Contents

Chapter 1

Classes

†

1.1 ring – for ring object

- **Classes**
  - **Ring**
  - **CommutativeRing**
  - **Field**
  - **QuotientField**
  - **RingElement**
  - **CommutativeRingElement**
  - **FieldElement**
  - **QuotientFieldElement**
  - **Ideal**
  - **ResidueClassRing**
  - **ResidueClass**
  - **CommutativeRingProperties**
- **Functions**
  - **getRingInstance**
  - **getRing**
  - **inverse**
  - **exact_division**

1.1.1 †Ring – abstract ring

Ring is an abstract class which expresses that the derived classes are (in mathematical meaning) rings.

Definition of ring (in mathematical meaning) is as follows: Ring is a structure with addition and multiplication. It is an abelian group with addition, and a monoid with multiplication. The multiplication obeys the distributive law.

This class is abstract and cannot be instantiated.

Attributes

zero additive unit

one multiplicative unit

Operations

operator	explanation
`A==B`	Return whether M and N are equal or not.

Methods

1.1.1.1 createElement – create an element

createElement(`self`, `seed`: ***(undefined)***) → ***RingElement***

Return an element of the ring with seed.

This is an abstract method.

1.1.1.2 getCharacteristic – characteristic as ring

getCharacteristic(`self`) → ***integer***

Return the characteristic of the ring.

The Characteristic of a ring is the smallest positive integer n s.t. $na = 0$ for any element a of the ring, or 0 if there is no such natural number.
This is an abstract method.

1.1.1.3 issubring – check subring

issubring(`self`, `other`: ***RingElement***) → ***True/False***

Report whether another ring contains the ring as a subring.

This is an abstract method.

1.1.1.4 issuperring – check superring

issuperring(`self`, `other`: ***RingElement***) → ***True/False***

Report whether the ring is a superring of another ring.

This is an abstract method.

1.1.1.5 getCommonSuperring – get common ring

getCommonSuperring(`self`, `other`: ***RingElement***) → ***RingElement***

Return common super ring of self and another ring.

This method uses **issubring**, **issuperring**.

1.1.2 †CommutativeRing – abstract commutative ring

CommutativeRing is an abstract subclass of **Ring** whose multiplication is commutative.

CommutativeRing is subclass of **Ring**.
There are some properties of commutative rings, algorithms should be chosen accordingly. To express such properties, there is a class **CommutativeRingProperties**.

This class is abstract and cannot be instantiated.

Attributes

properties an instance of **CommutativeRingProperties**

Methods

1.1.2.1 getQuotientField – create quotient field

getQuotientField(`self`) → ***QuotientField***

Return the quotient field of the ring.

This is an abstract method.
If quotient field of `self` is not available, it should raise exception.

1.1.2.2 isdomain – check domain

isdomain(`self`) → ***True/False/None***

Return True if the ring is actually a domain, False if not, or None if uncertain.

1.1.2.3 isnoetherian – check Noetherian domain

isnoetherian(`self`) → ***True/False/None***

Return True if the ring is actually a Noetherian domain, False if not, or None if uncertain.

1.1.2.4 isufd – check UFD

isufd(`self`) → ***True/False/None***

Return True if the ring is actually a unique factorization domain (UFD), False if not, or None if uncertain.

1.1.2.5 ispid – check PID

ispid(`self`) → ***True/False/None***

Return True if the ring is actually a principal ideal domain (PID), False if not, or None if uncertain.

1.1.2.6 iseuclidean – check Euclidean domain

iseuclidean(`self`) → ***True/False/None***

Return True if the ring is actually a Euclidean domain, False if not, or None if uncertain.

1.1.2.7 isfield – check field

isfield(`self`) → ***True/False/None***

Return True if the ring is actually a field, False if not, or None if uncertain.

1.1.2.8 registerModuleAction – register action as ring

registerModuleAction(`self`, `action_ring`: ***RingElement***, `action`: ***function***)
→ ***(None)***

Register a ring `action_ring`, which act on the ring through `action` so the ring be an `action_ring` module.

See **hasaction**, **getaction**.

1.1.2.9 hasaction – check if the action has

hasaction(`self`, `action_ring`: ***RingElement***) → ***True/False***

Return True if `action_ring` is registered to provide action.

See **registerModuleAction**, **getaction**.

1.1.2.10 getaction – get the registered action

hasaction(`self`, `action_ring`: ***RingElement***) → ***function***

Return the registered action for `action_ring`.

See **registerModuleAction**, **hasaction**.

1.1.3 †Field – abstract field

Field is an abstract class which expresses that the derived classes are (in mathematical meaning) fields, i.e., a commutative ring whose multiplicative monoid is a group.

Field is subclass of **CommutativeRing**. **getQuotientField** and **isfield** are not abstract (trivial methods).

This class is abstract and cannot be instantiated.

Methods

1.1.3.1 gcd – gcd

gcd(`self`, `a`: ***FieldElement***, `b`: ***FieldElement***) → ***FieldElement***

Return the greatest common divisor of `a` and `b`.

A field is trivially a UFD and should provide gcd. If we can implement an algorithm for computing gcd in an Euclidean domain, we should provide the method corresponding to the algorithm.

1.1.4 †QuotientField – abstract quotient field

QuotientField is an abstract class which expresses that the derived classes are (in mathematical meaning) quotient fields.

`self` is the quotient field of `domain`.
QuotientField is subclass of **Field**.
In the initialize step, it registers trivial action named as baseaction; i.e. it expresses that an element of a domain acts an element of the quotient field by using the multiplication in the domain.

This class is abstract and cannot be instantiated.

Attributes

basedomain domain which generates the quotient field `self`

1.1.5 †RingElement – abstract element of ring

RingElement is an abstract class for elements of rings.

This class is abstract and cannot be instantiated.

Operations

operator	explanation
`A==B`	equality (abstract)

Methods

1.1.5.1 getRing – getRing

getRing(`self`) → *Ring*

Return an object of a subclass of Ring, to which the element belongs.

This is an abstract method.

1.1.6 †CommutativeRingElement – abstract element of commutative ring

CommutativeRingElement is an abstract class for elements of commutative rings.

This class is abstract and cannot be instantiated.

Methods

1.1.6.1 mul_module_action – apply a module action

mul_module_action(`self`, `other`: ***RingElement***) → ***(undefined)***

Return the result of a module action. other must be in one of the action rings of self's ring.

This method uses **getRing**, **CommutativeRing**getaction. We should consider that the method is abstract.

1.1.6.2 exact_division – division exactly

exact_division(`self`, `other`: ***CommutativeRingElement***)
→ ***CommutativeRingElement***

In UFD, if `other` divides `self`, return the quotient as a UFD element.

The main difference with / is that / may return the quotient as an element of quotient field.
Simple cases:

- in a Euclidean domain, if remainder of euclidean division is zero, the division // is exact.
- in a field, there's no difference with /.

If `other` doesn't divide self, raise ValueError. Though __divmod__ can be used automatically, we should consider that the method is abstract.

1.1.7 †FieldElement – abstract element of field

FieldElement is an abstract class for elements of fields.

FieldElement is subclass of **CommutativeRingElement**. **exact_division** are not abstract (trivial methods).

This class is abstract and cannot be instantiated.

1.1.8 †QuotientFieldElement – abstract element of quotient field

QuotientFieldElement class is an abstract class to be used as a super class of concrete quotient field element classes.

QuotientFieldElement is subclass of **FieldElement**.
`self` expresses $\frac{\texttt{numerator}}{\texttt{denominator}}$ in the quotient field.

This class is abstract and should not be instantiated.
`denominator` should not be 0.

Attributes

numerator numerator of `self`

denominator denominator of `self`

Operations

operator	explanation
`A+B`	addition
`A-B`	subtraction
`A*B`	multiplication
`A**B`	powering
`A/B`	division
`-A`	sign reversion (additive inversion)
`inverse(A)`	multiplicative inversion
`A==B`	equality

1.1.9 †Ideal – abstract ideal

Ideal class is an abstract class to represent the finitely generated ideals.

†Because the finitely-generatedness is not a restriction for Noetherian rings and in the most cases only Noetherian rings are used, it is general enough.

This class is abstract and should not be instantiated.
`generators` must be an element of the `aring` or a list of elements of the `aring`. If `generators` is an element of the `aring`, we consider `self` is the principal ideal generated by `generators`.

Attributes

ring the ring belonged to by `self`

generators generators of the ideal `self`

Operations

operator	explanation
`I+J`	addition $\{i + j \mid i \in I,\ j \in J\}$
`I*J`	multiplication $IJ = \{\sum_{i,j} ij \mid i \in I,\ j \in J\}$
`I==J`	equality
`e in I`	For `e` in the ring, to which the ideal `I` belongs.

Methods

1.1.9.1 issubset – check subset

issubset(`self`, `other`: ***Ideal***) → ***True/False***

Report whether another ideal contains this ideal.

We should consider that the method is abstract.

1.1.9.2 issuperset – check superset

issuperset(`self`, `other`: ***Ideal***) → ***True/False***

Report whether this ideal contains another ideal.

We should consider that the method is abstract.

1.1.9.3 reduce – reduction with the ideal

issuperset(`self`, `other`: ***Ideal***) → ***True/False***

Reduce an element with the ideal to simpler representative.

This method is abstract.

1.1.10 †ResidueClassRing – abstract residue class ring

Initialize (Constructor)

ResidueClassRing(`ring`: ***CommutativeRing***, `ideal`: ***Ideal***)
→ **ResidueClassRing**

A residue class ring R/I, where R is a commutative ring and I is its ideal.

ResidueClassRing is subclass of **CommutativeRing**.
one, **zero** are not abstract (trivial methods).
Because we assume that `ring` is Noetherian, so is `ring`.

This class is abstract and should not be instantiated.
`ring` should be an instance of **CommutativeRing**, and `ideal` must be an instance of **Ideal**, whose ring attribute points the same ring with the given ring.

Attributes

ring the base ring R

ideal the ideal I

Operations

operator	explanation
`A==B`	equality
`e in A`	report whether `e` is in the residue ring `self`.

1.1.11 †ResidueClass – abstract an element of residue class ring

Initialize (Constructor)

ResidueClass(`x`: ***CommutativeRingElement***, `ideal`: ***Ideal***)
→ **ResidueClass**

Element of residue class ring $x + I$, where I is the modulus ideal and x is a representative element.

ResidueClass is subclass of **CommutativeRingElement**.

This class is abstract and should not be instantiated.
`ideal` corresponds to the ideal I.

Operations

These operations uses **reduce**.

operator	explanation
`x+y`	addition
`x-y`	subtraction
`x*y`	multiplication
`A==B`	equality

1.1.12 †CommutativeRingProperties – properties for CommutativeRingProperties

Initialize (Constructor)

CommutativeRingProperties((None)) → CommutativeRingProperties

Boolean properties of ring.

Each property can have one of three values; *True*, *False*, or *None*. Of course *True* is true and *False* is false, and *None* means that the property is not set neither directly nor indirectly.
CommutativeRingProperties class treats

- Euclidean (Euclidean domain),
- PID (Principal Ideal Domain),
- UFD (Unique Factorization Domain),
- Noetherian (Noetherian ring (domain)),
- field (Field)

Methods

1.1.12.1 isfield – check field

isfield(`self`) → ***True/False/None***

Return True/False according to the field flag value being set, otherwise return None.

1.1.12.2 setIsfield – set field

isfield(`self`, `value`**:** ***True/False***) → ***(None)***

Set True/False value to the field flag.
Propagation:

- True → euclidean

1.1.12.3 iseuclidean – check euclidean

iseuclidean(`self`) → ***True/False/None***

Return True/False according to the euclidean flag value being set, otherwise return None.

1.1.12.4 setIseuclidean – set euclidean

isfield(`self`, `value`**:** ***True/False***) → ***(None)***

Set True/False value to the euclidean flag.
Propagation:

- True → PID
- False → field

1.1.12.5 ispid – check PID

ispid(`self`) → ***True/False/None***

Return True/False according to the PID flag value being set, otherwise return None.

1.1.12.6 setIspid – set PID

ispid(`self`, `value`: ***True/False***) → ***(None)***

Set True/False value to the euclidean flag.
Propagation:

- True → UFD, Noetherian

- False → euclidean

1.1.12.7 isufd – check UFD

isufd(`self`) → ***True/False/None***

Return True/False according to the UFD flag value being set, otherwise return None.

1.1.12.8 setIsufd – set UFD

isufd(`self`, `value`: ***True/False***) → ***(None)***

Set True/False value to the UFD flag.
Propagation:

- True → domain

- False → PID

1.1.12.9 isnoetherian – check Noetherian

isnoetherian(`self`) → ***True/False/None***

Return True/False according to the Noetherian flag value being set, otherwise return None.

1.1.12.10 setIsnoetherian – set Noetherian

isnoetherian(`self`, `value`: ***True/False***) → ***(None)***

Set True/False value to the Noetherian flag.
Propagation:

- True → domain

- False → PID

1.1.12.11 isdomain – check domain

isdomain(`self`) → ***True/False/None***

Return True/False according to the domain flag value being set, otherwise return None.

1.1.12.12 setIsdomain – set domain

isdomain(`self`, `value`: ***True/False***) → ***(None)***

Set True/False value to the domain flag.
Propagation:

- False $\rightarrow$ UFD, Noetherian

1.1.13 getRingInstance(function)

getRingInstance(obj: ***RingElement***) → ***RingElement***

Return a RingElement instance which equals `obj`.

Mainly for python built-in objects such as int or float.

1.1.14 getRing(function)

getRing(obj: ***RingElement***) → ***Ring***

Return a ring to which `obj` belongs.

Mainly for python built-in objects such as int or float.

1.1.15 inverse(function)

inverse(obj: ***CommutativeRingElement***) → ***QuotientFieldElement***

Return the inverse of `obj`. The inverse can be in the quotient field, if the `obj` is an element of non-field domain.

Mainly for python built-in objects such as int or float.

1.1.16 exact_division(function)

exact_division(self: ***RingElement***, other: ***RingElement***)
→ ***RingElement***

Return the division of `self` by `other` if the division is exact.
Mainly for python built-in objects such as int or float.

Examples

```
>>> print ring.getRing(3)
Z
```

```
>>> print ring.exact_division(6, 3)
2L
```

Bibliography