In [189]:
```
import_statements("CategoryWithAxiom")
from sage.structure.element import Element
from sage.categories.category_with_axiom import CategoryWithAxiom
from sage.categories.morphism import SetMorphism
```

```
from sage.categories.category_with_axiom import CategoryWithAxiom
```

In [190]:
```
class Stroumphs(Category):
    def super_categories(self):
        return [Sets()]
    class ParentMethods:
        def party(self):
            print "party time!"
    class ElementMethods:
        def sing(self):
            print "I sing!"
    class Finite(CategoryWithAxiom):
        class ParentMethods:
            def party(self):
                print "party time"
                for stroumph in self:
                    stroumph.sing()
```

In [191]:
```
Stroumphs().element_class.sing
```

Out[191]:
```
<unbound method Stroumphs.element_class.sing>
```

In [192]:
```
F2(11)
```

Out[192]:
```
1
```

In [193]:
```
class Trivial(UniqueRepresentation, Parent):
    def __init__(self):
        Parent.__init__(self, category=Stroumphs() & Groups() & EnumeratedSe

    def semigroup_generators(self):
        return Family([self.an_element()])
    gens = semigroup_generators

    def an_element(self):
        return self("the unique element of Trivial")

    def one(self):
        return self.an_element()

    def __iter__(self):
        yield self.an_element()

    class Element(ElementWrapper):
        def _mul_(self, other):
            return self

        def __invert__(self):
            return self
```

In [ ]:

In [194]:
```
T = Trivial()
```

In [ ]:

In [195]:
```
phi = SetMorphism(Hom(T, GF(2)), lambda x: GF(2).one())
```

In [201]:
```
phi(T.an_element())
```

Out[201]: 1

In [202]:
```
phi.register_as_coercion()
```

In [204]:
```
T.an_element() * GF(2).one()
```

Out[204]: 1

In [185]:
```
T.party()
```

```
party time
I sing!
```

In [167]:
```
stroumph = T.an_element()
stroumph.__class__
stroumph.sing()
```

```
I sing!
```

In [160]:
```
stroumph.__class__.mro()
```

Out[160]:
```
[<class '__main__.Trivial_with_category.element_class'>,
 <class '__main__.Trivial.Element'>,
 <type 'sage.structure.element_wrapper.ElementWrapper'>,
 <type 'sage.structure.element.Element'>,
 <type 'sage.structure.sage_object.SageObject'>,
 <class 'sage.categories.category.JoinCategory.element_class'>,
 <class 'sage.categories.finite_groups.FiniteGroups.element_class'>,
 <class 'sage.categories.finite_monoids.FiniteMonoids.element_class'>,
 <class 'sage.categories.groups.Groups.element_class'>,
 <class 'sage.categories.monoids.Monoids.element_class'>,
 <class 'sage.categories.finite_semigroups.FiniteSemigroups.element_class'
>,
 <class 'sage.categories.semigroups.Semigroups.element_class'>,
 <class 'sage.categories.magmas.Magmas.Unital.Inverse.element_class'>,
 <class 'sage.categories.magmas.Magmas.Unital.element_class'>,
 <class 'sage.categories.magmas.Magmas.element_class'>,
 <class 'sage.categories.finite_enumerated_sets.FiniteEnumeratedSets.eleme
nt_class'>,
 <class 'sage.categories.enumerated_sets.EnumeratedSets.element_class'>,
 <class 'sage.categories.finite_sets.FiniteSets.element_class'>,
 <class '__main__.Stroumphs.element_class'>,
 <class 'sage.categories.sets_cat.Sets.element_class'>,
 <class 'sage.categories.sets_with_partial_maps.SetsWithPartialMaps.elemen
t_class'>,
 <class 'sage.categories.objects.Objects.element_class'>,
 <type 'object'>]
```

In [87]:
```
x = T.an_element(); x
```

Out[87]: 'the unique element of Trivial'

In [88]:
```
x * x
```

Out[88]: 'the unique element of Trivial'

In [89]:
```
x^4
```

Out[89]: 'the unique element of Trivial'

In [66]:
```
x.parent() is T
```

Out[66]: True

In [69]:
```
T.random_element()
```

Out[69]: 'the unique element of Trivial'

In [68]:
```
T.cardinality()
```

Out[68]: 1

In [67]:
```
list(T)
```

Out[67]: ['the unique element of Trivial']

In [151]:
```
TestSuite(T).run(verbose=True)
```

```
running ._test_an_element() . . . pass
running ._test_associativity() . . . pass
running ._test_category() . . . pass
running ._test_elements() . . .
  Running the test suite of self.an_element()
  running ._test_category() . . . pass
  running ._test_eq() . . . pass
  running ._test_not_implemented_methods() . . . pass
  running ._test_pickling() . . . fail
  Traceback (most recent call last):
    File "/opt/sage-git2/local/lib/python2.7/site-packages/sage/misc/sage_
unittest.py", line 282, in run
      test_method(tester = tester)
    File "sage/structure/sage_object.pyx", line 550, in sage.structure.sag
e_object.SageObject._test_pickling (build/cythonized/sage/structure/sage_o
bject.c:4495)
      tester.assertEqual(loads(dumps(self)), self)
    File "sage/structure/sage_object.pyx", line 1015, in sage.structure.sa
ge_object.dumps (build/cythonized/sage/structure/sage_object.c:11613)
      return obj.dumps(compress)
    File "sage/structure/sage_object.pyx", line 371, in sage.structure.sag
e_object.SageObject.dumps (build/cythonized/sage/structure/sage_object.c:3
183)
      s = cPickle.dumps(self, protocol=2)
  PicklingError: Can't pickle <class '__main__.Stroumphs'>: it's not the s
ame object as __main__.Stroumphs
  ----------------------------------------------------------------------
  The following tests failed: _test_pickling
running ._test_elements_eq_reflexive() . . . pass
running ._test_elements_eq_symmetric() . . . pass
running ._test_elements_eq_transitive() . . . pass
running ._test_elements_neq() . . . pass
running ._test_enumerated_set_contains() . . . pass
running ._test_enumerated_set_iter_cardinality() . . . pass
running ._test_enumerated_set_iter_list() . . . pass
running ._test_eq() . . . pass
running ._test_inverse() . . . pass
running ._test_not_implemented_methods() . . . pass
running ._test_one() . . . pass
running ._test_pickling() . . . pass
running ._test_prod() . . . pass
running ._test_some_elements() . . . pass
The following tests failed: _test_elements
```

In [12]:
```
T2 = Trivial()
```

In [17]:
```
ZZ.an_element()
```

Out[17]: 1

In [13]:
```
T == T2
```

Out[13]: True

In [14]:
```
T is T2
```

Out[14]: True

In [106]:
```
Groups().super_categories()
```

Out[106]:
```
[Category of monoids, Category of inverse unital magmas]
```

In [107]:
```
Groups().structure()
```

Out[107]:
```
frozenset({Category of unital magmas,
           Category of magmas,
           Category of sets with partial maps,
           Category of sets})
```

In [108]:
```
Groups().axioms()
```

Out[108]:
```
frozenset({'Associative', 'Inverse', 'Unital'})
```

In [113]:
```
G = Fields().category_graph()
```

In [114]:
```
G.set_latex_options(format="dot2tex")
```

In [117]:
```
view(G, tightpage=True)
```

In [118]:
```
C = Magmas()
```

In [129]:
```
D = (C.Associative().Unital() & AdditiveMagmas().AdditiveCommutative().Addi
```

In [132]:
```
D.Division() & Sets().Finite()
```

Out[132]:
```
Category of finite fields
```

In [ ]:
```

```