Contents

Chapter 1

Classes

1.1 poly.formalsum – 形式和

- **Classes**
  - †**FormalSumContainerInterface**
  - **DictFormalSum**
  - †**ListFormalSum**

形式和とは数学的な項の有限和で, 項は二つの部分から成る:係数と基数. 形式和での全ての係数は共通の環に属し, 一方で基数は任意.

二つの形式和は次に示す方法で足される. もし基数が共通である項があれば, それらは同じ基数と加えられた係数を持つ新しい項にまとめられる.

係数は基数より参照することができる. もし特定の基数が形式和に現れない場合, それは null を返す.

便宜上,`terminit` として次を参照:

terminit :
terminit は dict の初期化の型の一つを意味する. それにより構成された辞書は基数から係数への写像として考えられる.

Note for beginner **DictFormalSum** のみ使うことが必要となるかもしれないが, インターフェース (全てのメソッドの名前と意味付け) はその内で定義されているので **FormalSumContainerInterface** の説明を読まなければならないかもしれない.

1.1.1 FormalSumContainerInterface – インターフェースクラス

Initialize (Constructor)

インターフェースは抽象的なクラスなので, インスタンスは作らない.

インターフェースは“形式和”は何かということを定義している. 派生クラスには以下に示す演算とメソッドを定義しなければならない.

Operations

operator	explanation
`f + g`	和
`f - g`	差
`-f`	符号の変更
`+f`	新しいコピー
`f * a, a * f`	スカラー `a` 倍
`f == g`	等しいかどうか返す
`f != g`	等しくないかどうか返す
`f[b]`	基数 `b` に対応した係数を返す
`b in f`	基数 `b` が `f` に含まれているかどうか返す
`len(f)`	項の数
`hash(f)`	ハッシュ

Methods

1.1.1.1 construct_with_default – コピーを構成

construct_with_default(self, maindata: *terminit*) → *FormalSumContainerInterface*

maindata のみ与えられた (必要なら self が持つ情報を使用),self と同じクラスの新しい形式和を作成.

1.1.1.2 iterterms – 項のイテレータ

iterterms(self) → *iterator*

項のイテレータを返す.
イテレータより生成されたそれぞれの項は (base, coefficient) という組.

1.1.1.3 itercoefficients – 係数のイテレータ

itercoefficients(self) → *iterator*

係数のイテレータを返す.

1.1.1.4 iterbases – 基数のイテレータ

iterbases(self) → *iterator*

基数のイテレータを返す.

1.1.1.5 terms – 項のリスト

terms(self) → *list*

項のリストを返す.
返されるリストのそれぞれの項は (base, coefficient) という組.

1.1.1.6 coefficients – 係数のリスト

coefficients(self) → *list*

係数のリストを返す.

1.1.1.7 bases – 基数のリスト

bases(self) → ***list***

基数のリストを返す.

1.1.1.8 terms_map – 項に写像を施す

terms_map(self, func: ***function*****)** → ***FormalSumContainerInterface***

項に写像を施す, すなわち, それぞれの項に func を適用することにより新しい形式和を作成.
funcbase と coefficient という二つのパラメータをとらなければならず, その後新しい項の組を返す.

1.1.1.9 coefficients_map – 係数に写像を施す

coefficients_map(self, func: ***function*****)** → ***FormalSumContainerInterface***

係数に写像を施す, すなわち, 各係数に func を適用することにより新しい形式和を作成.

func は coefficient という一つのパラメータをとり, その後新しい係数を返す.

1.1.1.10 bases_map – 基数に写像を施す

bases_map(self, func: ***function*****)** → ***FormalSumContainerInterface***

基数に写像を施す, すなわち, 各基数に func を適用することにより新しい形式和を作成.

func は base という一つのパラメータをとり, その後新しい基数を返す.

1.1.2 DictFormalSum – 辞書で実装された形式和

`dict` を基に実装された形式和.

このクラスは **FormalSumContainerInterface** を継承. インターフェースの全てのメソッドは実装される.

Initialize (Constructor)

DictFormalSum(args: *terminit*, defaultvalue: *RingElement*=None)
→ *DictFormalSum*

args の型については **terminit** を参照. 基数から係数への写像を作る.
任意引数 defaultvalue は`__getitem__`への初期設定値, すなわち, もし指定の基数に関する項がなかったら検索を試み defaultvalue を返す. 従ってそれは他の係数が所属している環の元である.

1.1.3 ListFormalSum – リストで実装された形式和

リストを基に実装された形式和.

FormalSumContainerInterface を継承. インターフェースの全てのメソッドは実装される.

Initialize (Constructor)

ListFormalSum(args: *terminit*, defaultvalue: *RingElement*=None)
→ *ListFormalSum*

args の型については **terminit** を参照. 基数から係数への写像を作る.
任意引数 defaultvalue は`__getitem__`への初期設定値, すなわち, もし指定の基数に関する項がなかったら, 検索を試み defaultvalue を返す. 従ってそれは他の係数が所属している環の元である.

Bibliography