

Contents

1	Functions	2
1.1	poly.groebner – グレブナー基底	2
1.1.1	buchberger – グレブナー基底を得るための素朴なアルゴリズム	2
1.1.2	normal_strategy – グレブナー基底を得る普通のアルゴリズム	2
1.1.3	reduce_groebner – 簡約グレブナー基底	3
1.1.4	s_polynomial – S-polynomial	3

Chapter 1

Functions

1.1 poly.groebner – グレブナー基底

groebner モジュールは多変数多項式イデアルに対するグレブナー基底を計算するためのもの.

このモジュールは以下に示す型を使用:

polynomial :

polynomial は関数 **polynomial** によって生み出された多項式.

order :

order は多項式の項順序.

1.1.1 buchberger – グレブナー基底を得るための素朴なアルゴリズム

buchberger(generating: *list*, order: *order*) → [*polynomials*]

order についての与えられた多項式の生成集合により生成されるイデアルのグレブナー基底を返す.

この実装は非常に素朴なものだということに注意.

引数 generating は **Polynomial** のリスト; 引数 order は項順序.

1.1.2 normal_strategy – グレブナー基底を得る普通のアルゴリズム

normal_strategy(generating: *list*, order: *order*) → [*polynomials*]

order についての与えられた多項式の生成集合により生成されるイデアルのグレブナー基底を返す.

この関数は ‘普通の戦略’ を使用.

引数 `generating` は **Polynomial** のリスト; 引数 `order` は項順序.

1.1.3 `reduce_groebner` – 簡約グレブナー基底

`reduce_groebner(gbasis: list, order: order) → [polynomials]`

グレブナー基底から構成された簡約グレブナー基底を返す.

出力は以下を満たす:

- $\text{lb}(f)$ は $\text{lb}(g)$ を割り切る $\Rightarrow g$ は簡約グレブナー基底ではない.
- モニック多項式.

引数 `gbasis` は多項式のリストで, (単に生成集合であるだけでなく) グレブナー基底.

1.1.4 `s_polynomial` – S-polynomial

`s_polynomial(f: polynomial, g: polynomial, order: order)`
 $\rightarrow [polynomials]$

`order` についての f と g の S-多項式を返す.

$$S(f, g) = (\text{lc}(g) * T / \text{lb}(f)) * f - (\text{lc}(f) * T / \text{lb}(g)) * g,$$

$$T = \text{lcm}(\text{lb}(f), \text{lb}(g)).$$

Examples

```
>>> f = multiutil.polynomial({(1,0):2, (1,1):1},rational.theRationalField, 2)
>>> g = multiutil.polynomial({(0,1):-2, (1,1):1},rational.theRationalField, 2)
>>> lex = termorder.lexicographic_order
>>> groebner.s_polynomial(f, g, lex)
UniqueFactorizationDomainPolynomial({(1, 0): 2, (0, 1): 2})
>>> gb = groebner.normal_strategy([f, g], lex)
>>> for gb_poly in gb:
...     print gb_poly
...
UniqueFactorizationDomainPolynomial({(1, 1): 1, (1, 0): 2})
UniqueFactorizationDomainPolynomial({(1, 1): 1, (0, 1): -2})
```

```

UniqueFactorizationDomainPolynomial({(1, 0): 2, (0, 1): 2})
UniqueFactorizationDomainPolynomial({(0, 2): -2, (0, 1): -4.0})
>>> gb_red = groebner.reduce_groebner(gb, lex)
>>> for gb_poly in gb_red:
...     print gb_poly
...
UniqueFactorizationDomainPolynomial({(1, 0): Rational(1, 1), (0, 1): Rational(1, 1)})
UniqueFactorizationDomainPolynomial({(0, 2): Rational(1, 1), (0, 1): 2.0})

```

Bibliography