

Contents

1	Classes	2
1.1	poly.formalsum – 形式和	2
1.1.1	FormalSumContainerInterface – インターフェースクラス	3
1.1.1.1	construct_with_default – コピーを構成	4
1.1.1.2	iterterms – 項のイテレータ	4
1.1.1.3	itercoefficients – 係数のイテレータ	4
1.1.1.4	iterbases – 基数のイテレータ	4
1.1.1.5	terms – 項のリスト	4
1.1.1.6	coefficients – 係数のリスト	4
1.1.1.7	bases – 基数のリスト	5
1.1.1.8	terms_map – 項のリストを返す	5
1.1.1.9	coefficients_map – 係数のリスト	5
1.1.1.10	bases_map – 基数のリスト	5
1.1.2	DictFormalSum – 辞書で実装された形式和	6
1.1.3	ListFormalSum – リストで実装された形式和	6

Chapter 1

Classes

1.1 poly.formalsum – 形式和

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

形式和とは数学的に有限な項の合計で、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のもので、一方で基数は無作為.

二つの形式和は次に示す方法で足される. もし基数とともに項があれば、それらは同じ基数と係数が与えられた新しい項に結合される.

係数は基数より調べることができる. もし特定の基数が形式和で現れない場合、それは null.

terminit として次に利便性を参照:

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だろう.

Note for beginner **DictFormalSum** のみ使うことが必要となるだろうが、インターフェース (全てのメソッドの名前と意味論) はその内で定義されているので **FormalSumContainerInterface** の説明を読む必要はないだろう.

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、例示化はしない。

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
$-f$	否定
$+f$	新しいコピー
$f * a, a * f$	スカラー a 倍
$f == g$	等式
$f != g$	不等式
$f[b]$	基底 b に対応した係数を返す
$b \text{ in } f$	基底 b が f に入っているかどうか返す
$\text{len}(f)$	項の数
$\text{hash}(f)$	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, 必要なら `self` のコピーを使用した, `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.

イテレータより生み出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 – 項のリストを返す

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

項上の写像, 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ない, その後新しい項の組を返す.

1.1.1.9 coefficients_map – 係数のリスト

`coefficients_map(self) → FormalSumContainerInterface`

係数の写像, 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 – 基数のリスト

`bases_map(self) → FormalSumContainerInterface`

基数の写像, 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, そして新しい基数を返す.

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.
任意の引数 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要素となる.

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証される.

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.
任意の独立変数 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要素となる.