monoides de faces 2

Franco Saliola

10/3/2014

```
class FaceMonoid(Parent):
    def __init__(self, generators):
        Parent.__init__(self, category = FiniteMonoids())
        self._semigroup_generators = map(self._element_constructor_, \
           generators)

    def _repr_(self):
        return "Mono de de vecteurs de signes engendr  par: %s" % (self\
           ._semigroup_generators)

    def one(self):
        length = len(self._semigroup_generators[0].value)
        return self((0,) * length)

    def semigroup_generators(self):
        return [self.one()] + [self(g) for g in self.\
           _semigroup_generators]

    def product(self, x, y):
        xv = x.value
        yv = y.value
        z = tuple([xv[i] if xv[i] != 0 else yv[i] for i in range(len(xv))\
           ])
        return self(z)

    def regions(self):
        return [x for x in self if 0 not in x.value]

    class Element(ElementWrapper):
        wrapped_class = tuple

        def _repr_(self):
            d = {0:'0', 1:'+', -1:'-'}
            return "".join([d[i] for i in self.value])
```

```
F = FaceMonoid([(1,-1,0), (-1,1,0), (1,0,1), (-1,0,-1), (0,1,1), \
   (0,-1,-1)])
F
```

```
Monoïde de vecteurs de signes engendré par: [+-0, -+0, +0+, -0-, 0++, 0--]
```

```
C = F.regions()
C
```
```
[-++, +--, ---, -+-, +++, +-+]
```

```
prob = {}
prob[F((1,0,1))] = prob[F((0,-1,-1))] = prob[F((-1, 1,0))] = 1/3
prob
```
```
{-+0: 1/3, 0--: 1/3, +0+: 1/3}
```

```
T = matrix(QQ, len(C), len(C))
T
```
```
[0 0 0 0 0 0]
[0 0 0 0 0 0]
[0 0 0 0 0 0]
[0 0 0 0 0 0]
[0 0 0 0 0 0]
[0 0 0 0 0 0]
```

```
for i in range(len(C)):
    for j in range(len(C)):
        T[i, j] = sum(prob[x] for x in F if x in prob and x * C[i] == C[j\
           ])
```

```
T
```
```
[1/3   0 1/3   0 1/3   0]
[  0 1/3   0 1/3   0 1/3]
[  0   0 1/3 1/3   0 1/3]
[  0   0 1/3 1/3 1/3   0]
[1/3 1/3   0   0 1/3   0]
[1/3 1/3   0   0   0 1/3]
```

```
T.eigenvalues()
```
```
[1, 0, 0, 1/3, 1/3, 1/3]
```

```
T.left_eigenspaces()
```
```
[
(1, Vector space of degree 6 and dimension 1 over Rational Field
User basis matrix:
[1 1 1 1 1 1]),
(0, Vector space of degree 6 and dimension 2 over Rational Field
User basis matrix:
[ 1  1  0 -1  0 -1]
[ 0  0  1 -1  1 -1]),
(1/3, Vector space of degree 6 and dimension 3 over Rational Field
User basis matrix:
[ 1  0  0 -1  0  0]
[ 0  1 -1  0  0  0]
[ 0  0  0  0  1 -1])
]
```

```
F = FaceMonoid([(1,0,0),(-1,0,0),(0,1,0),(0,-1,0),(0,0,1),(0,0,-1)])
F
```
```
Monoïde de vecteurs de signes engendré par: [+00, -00, 0+0, 0-0, 00+, 00-]
```

```
F.cardinality()
```
```
27
```

```
prob = {}
for x in [(1,0,0),(-1,0,0),(0,1,0),(0,-1,0),(0,0,1),(0,0,-1)]:
    prob[F(x)] = 1/6
prob
```
```
{+00: 1/6, 00+: 1/6, 00-: 1/6, 0-0: 1/6, -00: 1/6, 0+0: 1/6}
```

```
C = F.regions()
C
```
```
[-+-, +++, ++-, --+, +-+, ---, +--, -++]
```

```
T = matrix(QQ, len(C), len(C))
```

```
for i in range(len(C)):
    for j in range(len(C)):
        T[i,j] = sum(prob[x] for x in prob if x * C[i] == C[j])
```

```
T
```
```
[1/2   0 1/6   0   0 1/6   0 1/6]
[  0 1/2 1/6   0 1/6   0   0 1/6]
[1/6 1/6 1/2   0   0   0 1/6   0]
[  0   0   0 1/2 1/6 1/6   0 1/6]
[  0 1/6   0 1/6 1/2   0 1/6   0]
[1/6   0   0 1/6   0 1/2 1/6   0]
[  0   0 1/6   0 1/6 1/6 1/2   0]
[1/6 1/6   0 1/6   0   0   0 1/2]
```

```
T.eigenvalues()
```
```
[1, 0, 2/3, 2/3, 2/3, 1/3, 1/3, 1/3]
```

```
T.is_diagonalizable()
```
```
True
```

```
F = FaceMonoid([(1,0),(-1,0),(0,1),(0,-1)])
F
```
```
Monoïde de vecteurs de signes engendré par: [+0, -0, 0+, 0-]
```

```
F.cardinality()
```
```
9
```

```
F.list()
```
```
[0+, -+, 00, -0, --, 0-, +0, +-, ++]
```

```
R = SR
p = var('p0,p1,p2,p3,p4,p5,p6,p7,p8')
```

```
prob = {}
for i in range(9):
    prob[F.list()[i]] = p[i]
```

```
prob
```
```
{0+: p0, -+: p1, 00: p2, -0: p3, --: p4, 0-: p5, +0: p6, +-: p7, ++: p8}
```

```
sum(prob.values())
```
```
p0 + p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8
```

```
C = F.regions()
C
```
```
[-+, --, +-, ++]
```

```
T = matrix(R, 4, 4)
for i in range(len(C)):
    for j in range(len(C)):
        T[i,j] = sum(prob[x] for x in prob if x * C[i] == C[j])
```

```
T
```
```
[p0 + p1 + p2 + p3          p4 + p5                p7          p6 + p8]
[         p0 + p1 p2 + p3 + p4 + p5          p6 + p7               p8]
[              p1          p3 + p4 p2 + p5 + p6 + p7          p0 + p8]
[         p1 + p3               p4          p5 + p7 p0 + p2 + p6 + p8]
```

```
show(T)
```
$$\begin{pmatrix} p_0+p_1+p_2+p_3 & p_4+p_5 & p_7 & p_6+p_8 \\ p_0+p_1 & p_2+p_3+p_4+p_5 & p_6+p_7 & p_8 \\ p_1 & p_3+p_4 & p_2+p_5+p_6+p_7 & p_0+p_8 \\ p_1+p_3 & p_4 & p_5+p_7 & p_0+p_2+p_6+p_8 \end{pmatrix}$$

```
f = T.characteristic_polynomial()
```

```
f
```
```
x^4 + (-2*p0 - p1 - 4*p2 - 2*p3 - p4 - 2*p5 - 2*p6 - p7 - p8)*x^3 + (p0^2 + p0*p1 +
6*p0*p2 + 3*p1*p2 + 6*p2^2 + 3*p0*p3 + p1*p3 + 6*p2*p3 + p3^2 + p0*p4 + 3*p2*p4 + p3*p4 +
2*p0*p5 + p1*p5 + 6*p2*p5 + 3*p3*p5 + p4*p5 + p5^2 + 3*p0*p6 + p1*p6 + 6*p2*p6 + 2*p3*p6 +
p4*p6 + 3*p5*p6 + p6^2 + p0*p7 + 3*p2*p7 + p3*p7 + p5*p7 + p6*p7 + p0*p8 + 3*p2*p8 + p3*p8
+ p5*p8 + p6*p8)*x^2 + (-2*p0^2*p2 - 2*p0*p1*p2 - 6*p0*p2^2 - 3*p1*p2^2 - 4*p2^3 - p0^2*p3
- p0*p1*p3 - 6*p0*p2*p3 - 2*p1*p2*p3 - 6*p2^2*p3 - p0*p3^2 - 2*p2*p3^2 - 2*p0*p2*p4 -
3*p2^2*p4 - p0*p3*p4 - 2*p2*p3*p4 - 4*p0*p2*p5 - 2*p1*p2*p5 - 6*p2^2*p5 - 2*p0*p3*p5 -
p1*p3*p5 - 6*p2*p3*p5 - p3^2*p5 - 2*p2*p4*p5 - p3*p4*p5 - 2*p2*p5^2 - p3*p5^2 - p0^2*p6 -
p0*p1*p6 - 6*p0*p2*p6 - 2*p1*p2*p6 - 6*p2^2*p6 - 2*p0*p3*p6 - 4*p2*p3*p6 - p0*p4*p6 -
2*p2*p4*p6 - 2*p0*p5*p6 - p1*p5*p6 - 6*p2*p5*p6 - 2*p3*p5*p6 - p4*p5*p6 - p5^2*p6 -
p0*p6^2 - 2*p2*p6^2 - p5*p6^2 - 2*p0*p2*p7 - 3*p2^2*p7 - p0*p3*p7 - 2*p2*p3*p7 -
2*p2*p5*p7 - p3*p5*p7 - p0*p6*p7 - 2*p2*p6*p7 - p5*p6*p7 - 2*p0*p2*p8 - 3*p2^2*p8 -
p0*p3*p8 - 2*p2*p3*p8 - 2*p2*p5*p8 - p3*p5*p8 - p0*p6*p8 - 2*p2*p6*p8 - p5*p6*p8)*x +
p0^2*p2^2 + p0*p1*p2^2 + 2*p0*p2^3 + p1*p2^3 + p2^4 + p0^2*p2*p3 + p0*p1*p2*p3 +
3*p0*p2^2*p3 + p1*p2^2*p3 + 2*p2^3*p3 + p0*p2*p3^2 + p2^2*p3^2 + p0*p2^2*p4 + p2^3*p4 +
p0*p2*p3*p4 + p2^2*p3*p4 + 2*p0*p2^2*p5 + p1*p2^2*p5 + 2*p2^3*p5 + 2*p0*p2*p3*p5 +
p1*p2*p3*p5 + 3*p2^2*p3*p5 + p2*p3^2*p5 + p2^2*p4*p5 + p2*p3*p4*p5 + p2^2*p5^2 +
```

```
p2*p3*p5^2 + p0^2*p2*p6 + p0*p1*p2*p6 + 3*p0*p2^2*p6 + p1*p2^2*p6 + 2*p2^3*p6 +
2*p0*p2*p3*p6 + 2*p2^2*p3*p6 + p0*p2*p4*p6 + p2^2*p4*p6 + 2*p0*p2*p5*p6 + p1*p2*p5*p6 +
3*p2^2*p5*p6 + 2*p2*p3*p5*p6 + p2*p4*p5*p6 + p2*p5^2*p6 + p0*p2*p6^2 + p2^2*p6^2 +
p2*p5*p6^2 + p0*p2^2*p7 + p2^3*p7 + p0*p2*p3*p7 + p2^2*p3*p7 + p2^2*p5*p7 + p2*p3*p5*p7 +
p0*p2*p6*p7 + p2^2*p6*p7 + p2*p5*p6*p7 + p0*p2^2*p8 + p2^3*p8 + p0*p2*p3*p8 + p2^2*p3*p8 +
p2^2*p5*p8 + p2*p3*p5*p8 + p0*p2*p6*p8 + p2^2*p6*p8 + p2*p5*p6*p8
```

```
T.eigenvalues()
```
```
[p0 + p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8, p2 + p3 + p6, p0 + p2 + p5, p2]
```

```
for v in T.eigenvalues():
    show(v)
```

$$p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8$$

$$p_2 + p_3 + p_6$$

$$p_0 + p_2 + p_5$$

$$p_2$$

```
T.is_diagonalizable()
```
```
Error in lines 1-1
Traceback (most recent call last):
  File "/projects/25c7c359-325c-40ee-88bc-3d83396ec94d/.sagemathcloud/sage_server.py",
line 733, in execute
    exec compile(block+'\n', '', 'single') in namespace, locals
  File "", line 1, in <module>
  File "matrix2.pyx", line 9340, in sage.matrix.matrix2.Matrix.is_diagonalizable
(sage/matrix/matrix2.c:47606)
ValueError: base field must be exact, not Symbolic Ring
```