Contents

Chapter 1

Classes

1.1 poly.termorder – term orders

- **Classes**
  - †**TermOrderInterface**
  - †**UnivarTermOrder**
  - **MultivarTermOrder**
- **Functions**
  - **weight_order**

1.1.1 TermOrderInterface – interface of term order

Initialize (Constructor)

TermOrderInterface(`comparator`: ***function***) → ***TermOrderInterface***

A term order is primarily a function, which determines precedence between two terms (or monomials). By the precedence, all terms are ordered.

More precisely in terms of `Python` , a term order accepts two tuples of integers, each of which represents power indices of the term, and returns 0, 1 or -1 just like cmp built-in function.

A `TermOrder` object provides not only the precedence function, but also a method to format a string for a polynomial, to tell degree, leading coefficients, etc.

`comparator` accepts two tuple-like objects of integers, each of which represents power indices of the term, and returns 0, 1 or -1 just like `cmp` built-in function.

This class is abstract and should not be instantiated. The methods below have to be overridden.

Methods

1.1.1.1 cmp

cmp(`self`, `left`: *tuple*, `right`: *tuple*) → *integer*

Compare two index tuples `left` and `right` and determine precedence.

1.1.1.2 format

format(`self`, `polynom`: *polynomial*, **`keywords`: *dict*)
→ *string*

Return the formatted string of the polynomial `polynom`.

1.1.1.3 leading_coefficient

leading_coefficient(`self`, `polynom`: *polynomial*) → *CommutativeRingElement*

Return the leading coefficient of polynomial `polynom` with respect to the term order.

1.1.1.4 leading_term

leading_term(`self`, `polynom`: *polynomial*) → *tuple*

Return the leading term of polynomial `polynom` as tuple of `(degree index, coefficient)` with respect to the term order.

1.1.2 UnivarTermOrder – term order for univariate polynomials

Initialize (Constructor)

UnivarTermOrder(`comparator`: *function*) → *UnivarTermOrder*

There is one unique term order for univariate polynomials. It's known as degree.

One thing special to univariate case is that powers are not tuples but bare integers. According to the fact, method signatures also need be translated from the definitions in TermOrderInterface, but its easy, and we omit some explanations.

`comparator` can be any callable that accepts two integers and returns 0, 1 or -1 just like `cmp`, i.e. if they are equal it returns 0, first one is greater 1, and otherwise -1. Theoretically acceptable comparator is only the `cmp` function.

This class inherits **TermOrderInterface**.

Methods

1.1.2.1 format

format(`self`, `polynom`: ***polynomial***, `varname`: ***string***=**'X'**, `reverse`: ***bool***=**False**)
→ ***string***

Return the formatted string of the polynomial `polynom`.

- `polynom` must be a univariate polynomial.
- `varname` can be set to the name of the variable.
- `reverse` can be either `True` or `False`. If it's `True`, terms appear in reverse (descending) order.

1.1.2.2 degree

degree(`self`, `polynom`: ***polynomial***) → ***integer***

Return the degree of the polynomial `polynom`.

1.1.2.3 tail_degree

tail_degree(`self`, `polynom`: ***polynomial***) → ***integer***

Return the least degree among all terms of the `polynom`.

This method is *experimental*.

1.1.3 MultivarTermOrder – term order for multivariate polynomials

Initialize (Constructor)

MultivarTermOrder(`comparator`: ***function***) → ***MultivarTermOrder***

This class inherits **TermOrderInterface**.

Methods

1.1.3.1 format

format(`self`, `polynom`: *polynomial*, `varname`: *tuple*=**None**, `reverse`: *bool*=**False**, **`kwds`: *dict*)
→ *string*

Return the formatted string of the polynomial `polynom`.

An additional argument `varnames` is required to name variables.

- `polynom` is a multivariate polynomial.
- `varnames` is the sequence of the variable names.
- `reverse` can be either `True` or `False`. If it's `True`, terms appear in reverse (descending) order.

1.1.4 weight_order – weight order

weight_order(`weight`: *sequence*, `tie_breaker`: *function*=**None**)
→ *function*

Return a comparator of weight ordering by `weight`.

Let w denote the `weight`. The weight ordering is defined for arguments x and y that $x < y$ if $w \cdot x < w \cdot y$ or $w \cdot x == w \cdot y$ and tie breaker tells $x < y$.

The option `tie_breaker` is another comparator that will be used if dot products with the weight vector leaves arguments tie. If the option is `None` (default) and a tie breaker is indeed necessary to order given arguments, a `TypeError` is raised.

Examples

```
>>> w = termorder.MultivarTermOrder(
...     termorder.weight_order((6, 3, 1), cmp))
>>> w.cmp((1, 0, 0), (0, 1, 2))
1
```