15*I      -0.9999999999999
999 - 1.38083988687754e-14*I]
Isometry in PD
[ 1.00000000000000 - 2.65898414397725e-14*I -2.6811886044697
5e-14 - 7.82707232360735e-15*I]
[-2.68118860446975e-14 + 7.82707232360735e-15*I      1.000000000
0000 + 2.65898414397725e-14*I]
Isometry in PD
[ 1.00000000000000 - 2.77555756156289e-16*I -6.1617377866696
2e-15 + 2.55351295663786e-15*I]
[-6.16173778666962e-15 - 2.55351295663786e-15*I      1.000000000
0000 + 2.77555756156289e-16*I]
Isometry in PD
[ -1.00000000000000 - 9.29811783123569e-16*I 4.37150315946155e
-15 - 2.44249065417534e-15*I]
[4.37150315946155e-15 + 2.44249065417534e-15*I      -1.00000000000
000 + 9.29811783123569e-16*I]
Isometry in PD
[ 0.999999999999998 - 3.23630011678233e-14*I -2.4480417692984
7e-14 + 2.04281036531029e-14*I]
[-2.44804176929847e-14 - 2.04281036531029e-14*I      0.999999999
9998 + 3.23630011678233e-14*I]
Isometry in PD
[ 0.999999999999996 - 2.90600876695635e-14*I -2.4896751327219
1e-14 + 1.62647673107585e-14*I]
[-2.48967513272191e-14 - 1.62647673107585e-14*I      0.999999999
9996 + 2.90600876695635e-14*I]
Isometry in PD
[ -0.999999999999996 + 4.62269111878300e-14*I 3.75949271713694e
-14 - 2.65343302885412e-14*I]
[3.75949271713694e-14 + 2.65343302885412e-14*I      -0.9999999999999
996 - 4.62269111878300e-14*I]

```

In [8]:

```

#####
Cell 7: the fixed point of tau
tau.fixed_point_set()

```

Out[8]:

```
[Point in PD 0.128823227467489 + 0.422224421195793*I]
```