

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=8
```

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=8 #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 rotasionpi(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

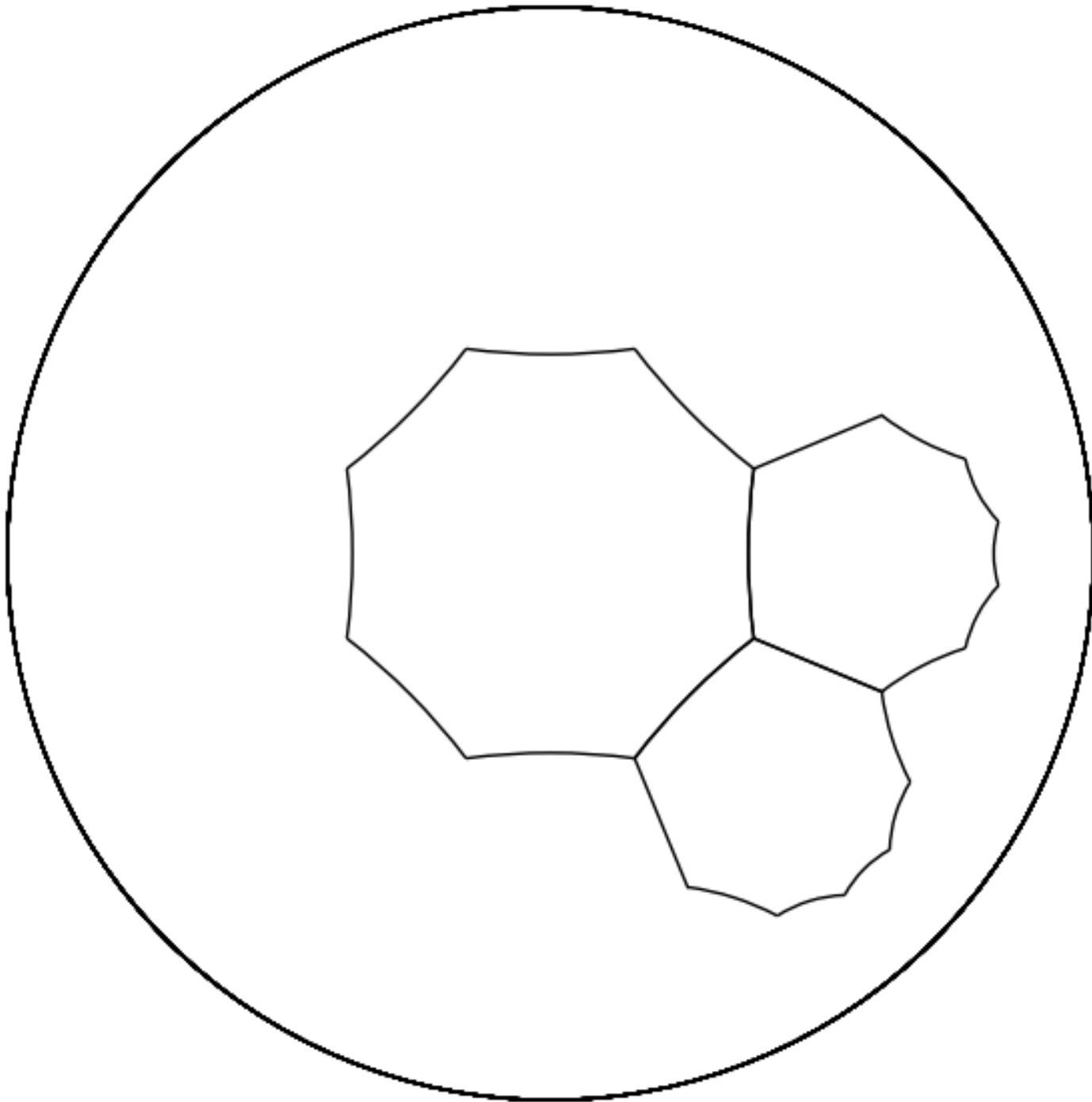
plotdom=moverpol(a*a^-1,'black')
for k in [1..2]:
    plotdom=plotdom+moverpol((c*a)^k*a*b*(c*a)^-k,'black')

show(plotdom)

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

sidepairings.append(G1(-2)*(c*a)*b*(c*a)^-1*G1(4))          #1 Side pairi
ng (-, 3_4, 3_6)
sidepairingscode.append('(-, 3_4, 3_6)')
sidepairings.append(G2(2)*G1(4))          #2 Side pairi
ng (+, 3_3, 4_3)
sidepairingscode.append('(+, 3_3, 4_3)')
sidepairings.append((c*a)^2*(c*a)*a*(c*a)^-1*G1(-2))      #3 Side pairi
ng (-, 3_2, 1_3)
sidepairingscode.append('(-, 3_2, 1_3)')
sidepairings.append(G2(1)*G1(-1)*(c*a)*a*(c*a)^-1*G1(3))  #4 Side pairi
ng (-, 3_5, 4_2)
sidepairingscode.append('(-, 3_5, 4_2)')
sidepairings.append(G2(-2)*(c*a)^2*a)          #5 Side pairi
ng (-, 1_0, 4_6)
sidepairingscode.append('(-, 1_0, 4_6)')
sidepairings.append((c*a)^-3*b*(c*a))          #6 Side pairi
ng (-, 1_7, 1_5)
sidepairingscode.append('(-, 1_7, 1_5)')
sidepairings.append(G2(-3)*(c*a)^4)          #7 Side pairi
ng (+, 1_6, 4_5)
sidepairingscode.append('(+, 1_6, 4_5)')
sidepairings.append(G2(-4)*(c*a)^2*a*(c*a)^4)          #8 Side pairi
ng (-, 1_4, 4_2)
sidepairingscode.append('(-, 1_4, 4_2)')
sidepairings.append(G1(1)*(c*a)^-1*(c*a)^2*b*(c*a)^-2*G2(1)) #9 Side pairi
ng (-, 4_7, 3_1)
sidepairingscode.append('(-, 4_7, 3_1)')
```

/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
```

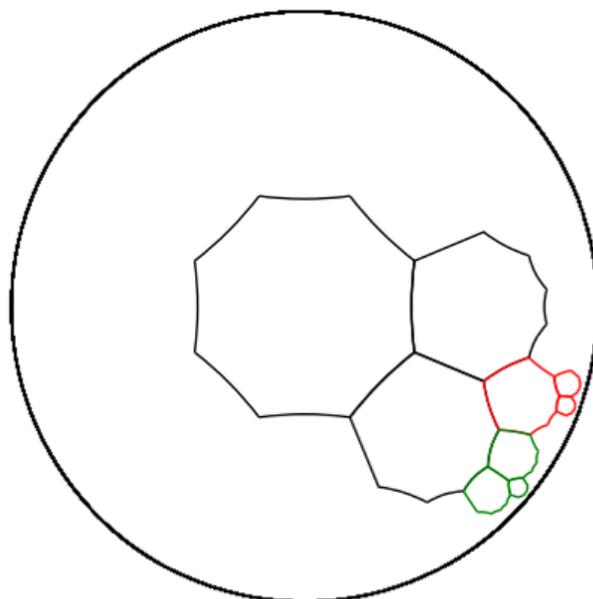
```
total=plotdom;
for j in [1..len(sidepairings)-1]:

    pairing=sidepairings[j]
    resultado=plotdom+moverpol(pairing,'red')
    total=total+moverpol(pairing,'black')
    for k in [1..2]:
        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')

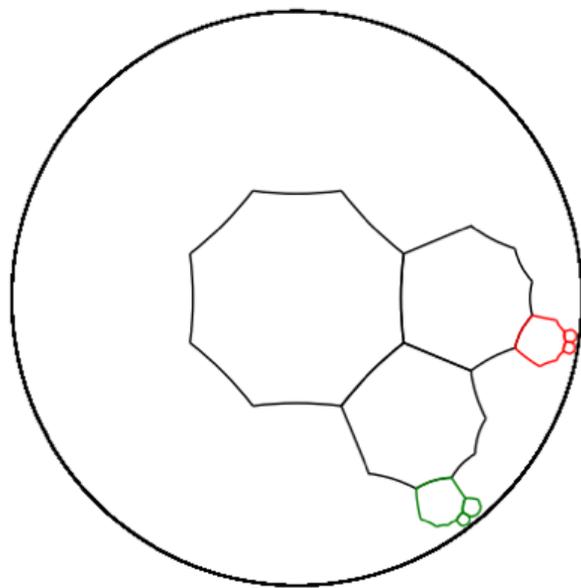
    pairing_inv=sidepairings[j]^(-1)
    resultado=resultado+moverpol(pairing_inv,'green')
    total=total+moverpol(pairing_inv,'black')
    for k in [1..2]:
        resultado=resultado+moverpol(pairing_inv*(c*a)^k*a*b*(c*a)^-k,'green')
        total=total+moverpol(pairing_inv*(c*a)^k*a*b*(c*a)^-k,'black')
    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 tessellation induced by the group generated by these side pairings', title_pos=(0.4,-0.05))
```

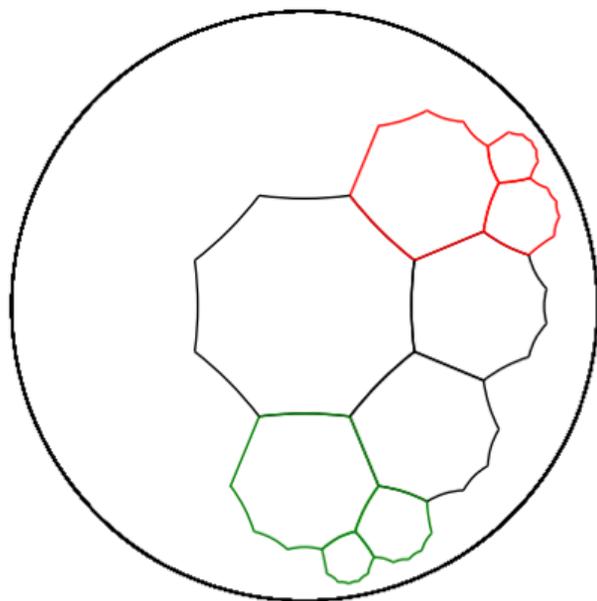
/Applications/SageMath-8.2.app/Contents/Resources/sage/local/lib/python2.7/site-packages/sage/repl/ipython_kernel/__main__.py:31: DeprecationWarning: show is deprecated. Please use plot instead. See <http://trac.sagemath.org/20530> for details.



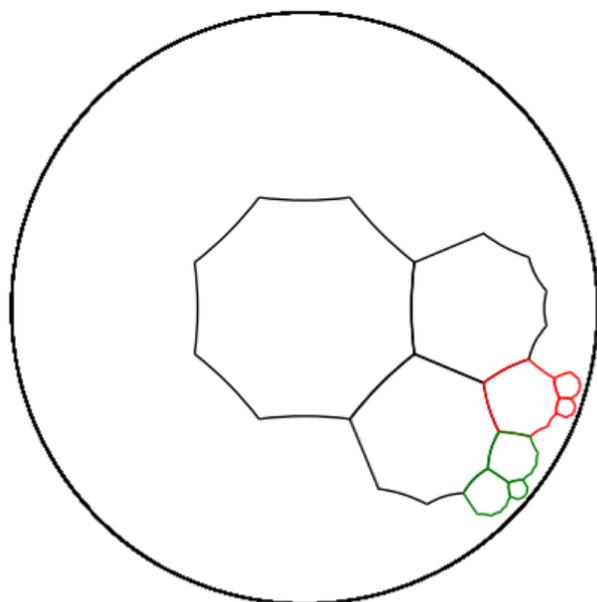
Side pairing $(-, 3_4, 3_6)$ maps the black domain to the red one, its inverse maps the black domain to the green one



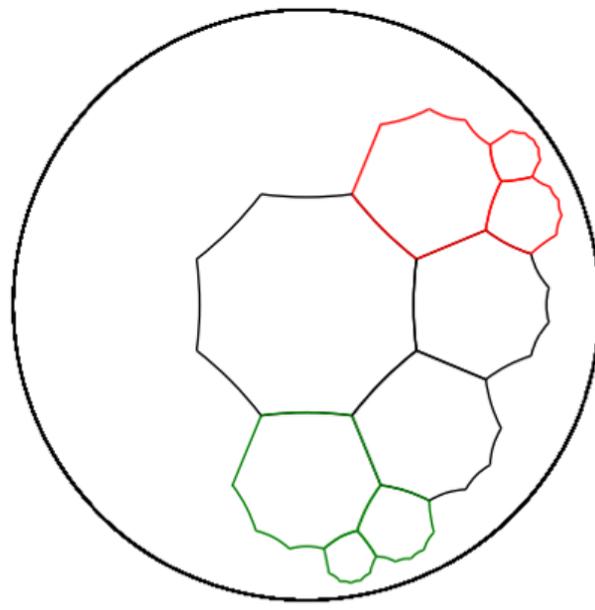
Side pairing $(+, 3_3, 4_3)$ maps the black domain to the red one, its inverse maps the black domain to the green one



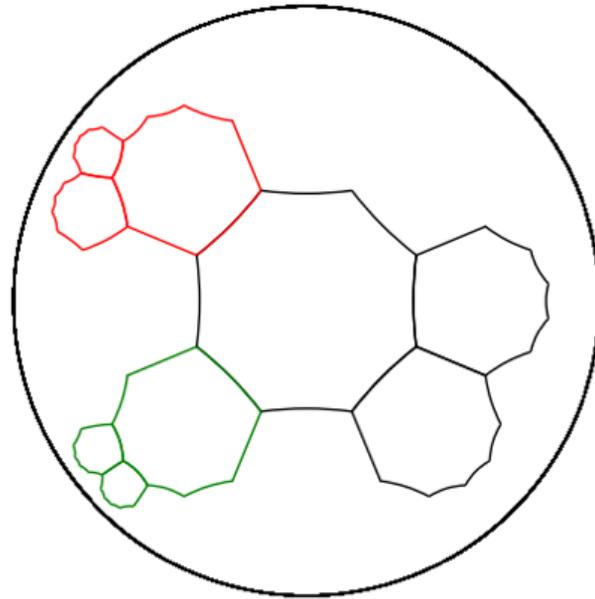
Side pairing $(-, 3_2, 1_3)$ maps the black domain to the red one, its inverse maps the black domain to the green one



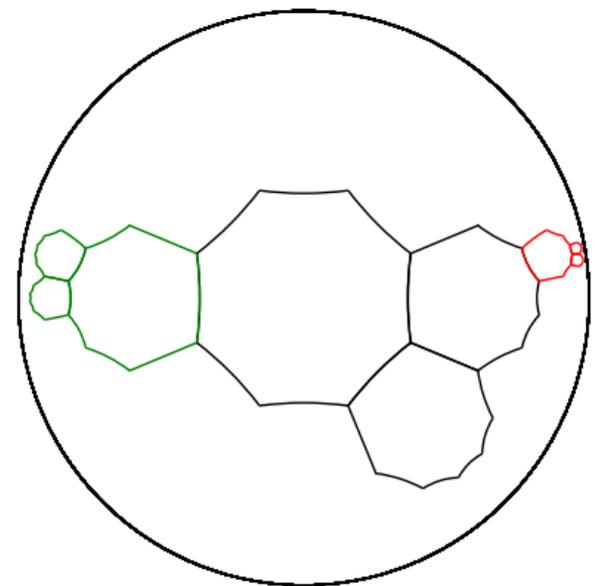
Side pairing $(-, 3_5, 4_2)$ maps the black domain to the red one, its inverse maps the black domain to the green one



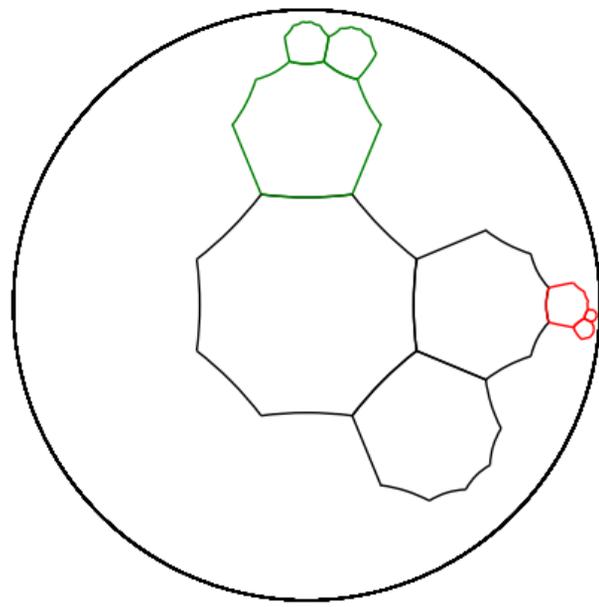
Side pairing $(-, 1_0, 4_6)$ maps the black domain to the red one, its inverse maps the black domain to the green one



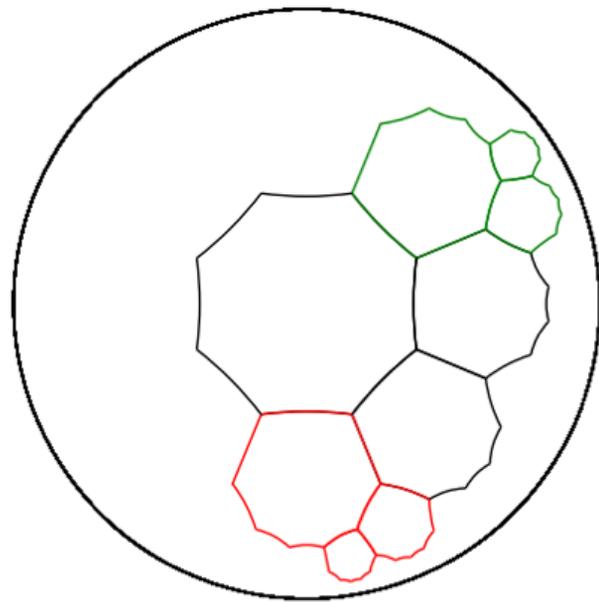
Side pairing $(-, 1_7, 1_5)$ maps the black domain to the red one, its inverse maps the black domain to the green one



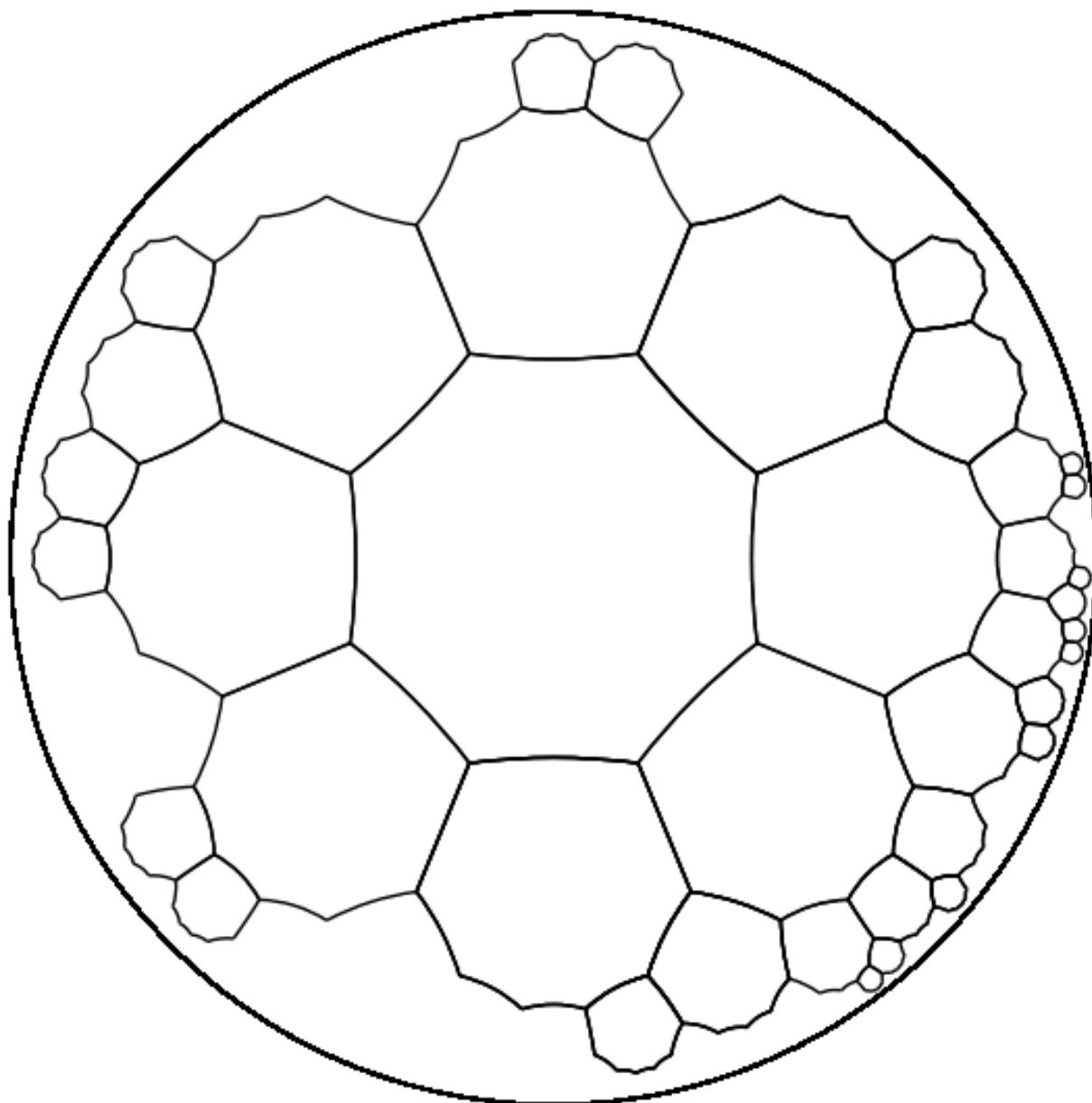
Side pairing $(+, 1_6, 4_5)$ maps the black domain to the red one, its inverse maps the black domain to the green one



Side pairing $(-, 1_4, 4_2)$ maps the black domain to the red one, its inverse maps the black domain to the green one



Side pairing $(-, 4_7, 3_1)$ 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 EXTERNAL 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=0.721675328781099 + 0.188504392343353*I
tau=rotationpi((B).coordinates(),candidate)

conjugates=['cero',
sidepairings[8]^(-1)*sidepairings[4]*sidepairings[8],
sidepairings[3]*sidepairings[8],
sidepairings[3],
sidepairings[8]^(-1)*sidepairings[4]*sidepairings[8],
sidepairings[5],
sidepairings[9]^(-1)*sidepairings[7]^(-1),
sidepairings[7],
sidepairings[8],
sidepairings[5]^(-1)]

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

```

Isometry in PD
[ 0.9999999999999992 - 5.71764857681956e-14*I -3.51940698806175e
-14 + 4.45199432874688e-14*I]
[-3.51940698806175e-14 - 4.45199432874688e-14*I 0.999999999999
992 + 5.71764857681956e-14*I]
Isometry in PD
[ 0.9999999999999999 + 1.56541446472147e-13*I 9.90318937965640e-1
4 - 1.21347376591530e-13*I]
[9.90318937965640e-14 + 1.21347376591530e-13*I 0.99999999999999
9 - 1.56541446472147e-13*I]
Isometry in PD
[ -0.9999999999999997 + 3.43058914609173e-14*I 1.87627691161651e-1
4 - 3.06421554796543e-14*I]
[1.87627691161651e-14 + 3.06421554796543e-14*I -0.99999999999999
7 - 3.43058914609173e-14*I]
Isometry in PD
[ -0.9999999999999992 + 4.45199432874688e-14*I 2.58681964737661e-1
4 - 3.57491813929300e-14*I]
[2.58681964737661e-14 + 3.57491813929300e-14*I -0.99999999999999
2 - 4.45199432874688e-14*I]
Isometry in PD
[ -0.9999999999999998 - 1.58761892521397e-14*I -9.65894031423886e
-15 + 1.38777878078145e-14*I]
[-9.65894031423886e-15 - 1.38777878078145e-14*I -0.99999999999999
998 + 1.58761892521397e-14*I]
Isometry in PD
[ -0.9999999999999998 - 8.70414851306123e-14*I -3.28626015289046e
-14 + 7.89368570508486e-14*I]
[-3.28626015289046e-14 - 7.89368570508486e-14*I -0.99999999999999
998 + 8.70414851306123e-14*I]
Isometry in PD
[ 0.9999999999999999 + 3.46389583683049e-14*I 7.10542735760100e-1
5 - 3.34177130412172e-14*I]
[7.10542735760100e-15 + 3.34177130412172e-14*I 0.9999999999999999
9 - 3.46389583683049e-14*I]
Isometry in PD
[ -0.9999999999999998 + 4.51860771022439e-14*I -6.10622663543836e
-15 - 4.66293670342566e-14*I]
[-6.10622663543836e-15 + 4.66293670342566e-14*I -0.99999999999999
998 - 4.51860771022439e-14*I]
Isometry in PD
[ -0.9999999999999999 + 9.99200722162641e-15*I -2.66453525910038e
-15 - 3.66373598126302e-15*I]
[-2.66453525910038e-15 + 3.66373598126302e-15*I -0.99999999999999
999 - 9.99200722162641e-15*I]

```

In [8]:

```

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

```

Out[8]:

```

[Point in PD 0.433159962429712 + 0.11314306066578828*I]

```