16.30 IDEALS: Arithmetic for polynomial ideals

This package implements the basic arithmetic for polynomial ideals by exploiting the Gröbner bases package of REDUCE. In order to save computing time all intermediate Gröbner bases are stored internally such that time consuming repetitions are inhibited.

Author: Herbert Melenk.

16.30.1 Introduction

This package implements the basic arithmetic for polynomial ideals by exploiting the Gröbner bases package of REDUCE. In order to save computing time all intermediate Gröbner bases are stored internally such that time consuming repetitions are inhibited. A uniform setting facilitates the access.

16.30.2 Initialization

Prior to any computation the set of variables has to be declared by calling the operator $I\_setting$. E.g. in order to initiate computations in the polynomial ring $Q[x,y,z]$ call

```
    I_setting(x,y,z);
```

A subsequent call to $I\_setting$ allows one to select another set of variables; at the same time the internal data structures are cleared in order to free memory resources.

16.30.3 Bases

An ideal is represented by a basis (set of polynomials) tagged with the symbol I, e.g.

```
   u := I(x*z-y**2, x**3-y*z);
```

Alternatively a list of polynomials can be used as input basis; however, all arithmetic results will be presented in the above form. The operator $ideal2list$ allows one to convert an ideal basis into a conventional REDUCE list.

Operators

Because of syntactical restrictions in REDUCE, special operators have to be used for ideal arithmetic:

```
.+            ideal sum (infix)
.*            ideal product (infix)
.:            ideal quotient (infix)
./            ideal quotient (infix)
.=            ideal equality test (infix)
subset        ideal inclusion test (infix)
intersection  ideal intersection (prefix,binary)
member        test for membership in an ideal
                (infix: polynomial and ideal)
gb            Groebner basis of an ideal (prefix, unary)
ideal2list    convert ideal basis to polynomial list
                (prefix,unary)
```

Example:

```
 I(x+y,x^2) .* I(x-z);

   2                      2    2
I(X  + X*Y - X*Z - Y*Z,X*Y  - Y *Z)
```

The test operators return the values 1 (=true) or 0 (=false) such that they can be used in REDUCE $if - then - else$ statements directly.

The results of $sum, product, quotient, intersction$ are ideals represented by their Gröbner basis in the current setting and term order. The term order can be modified using the operator $torder$ from the Gröbner package. Note that ideal equality cannot be tested with the REDUCE equal sign:

```
I(x,y)  = I(y,x)       is false
I(x,y) .= I(y,x)       is true
```

16.30.4 Algorithms

The operators $groebner$, $preduce$ and $idealquotient$ of the REDUCE Gröbner package support the basic algorithms:

$GB(Iu_1, u_2...) \rightarrow groebner(\{u_1, u_2...\}, \{x, ...\})$

$p \in I_1 \rightarrow p = 0\, mod\, I_1$

$I_1 : I(p) \rightarrow (I_1 \bigcap I(p))/p\, elementwise$

On top of these the Ideals package implements the following operations:

$I(u_1, u_2...) + I(v_1, v_2...) \rightarrow GB(I(u_1, u_2..., v_1, v_2...))$

$I(u_1, u_2...) * I(v_1, v_2...) \rightarrow GB(I(u_1 * v_1, u_1 * v2, ..., u_2 * v_1, u_2 * v_2...))$

$I_1 \bigcap I_2 \rightarrow Q[x, ...] \bigcap GB_{lex}(t * I_1 + (1 - t) * I_2, \{t, x, ..\})$

$I_1 : I(p_1, p_2, ...) \rightarrow I_1 : I(p_1) \bigcap I_1 : I(p_2) \bigcap ...$

$I_1 = I_2 \rightarrow GB(I_1) = GB(I_2)$

$I_1 \subseteq I_2 \rightarrow \ u_i \in I_2 \, \forall \, u_i \in I_1 = I(u_1, u_2...)$

16.30.5 Examples

Please consult the file `ideals.tst`.