

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=9
```

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=9 #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

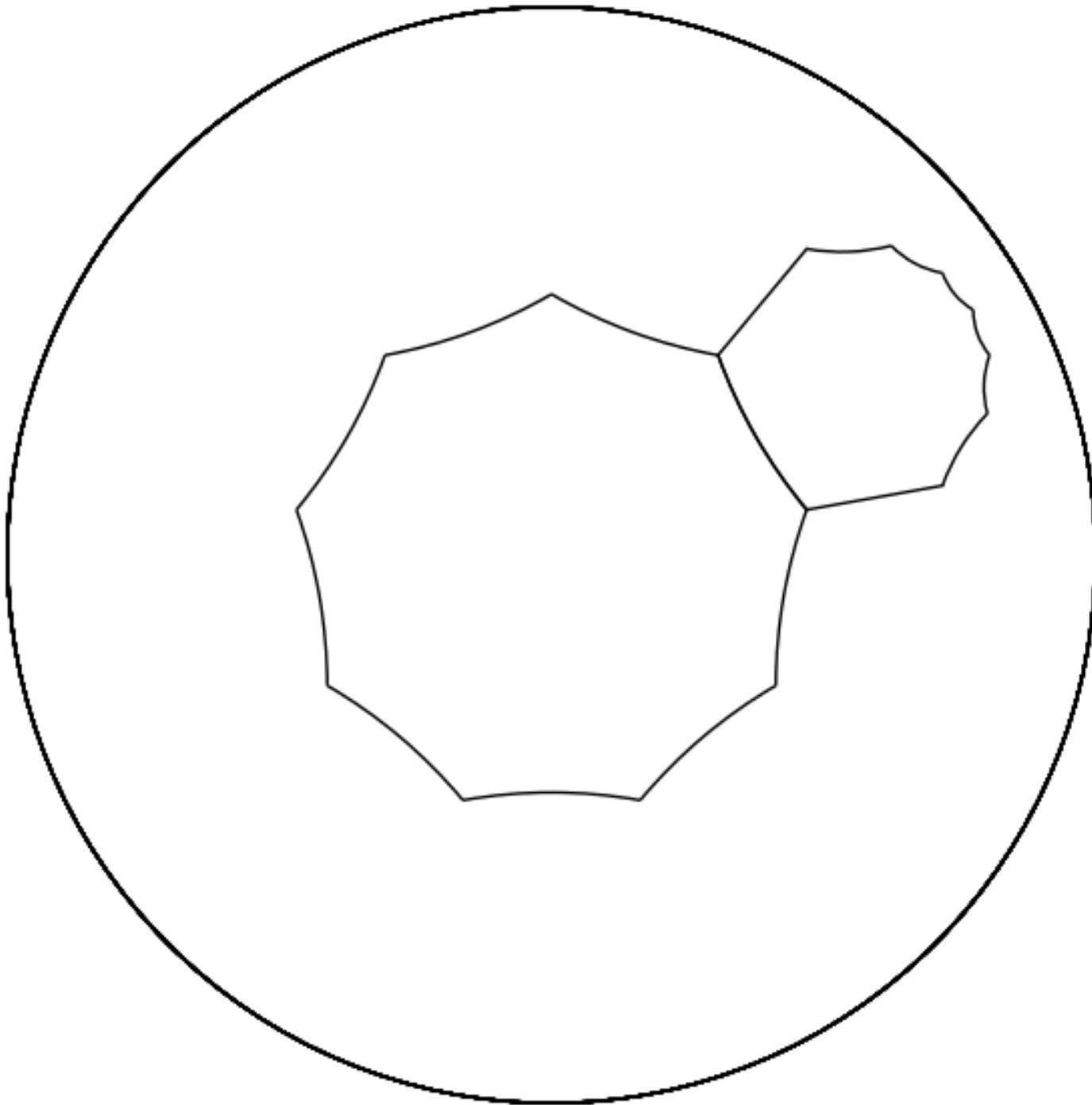
plotdom=moverpol(a*a^-1,'black')
plotdom=plotdom+moverpol((c*a)^3*a*b*(c*a)^-3,'black')

show(plotdom)

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

sidepairings.append((c*a)*b) #1 Side pairing (-, 1
_0, 1_1)
sidepairingscode.append('(-, 1_0, 1_1)')
sidepairings.append((c*a)^2*a*b*(c*a)) #2 Side pairing (+, 1
_8, 1_2)
sidepairingscode.append('(+, 1_8, 1_2)')
sidepairings.append(G3(1)*(c*a)^5) #3 Side pairing (+, 1
_7, 5_1)
sidepairingscode.append('(+, 1_7, 5_1)')
sidepairings.append(G3(3)*(c*a)^3*a*(c*a)^3) #4 Side pairing (-, 1
_6, 5_3)
sidepairingscode.append('(-, 1_6, 5_3)')
sidepairings.append(G3(5)*(c*a)^7) #5 Side pairing (+, 1
_5, 5_5)
sidepairingscode.append('(+, 1_5, 5_5)')
sidepairings.append(G3(-2)*(c*a)^3*a*(c*a)^5) #6 Side pairing (-, 1
_4, 5_7)
sidepairingscode.append('(-, 1_4, 5_7)')
sidepairings.append(G3(-3)*(c*a)^3*b*(c*a)^-3*G3(1)) #7 Side pairing (-, 5
_8, 5_6)
sidepairingscode.append('(-, 5_8, 5_6)')
sidepairings.append(G3(4)*(c*a)^3*b*(c*a)^-3*G3(-2)) #8 Side pairing (-, 5
_2, 5_4)
sidepairingscode.append('(-, 5_2, 5_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
```

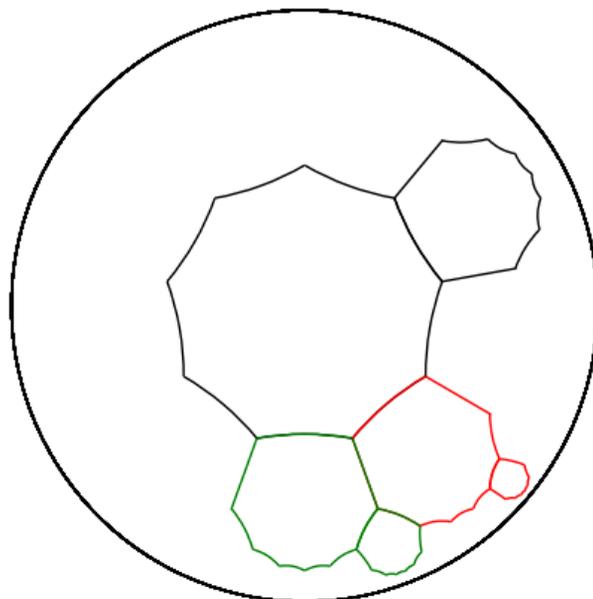
```
total=plotdom;
for j in [1..len(sidepairings)-1]:

    pairing=sidepairings[j]
    resultado=plotdom+moverpol(pairing,'red')
    total=total+moverpol(pairing,'black')
    for k in [3..3]:
        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')

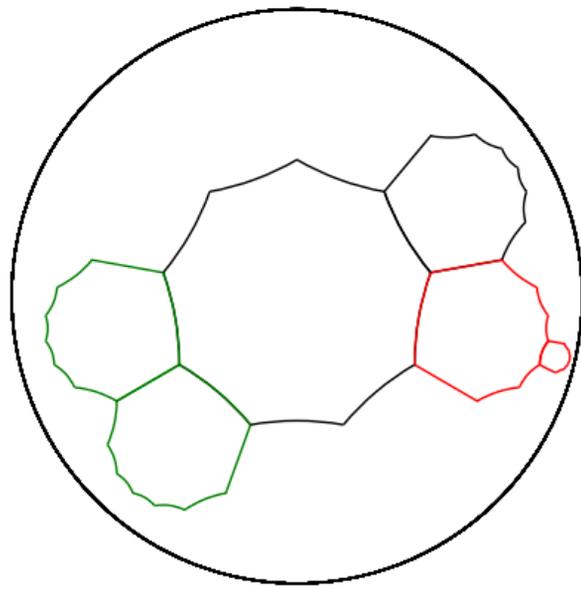
    pairing_inv=sidepairings[j]^(-1)
    resultado=resultado+moverpol(pairing_inv,'green')
    total=total+moverpol(pairing_inv,'black')
    for k in [3..3]:
        resultado=resultado+moverpol(pairing_inv*(c*a)^k*a*b*(c*a)^-k,'green')
        total=total+moverpol(pairing_inv*(c*a)^k*a*b*(c*a)^-k,'black')
    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))
```

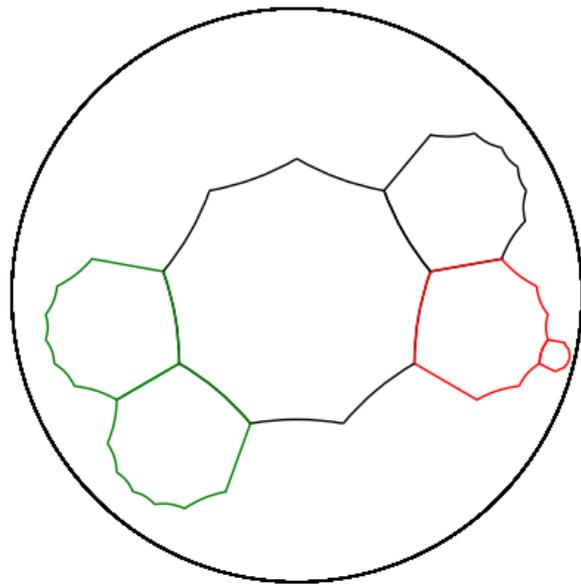
/Applications/SageMath-8.2.app/Contents/Resources/sage/local/lib/python2.7/site-packages/sage/repl/ipython_kernel/__main__.py:31: DeprecationWarning: show is deprecated. Please use plot instead. See <http://trac.sagemath.org/20530> for details.



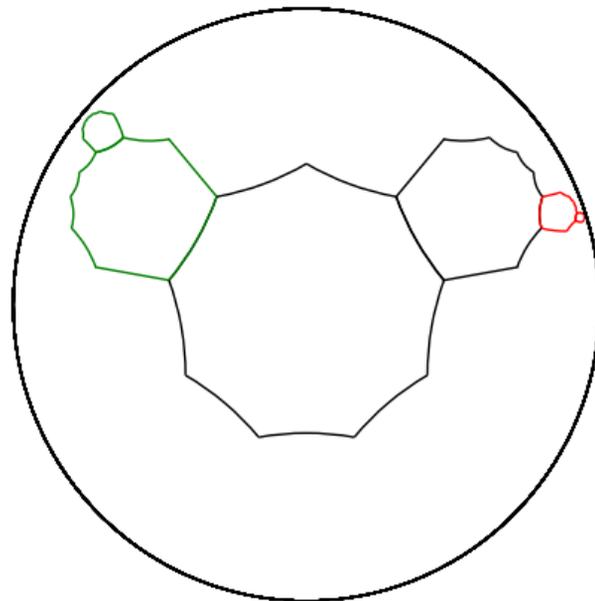
Side pairing $(-, 1_0, 1_1)$ maps the black domain to the red one, its inverse maps the black domain to the green one



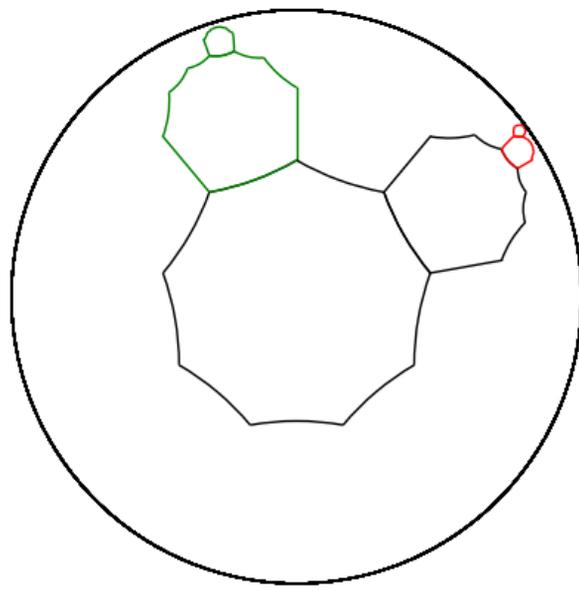
Side pairing $(+, 1_8, 1_2)$ maps the black domain to the red one, its inverse maps the black domain to the green one



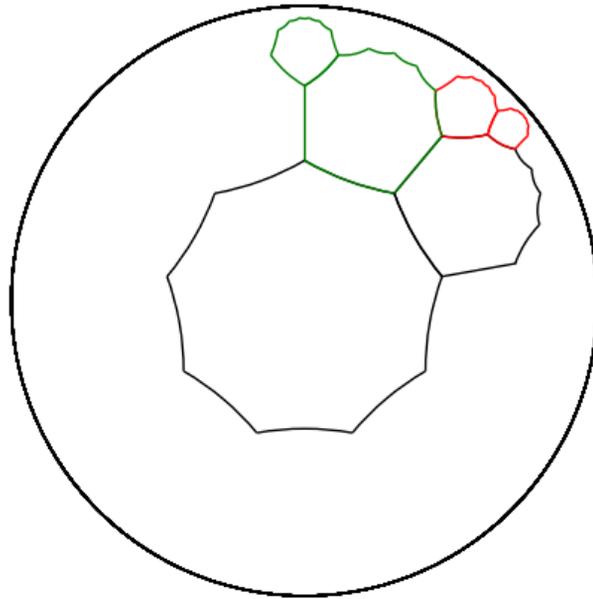
Side pairing $(+, 1_7, 5_1)$ maps the black domain to the red one, its inverse maps the black domain to the green one



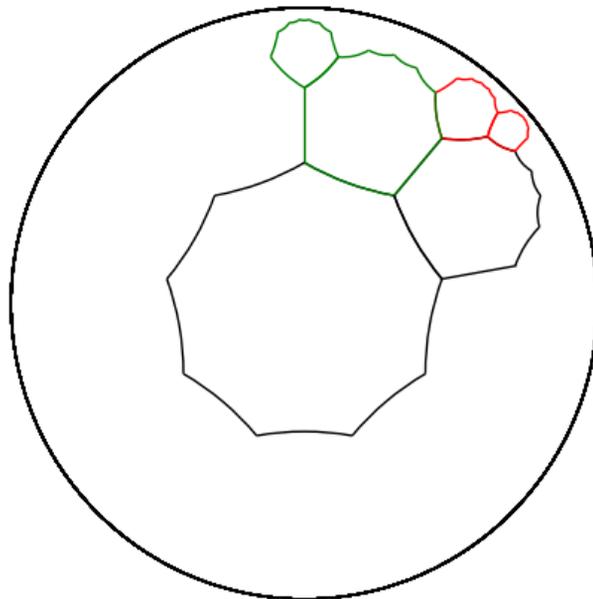
Side pairing $(-, 1_6, 5_3)$ maps the black domain to the red one, its inverse maps the black domain to the green one



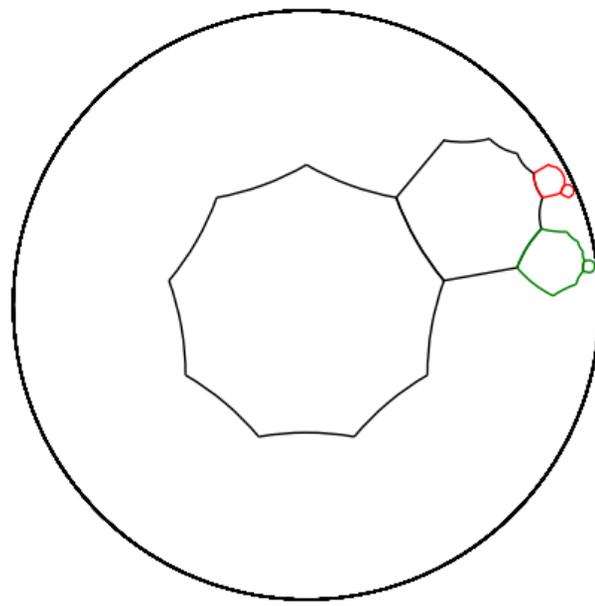
Side pairing (+, 1_5, 5_5) maps the black domain to the red one, its inverse maps the black domain to the green one



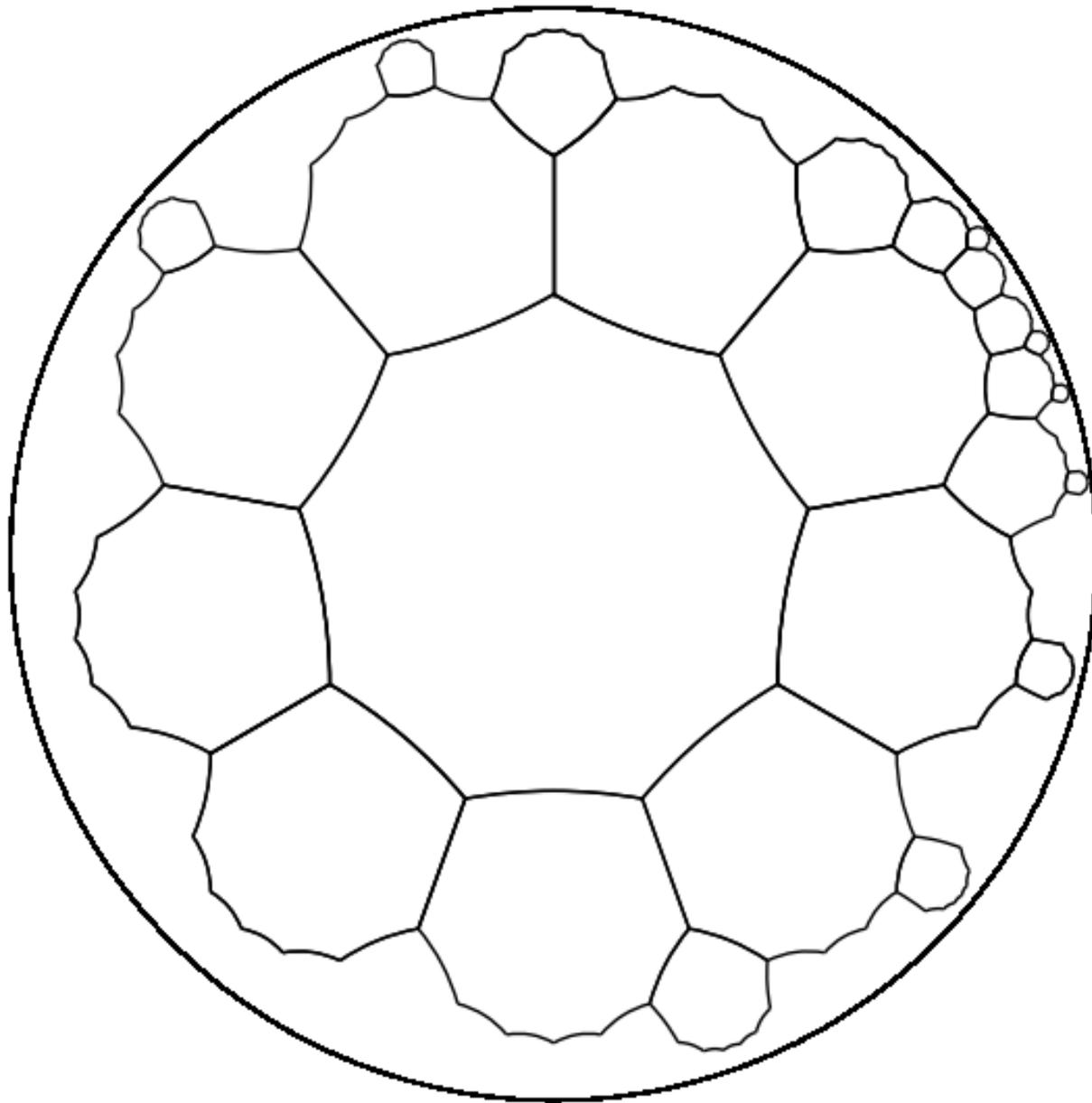
Side pairing (-, 1_4, 5_7) maps the black domain to the red one, its inverse maps the black domain to the green one



Side pairing (-, 5_8, 5_6) maps the black domain to the red one, its inverse maps the black domain to the green one



Side pairing $(-, 5_2, 5_4)$ 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 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=0.714302207008976 + 0.451272035332353*I
tau=rotationpi(((c*a)^3*b*B).coordinates(),candidate)

conjugates=['cero',
(sidepairings[5]*sidepairings[1]^(-1)*sidepairings[8]^(-2))^(-1),
(sidepairings[5]*sidepairings[8]^(-2))^(-1),
(sidepairings[7]^2*sidepairings[8]^(-2))^(-1),
(sidepairings[7]*sidepairings[8]^(-2))^(-1),
(sidepairings[3]*sidepairings[5]^(-1))^(-1),
sidepairings[8],
sidepairings[8],
sidepairings[6]]

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

```

Isometry in PD
[ -1.000000000000019 - 1.67017510932510e-11*I -9.15134634738024e
-12 + 1.39666056497845e-11*I]
[-9.15134634738024e-12 - 1.39666056497845e-11*I -1.000000000000
019 + 1.67017510932510e-11*I]
Isometry in PD
[ -0.999999999999788 + 9.07762753854513e-12*I 4.68913796680681e-1
2 - 7.77600206447460e-12*I]
[4.68913796680681e-12 + 7.77600206447460e-12*I -0.99999999999978
8 - 9.07762753854513e-12*I]
Isometry in PD
[ -0.999999999999923 + 1.13398179735213e-11*I 6.07203176627991e-1
2 - 9.58433332698405e-12*I]
[6.07203176627991e-12 + 9.58433332698405e-12*I -0.99999999999992
3 - 1.13398179735213e-11*I]
Isometry in PD
[ 1.000000000000001 + 4.49151826842353e-12*I 2.33679742223103e-1
2 - 3.83248988100604e-12*I]
[2.33679742223103e-12 + 3.83248988100604e-12*I 1.00000000000000
1 - 4.49151826842353e-12*I]
Isometry in PD
[ -0.999999999999901 - 2.05124806029744e-12*I -1.06448183601060e
-12 + 1.75326420048805e-12*I]
[-1.06448183601060e-12 - 1.75326420048805e-12*I -0.999999999999
901 + 2.05124806029744e-12*I]
Isometry in PD
[ -0.999999999999997 + 9.92539384014890e-13*I 4.99600361081320e-1
3 - 8.60644888689421e-13*I]
[4.99600361081320e-13 + 8.60644888689421e-13*I -0.99999999999999
7 - 9.92539384014890e-13*I]
Isometry in PD
[ 0.999999999999999 - 9.33475519104832e-13*I -4.73399097700167e
-13 + 8.07354183507414e-13*I]
[-4.73399097700167e-13 - 8.07354183507414e-13*I 0.999999999999
999 + 9.33475519104832e-13*I]
Isometry in PD
[ -0.999999999999994 + 4.48530101948563e-13*I 2.58459920132736e-1
3 - 3.73923114693753e-13*I]
[2.58459920132736e-13 + 3.73923114693753e-13*I -0.99999999999999
4 - 4.48530101948563e-13*I]

```

In [8]:

```

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

```

Out[8]:

```

[Point in PD 0.676705896528954 + 0.414595401210975*I]

```