# Contents

# Chapter 1

# Functions

## 1.1 poly.groebner – Gröbner Basis

The groebner module is for computing Gröbner bases for multivariate polynomial ideals.

This module uses the following types:

**polynomial** :
`polynomial` is the polynomial generated by function **polynomial**.

**order** :
`order` is the order on terms of polynomials.

### 1.1.1 buchberger – naïve algorithm for obtaining Gröbner basis

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

Return a Gröbner basis of the ideal generated by given generating set of polynomials with respect to the `order`.

Be careful, this implementation is very naive.

The argument `generating` is a list of **Polynomial**; the argument `order` is an order.

### 1.1.2 normal_strategy – normal algorithm for obtaining Gröbner basis

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

Return a Gröbner basis of the ideal generated by given generating set of polynomials with respect to the `order`.

This function uses the 'normal strategy'.

The argument `generating` is a list of **Polynomial**; the argument `order` is an order.

### 1.1.3 reduce_groebner – reduce Gröbner basis

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

Return the reduced Gröbner basis constructed from a Gröbner basis.

The output satisfies that:

- $\mathrm{lb}(f)$ divides $\mathrm{lb}(g) \Rightarrow g$ is not in reduced Gröbner basis, and
- monic.

The argument `gbasis` is a list of polynomials, a Gröbner basis (not merely a generating set).

### 1.1.4 s_polynomial – S-polynomial

**s_polynomial**(`f`: ***polynomial***, `g`: ***polynomial***, `order`: ***order***)
→ ***[polynomials]***

Return S-polynomial of `f` and `g` with respect to the `order`.

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

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

#### 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})
```