Standard Semirings

Release 10.6

The Sage Development Team

Jun 27, 2025

CONTENTS

CHAPTER

ONE

NON NEGATIVE INTEGER SEMIRING

class sage.rings.semirings.non_negative_integer_semiring.**NonNegativeIntegerSemiring**

Bases: NonNegativeIntegers

A class for the semiring of the nonnegative integers.

This parent inherits from the infinite enumerated set of non negative integers and endows it with its natural semiring structure.

EXAMPLES:

```
sage: NonNegativeIntegerSemiring()
Non negative integer semiring
```

```
>>> from sage.all import *
>>> NonNegativeIntegerSemiring()
Non negative integer semiring
```

For convenience, `NN` is a shortcut for `NonNegativeIntegerSemiring()`:

```
sage: NN == NonNegativeIntegerSemiring()
True

sage: NN.category()
Category of facade infinite enumerated commutative semirings
```

```
>>> from sage.all import *
>>> NN == NonNegativeIntegerSemiring()
True

>>> NN.category()
Category of facade infinite enumerated commutative semirings
```

Here is a piece of the Cayley graph for the multiplicative structure:

```
sage: G = NN.cayley_graph(elements=range(9), generators=[0,1,2,3,5,7])          #␣
↪needs sage.graphs
sage: G                                                                          #␣
↪needs sage.graphs
Looped multi-digraph on 9 vertices
sage: G.plot()                                                                   #␣
↪needs sage.graphs sage.plot
Graphics object consisting of 48 graphics primitives
```

```
>>> from sage.all import *
>>> G = NN.cayley_graph(elements=range(Integer(9)), generators=[Integer(0),
↪Integer(1),Integer(2),Integer(3),Integer(5),Integer(7)])          # needs sage.
↪graphs
>>> G                                                                         #␣
↪needs sage.graphs
Looped multi-digraph on 9 vertices
>>> G.plot()                                                                  #␣
↪needs sage.graphs sage.plot
Graphics object consisting of 48 graphics primitives
```

This is the Hasse diagram of the divisibility order on NN.

> sage: Poset(NN.cayley_graph(elements=[1..12], generators=[2,3,5,7,11])).show() # needs sage.combinat sage.graphs sage.plot

Note: as for NonNegativeIntegers, NN is currently just a "facade" parent; namely its elements are plain Sage Integers with Integer Ring as parent:

```
sage: x = NN(15); type(x)
<class 'sage.rings.integer.Integer'>
sage: x.parent()
Integer Ring
sage: x+3
18
```

```
>>> from sage.all import *
>>> x = NN(Integer(15)); type(x)
<class 'sage.rings.integer.Integer'>
>>> x.parent()
Integer Ring
>>> x+Integer(3)
18
```

additive_semigroup_generators()

Return the additive semigroup generators of self.

EXAMPLES:

```
sage: NN.additive_semigroup_generators()
Family (0, 1)
```

```
>>> from sage.all import *
>>> NN.additive_semigroup_generators()
Family (0, 1)
```

CHAPTER

TWO

TROPICAL SEMIRINGS

AUTHORS:

- Travis Scrimshaw (2013-04-28) - Initial version

class sage.rings.semirings.tropical_semiring.**TropicalSemiring**(*base*, *use_min=True*)

Bases: Parent, UniqueRepresentation

The tropical semiring.

Given an ordered additive semigroup R, we define the tropical semiring $T = R \cup \{+\infty\}$ by defining tropical addition and multiplication as follows:

$$a \oplus b = \min(a, b), \qquad a \odot b = a + b.$$

In particular, note that there are no (tropical) additive inverses (except for ∞), and every element in R has a (tropical) multiplicative inverse.

There is an alternative definition where we define $T = R \cup \{-\infty\}$ and alter tropical addition to be defined by

$$a \oplus b = \max(a, b).$$

To use the max definition, set the argument `use_min = False`.

Warning

`zero()` and `one()` refer to the tropical additive and multiplicative identities respectively. These are **not** the same as calling `T(0)` and `T(1)` respectively as these are **not** the tropical additive and multiplicative identities respectively.

Specifically do not use `sum(...)` as this converts 0 to 0 as a tropical element, which is not the same as `zero()`. Instead use the `sum` method of the tropical semiring:

```
sage: T = TropicalSemiring(QQ)

sage: sum([T(1), T(2)]) # This is wrong
0
sage: T.sum([T(1), T(2)]) # This is correct
1
```

```
>>> from sage.all import *
>>> T = TropicalSemiring(QQ)

>>> sum([T(Integer(1)), T(Integer(2))]) # This is wrong
0
>>> T.sum([T(Integer(1)), T(Integer(2))]) # This is correct
1
```

Be careful about using code that has not been checked for tropical safety.

INPUT:

- `base` – the base ordered additive semigroup R
- `use_min` – boolean (default: `True`); if `True`, then the semiring uses $a \oplus b = \min(a, b)$. Otherwise uses $a \oplus b = \max(a, b)$.

EXAMPLES:

```
sage: T = TropicalSemiring(QQ)
sage: elt = T(2); elt
2
```

```
>>> from sage.all import *
>>> T = TropicalSemiring(QQ)
>>> elt = T(Integer(2)); elt
2
```

Recall that tropical addition is the minimum of two elements:

```
sage: T(3) + T(5)
3
```

```
>>> from sage.all import *
>>> T(Integer(3)) + T(Integer(5))
3
```

Tropical multiplication is the addition of two elements:

```
sage: T(2) * T(3)
5
sage: T(0) * T(-2)
-2
```

```
>>> from sage.all import *
>>> T(Integer(2)) * T(Integer(3))
5
>>> T(Integer(0)) * T(-Integer(2))
-2
```

We can also do tropical division and arbitrary tropical exponentiation:

```
sage: T(2) / T(1)
1
sage: T(2)^(-3/7)
-6/7
```

```
>>> from sage.all import *
>>> T(Integer(2)) / T(Integer(1))
1
>>> T(Integer(2))**(-Integer(3)/Integer(7))
-6/7
```

Note that "zero" and "one" are the additive and multiplicative identities of the tropical semiring. In general, they are **not** the elements 0 and 1 of R, respectively, even if such elements exist (e.g., for $R = \mathbf{Z}$), but instead the (tropical) additive and multiplicative identities $+\infty$ and 0 respectively:

```
sage: T.zero() + T(3) == T(3)
True
sage: T.one() * T(3) == T(3)
True
sage: T.zero() == T(0)
False
sage: T.one() == T(1)
False
```

```
>>> from sage.all import *
>>> T.zero() + T(Integer(3)) == T(Integer(3))
True
>>> T.one() * T(Integer(3)) == T(Integer(3))
True
>>> T.zero() == T(Integer(0))
False
>>> T.one() == T(Integer(1))
False
```

Element

alias of *TropicalSemiringElement*

additive_identity()

Return the (tropical) additive identity element $+\infty$.

EXAMPLES:

```
sage: T = TropicalSemiring(QQ)
sage: T.zero()
+infinity
```

```
>>> from sage.all import *
>>> T = TropicalSemiring(QQ)
>>> T.zero()
+infinity
```

gens()

Return the generators of `self`.

EXAMPLES:

```
sage: T = TropicalSemiring(QQ)
sage: T.gens()
(1, +infinity)
```

```
>>> from sage.all import *
>>> T = TropicalSemiring(QQ)
>>> T.gens()
(1, +infinity)
```

infinity()

Return the (tropical) additive identity element $+\infty$.

EXAMPLES:

```
sage: T = TropicalSemiring(QQ)
sage: T.zero()
+infinity
```

```
>>> from sage.all import *
>>> T = TropicalSemiring(QQ)
>>> T.zero()
+infinity
```

multiplicative_identity()

Return the (tropical) multiplicative identity element 0.

EXAMPLES:

```
sage: T = TropicalSemiring(QQ)
sage: T.one()
0
```

```
>>> from sage.all import *
>>> T = TropicalSemiring(QQ)
>>> T.one()
0
```

one()

Return the (tropical) multiplicative identity element 0.

EXAMPLES:

```
sage: T = TropicalSemiring(QQ)
sage: T.one()
0
```

```
>>> from sage.all import *
>>> T = TropicalSemiring(QQ)
>>> T.one()
0
```

zero()

Return the (tropical) additive identity element $+\infty$.

EXAMPLES:

```
sage: T = TropicalSemiring(QQ)
sage: T.zero()
+infinity
```

```
>>> from sage.all import *
>>> T = TropicalSemiring(QQ)
>>> T.zero()
+infinity
```

class sage.rings.semirings.tropical_semiring.**TropicalSemiringElement**

Bases: Element

An element in the tropical semiring over an ordered additive semigroup R. Either in R or ∞. The operators $+, \cdot$ are defined as the tropical operators $\oplus, \odot$ respectively.

lift()

Return the value of self lifted to the base.

EXAMPLES:

```
sage: T = TropicalSemiring(QQ)
sage: elt = T(2)
sage: elt.lift()
2
sage: elt.lift().parent() is QQ
True
sage: T.additive_identity().lift().parent()
The Infinity Ring
```

```
>>> from sage.all import *
>>> T = TropicalSemiring(QQ)
>>> elt = T(Integer(2))
>>> elt.lift()
2
>>> elt.lift().parent() is QQ
True
>>> T.additive_identity().lift().parent()
The Infinity Ring
```

multiplicative_order()

Return the multiplicative order of self.

EXAMPLES:

```
sage: T = TropicalSemiring(QQ)
sage: T.multiplicative_identity().multiplicative_order()
1
sage: T.additive_identity().multiplicative_order()
+Infinity
```

```
>>> from sage.all import *
>>> T = TropicalSemiring(QQ)
>>> T.multiplicative_identity().multiplicative_order()
1
>>> T.additive_identity().multiplicative_order()
+Infinity
```

class sage.rings.semirings.tropical_semiring.**TropicalToTropical**

Bases: Map

Map from the tropical semiring to itself (possibly with different bases). Used in coercion.

CHAPTER

THREE

INDICES AND TABLES

- Index
- Module Index
- Search Page

PYTHON MODULE INDEX

r

INDEX