Gb – A Mathematica Interface to Faugère's Gb Library

by Manuel Kauers and Burkhard Zimmermann

Abstract

This is the user manual of a free Mathematica package which provides an interface to the C++ library Gb of Jean-Charles Faugère[1] for fast Gröbner basis computation.

The interface has been developed for Mathematica 5.0, but it should also work under the later Mathematica 4.∗ versions. It will definitely not work for Mathematica 3.∗.

Contents

1 A Quick Start

We start out by showing the package in action. A full description of the functionality will be given in later sections.

1.1 Loading the Package

In[1]:= << Gb.m

Gb Library by Jean-Charles Faugère – © 2003
http://fgbrs.lip6.fr/jcf/
Gb Interface to Mathematica by Manuel Kauers and Burkhard Zimmermann – RISC Linz – V 0.7 (04-01-22)

[1] available for academic people and non commercial use at http://fgbrs.lip6.fr/jcf/index.html

1.2 Computing A Gröbner Basis

One way of computing a Gröbner basis by Gb is analogous to the usage of the Mathematica function GroebnerBasis.

In[2]:= $\text{c4base} = \{a + b + c + d, ab + bc + cd + da, abc + bcd + cda + dab, abcd - 1\};$

In[3]:= $\text{gb} = \text{GBasis}[\text{c4base}, \{a, b, c, d\}, \text{MonomialOrder} \to \text{DegreeReverseLexicographic}]$

Out[3]= $\{bc - bd + cd - 2d^2 + c^2d^4, -c - d + c^3d^2 + c^2d^3, -b - d + bd^4 + d^5, -1 + bcd^2 + c^2d^2 - bd^3 + cd^3 - d^4, bc^2 + c^2d - bd^2 - d^3, b^2 + 2bd + d^2, a + b + c + d\}$

1.3 The Type "Ideal"

For more sophisticated functions the usage of Ideal objects is preferable. Ideals are defined by the same data as Gröbner bases.

In[4]:= $\text{c4} = \text{GbIdeal}[\text{c4base}, \{a, b, c, d\}, \text{MonomialOrder} \to \text{DegreeReverseLexicographic}]$

Out[4]= $\langle bc - bd + cd - 2d^2 + c^2d^4, -c - d + c^3d^2 + c^2d^3, -b - d + bd^4 + d^5, -1 + bcd^2 + c^2d^2 - bd^3 + cd^3 - d^4, bc^2 + c^2d - bd^2 - d^3, b^2 + 2bd + d^2, a + b + c + d\rangle$

In[5]:= Head[%]

Out[5]= Ideal

Various sorts of information can be extracted from an ideal object.

In[6]:= GBasis[c4]

Out[6]= $\{bc - bd + cd - 2d^2 + c^2d^4, -c - d + c^3d^2 + c^2d^3, -b - d + bd^4 + d^5, -1 + bcd^2 + c^2d^2 - bd^3 + cd^3 - d^4, bc^2 + c^2d - bd^2 - d^3, b^2 + 2bd + d^2, a + b + c + d\}$

In[7]:= GbDimension[c4]

Out[7]= 1

In[8]:= GbDegree[c4]

Out[8]= 4

In[9]:= $\text{GbHilbertFunction}[\text{c4}, \text{Variable} \to t]$

Out[9]= $$\frac{1 + 2t + 2t^2 + t^3 - t^4 - t^5}{(1 - t)^3}$$

It is also possible to ask for the time that was needed to compute the ideal object, (i.e., the underlying Gröbner basis). Note that this time is not covered by the Mathematica Timing command because the computations are done in an external process.[2] The computation of c4 was immediate:

In[10]:= GbTime[c4]

Out[10]= 0.Second

[2]Since Mathematica 5, the command AbsoluteTiming is available which computes timings according to the watch time rather than the spent cpu time of Mathematica only.

All the mentioned functions are also available for polynomial systems in place of ideal objects. The arguments are analogous to the arguments of GBasis. Example:

In[11]:= GbDimension[c4base, $\{a, b, c, d\}$, MonomialOrder → DegreeReverseLexicographic]

Out[11]= 1

In[12]:= GbHilbertFunction[c4base, $\{a, b, c, d\}$,
MonomialOrder → DegreeReverseLexicographic,
Variable → t]

Out[12]=
$$\frac{1 + 2t + 2t^2 + t^3 - t^4 - t^5}{(1-t)^3}$$

Note that in this way the Gröbner basis for c4base is recomputed in every step.

2 Installation

Proceed as follows:

1. Download the Gb binaries from http://fgbrs.lip6.fr/jcf/index.html and install it on your computer following the instructions contained in that package. For full functionality, a registration is required.
2. Put the Mathematica interface package file Gb.m into a directory where Mathematica can find it.
3. In Mathematica, set the following variables:
   - GbHome – String specifying the directory in which the Gb binaries can be found, e.g., `"/home/mkauers/systems/Gb"`. The default is `"."`, pointing to the directory in which Mathematica has been started.
   - GbBash – String specifying the command line for calling bash. The default `"bash"` should work on most systems, another appropriate setting might be `"/usr/bin/bash"`. (The bash is required only when the TimeConstraint option is used.)

   It might be convenient to do this definitions in the `init.m` file once and for all.

3 Function Specifications

3.1 The "usual" options

Many functions of the package share the same set of options. These will be refered to as the "usual" options. The usual options are the following:

- MonomialOrder one of the following
  - Lexicographic : pure lexicographic ordering
  - DegreeReverseLexicographic : degree reverse lexicographic (default)
  - Lex : synonym for Lexicographic
  - DRL : synonym for DegreeReverseLexicographic

- MonomialOrdering synonym for MonomialOrder
- Characteristic 0 (for $\mathbb{Q}$, this is default) or a prime p, $1 < p < 65522$ (for $\mathbb{Z}_p$).
- Modulus synomym for Characteristic
- CoefficientDomain one of the following
  - Automatic : determine the ground field from the given characteristic (default).
  - Rationals : compute in $\mathbb{Q}$
  - FiniteField : compute in $\mathbb{Z}_p$

  This option is only for consistency with Mathematica's builtin GroebnerBasis command. In Gb, it does not contribute any information to the computation, but raises an error if specified CoefficientDomain is inconsistent with the given characteristic or unknown.
- Eliminate True or False. This option is only applicable for elimination orderings, i.e., when two lists of variables are given. If True, then polynomials containing variables from the first list will not be contained in the output. Default is False.
- Algorithm string prefix of the serveur name to be used, e.g., `"serveur"` (this is default) or `"accel_serveur"`
- InFileName a string specifying the file into which the input for Gb will be written. If this option is not given (or it points to something which is not a string), a tmp file will be used. (This is default.)
- OutFileName a string specifying the file into which Gb will be asked to write its output. If this option is not given (or it points to something which is not a string), a tmp file will be used. (This is default.)
- TimeConstraint a positive integer number (or Infinity) specifying the maximum number of seconds to be spent for the computation. (This option requires the GbBash variable to be set.) The default is Infinity.

The default values for the above options are mainained as Options[Gb]. The Mathematica functions like SetOptions, applied to Gb, modify the usual options in one go for all functions of the package.

3.2 Public Functions

The following list describes all functions which are offered by this Gb interface.

GBasis — computes a Gröbner basis

GBasis[$\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}$]
GBasis[$\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}, \{y_1, y_2, \ldots\}$]
GBasis[*ideal*]

Description: Computes a Gröbner basis of the given polynomial system, or extracts the Gröbner basis of a given ideal. Two lists of variables specify an elimination ordering in which the variables of the first list are to be eliminated. Any expression whose head is different from Plus, Times, or Power is admitted as variable.

If the computation was not successful for some reason, the value $Failed is returned.

Options: The first two calls admit the usual options.

See also: GbPretend

GbBash — command for executing bash

Description: This variable should evaluate to a string which describes a command line to start a UNIX bash. The default setting `"bash"` will most likely work without modification. An alternative setting might be `"/usr/bin/bash"`.

A bash script is used for killing the external process when a timeout occurs. If no finite TimeConstraint is specified, the bash is not needed.

GbCharacteristic — extracts the characteristic of a given ideal

GbCharacteristic[*ideal*]

Description: Returns the characteristic of the given ideal. The ideal's characteristic is defined by the option Characteristic or Modulus in GBasis or GbPretend

See also: GBasis, GbPretend

GbCofactors — computes cofactors of a polynomial wrt. an ideal

GbCofactors[$poly, \{p_1, p_2, \ldots, \}, \{x_1, x_2, \ldots\}$]

Description: If *poly* belongs to the ideal generated by $p_1, p_2, \ldots$, then, by the definition of ideals, there exist polynomials $q_1, q_2, \ldots$ such that

$$poly = \sum_i p_i q_i$$

The polynomials q_i are called *cofactors* of *poly* wrt. $\{p_1, p_2, \ldots\}$, and they can be computed by means of Gröbner bases.

This function returns a list $\{q_1, q_2, \ldots\}$ of cofactors for *poly* wrt. $\{p_1, p_2, \ldots\}$. (This functionality is not offered by Gb directly, but the function makes use of Gb functions internally.)

If *poly* does not live in the ideal, the function returnes the value $Failed.

Note that this function does not work for elimination orderings, i.e. only one list of variables must be specified.

Options: the usual options

See also: GBasis, GbExtendedGBasis

GbDegree — computes the degree of an ideal

GbDegree[$\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}$]
GbDegree[$\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}, \{y_1, y_2, \ldots\}$]
GbDegree[*ideal*]

Description: Computes the degree of the ideal defined by the given polynomial system, or the given ideal object, respectively. If two lists of variables are given, and the Elimination option is set to True, then the degree of the elimination ideal is returned. Otherwise, an elimination ordering is used for the Gröber basis computation, but the degree of the whole ideal is returned.

If the computation was not successful for some reason, the value $Failed is returned.

Options: the usual options

GbDimension — computes the affine dimension of an ideal

GbDegree[$\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}$]
GbDegree[$\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}, \{y_1, y_2, \ldots\}$]
GbDegree[*ideal*]

Description: Computes the affine dimension of the ideal defined by the given polynomial system, or the given ideal object, respectively. If two lists of variables are given, and the Elimination option is set to True, then the dimension of the elimination ideal is returned. Otherwise, an elimination ordering is used for the Gröber basis computation, but the dimension of the whole ideal is returned.

If the computation was not successful for some reason, the value $Failed is returned.

Options: The first two calls admit the usual options

GbEliminate — compute an elimination ideal

GbEliminate$[\{p_1, p_2, \ldots, \}, \{x_1, x_2, \ldots\}, \{y_1, y_2, \ldots\}]$
GbEliminate$[\mathit{ideal}, \{x_1, x_2, \ldots\}]$

Description: Computes the elimination ideal of $\{p_1, p_2, \ldots\}$ wrt. the first list of variables. The second list of variables specifies which variables should remain. In the second variant, the y_i are determined by GbVariables[*ideal*].

The first variant is synonym to

$$\text{GBasis}[\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}, \{y_1, y_2, \ldots\}, \text{Eliminate} \to \text{True}]$$

If the computation was not successful for some reason, the value $Failed is returned.

Options: the usual options, any value assigned to Eliminate will be overriden to True

See also: GBasis, GbVariables

GbExtendedGBasis — computed a Gröbner basis with cofactors

GbExtendedGBasis$[\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}]$

Description: Computes a list

$$\begin{aligned}\{\ &\{b_1, \{q_{11}, q_{12}, \ldots\}\},\\ &\{b_2, \{q_{21}, q_{22}, \ldots\}\},\\ &\{b_3, \{q_{31}, q_{32}, \ldots\}\},\\ &\ldots\}\end{aligned}$$

such that $\{b_1, b_2, b_3, \ldots\}$ is a Gröbner basis of $\{p_1, p_2, \ldots\}$ and for each i, we have

$$b_i = \sum_j q_{ij} p_i$$

as a representation of b_i in terms of the basis $\{p_1, p_2, \ldots\}$. This means, in addition to the Gröbner basis, this function computes also the cofactors of the resulting Gröbner basis elements.

(This functionality is not directly offered by Gb, but rather part of the interface package.)

Note that is is not possible for this function to specify an elimination ordering, i.e., exactly one list of variables has to be specified.

Options: the usual options

See also: GBasis, GbCofactors

GbFGLM — computes a lexicographic Gröbner basis

GbFGLM[*ideal*]

Description: Applies the FGLM algorithm to the given ideal, and returns an ideal wrt. lexicographic monomial order. The input ideal has to be zero dimensional and and its monomial order has to be degree reverse lexicographic.

The result is of type Ideal, or it is the value $Failed, if something went wrong.

If the computation was not successful for some reason, the value $Failed is returned.

Options: TimeConstraint, InFileName, OutFileName, with the meaning described in the previous section.

See also: GbMonomialOrder, GbDimension

GbHilbertFunction — computes the Hilbert Function of an ideal

GbHilbertFunction[$\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}$]
GbHilbertFunction[$\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}, \{y_1, y_2, \ldots\}$]
GbHilbertFunction[*ideal*]

Description: Computes the Hilbert function of the ideal defined by the given polynomial system, or the given ideal object, respectively. If two lists of variables are given, and the Elimination option is set to True, then the Hilbert function of the elimination ideal is returned. Otherwise, an elimination ordering is used for the Gröber basis computation, but the Hilbert function of the whole ideal is returned.

If the computation was not successful for some reason, the value $Failed is returned.

Options: Variable: expression in which the function should be expressed. The first two calls also admit the usual options.

See also: GbHilbertPolynomial

GbHilbertPolynomial — computes the Hilbert Polynomial of an ideal

GbHilbertPolynomial[$\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}$]
GbHilbertPolynomial[$\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}, \{y_1, y_2, \ldots\}$]
GbHilbertPolynomial[*ideal*]

Description: Computes the Hilbert polynomial of the ideal defined by the given polynomial system, or the given ideal object, respectively. If two lists of variables are given, and the Elimination option is set to True, then the Hilbert polynomial of the elimination ideal is returned. Otherwise, an elimination ordering is used for the Gröber basis computation, but the Hilbert polynomial of the whole ideal is returned.

If the computation was not successful for some reason, the value $Failed is returned.

Options: Variable: expression in which the polynomial should be expressed. The first two calls also admit the usual options.

See also: GbHilbertFunction

GbHome — specifies the Gb directory

Description: This variable should be set such that GbHome `<>` `"/"` `<>` *file* evaluates to a file name of an existing serveur binary, if *file* is a valid serveur name. The default setting `"."` points to the directory from which Mathematica has been started, and will most likely have to be changed.

GbIdeal — constructs an ideal object from a polynomial system

GbIdeal$[\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}]$
GbIdeal$[\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}, \{y_1, y_2, \ldots\}]$

Description: Computes an expression representing the ideal generated by the given polynomial system over the given variables. The ground field ($\mathbb{Q}$ or $\mathbb{Z}_p$ for p prime) and the monomial order (lexicographic or degree reverse lexicographic) are specified by options. If two lists of variables are given, a block order will be used. Each block is then ordered according to the specified ordering. If, furthermore, the option Elimination is switched to True (by default, it is not), then the resulting expression represents the elimination ideal.

By default, ideals are printed in the form $\langle b_1, b_2, \ldots, b_n \rangle$ (angle brackets) where the b_i constitute the ideal's underlying Gröber basis. More advanced users may want to switch off the pretty print option by setting the $PrettyPrintIdeals flag to False.

In order to access the information of an ideal, apply the appropriate extraction function (GBasis, GbDegree, GbDimension, GbHilbertFunction, GbHilbertPolynomial,...) to it.

If the computation was not successful for some reason, the value $Failed is returned.

Options: the usual options

See also: GbPretend

GbMonomialOrder — Extract the monomial order from an ideal object

GbMonomialOrder[*ideal*]

Description: returns the monomial order (either the symbol DRL or the symbol Lex) of the given ideal object. The ideal's monomial order was defined by the option MonomialOrder in GBasis or GbPretend

The monomial order is defined by the return value of this function together with the return value of GbVariables.

See also: GBasis, GbPretend, GbVariables

GbNormalForm — Compute normal forms wrt. ideals

GbNormalForm[*poly*, *ideal*]

Description: Computes the normal form of a polynomial wrt. the given ideal. The polynomial is assumed to be a polynomial expression in the variables of the ideal.

If the computation was not successful for some reason, the value $Failed is returned.

Options: TimeConstraint, InFileName, OutFileName, Algorithm, with the meaning described in the previous section.

GbPretend — Construct an ideal object from the given data

GbIdeal$[\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}]$
GbIdeal$[\{p_1, p_2, \ldots\}, \{x_1, x_2, \ldots\}, \{y_1, y_2, \ldots\}]$

Description: Computes an expression representing the ideal generated by the given polynomial system over the given variables. The ground field ($\mathbb{Q}$ or $\mathbb{Z}_p$ for p prime) and the monomial order (lexicographic or degree reverse lexicographic) are specified by options. If two lists of variables are given, a block order will be used. Each block is then ordered according to the specified ordering. If, furthermore, the option Elimination is switched to True (by default, it is not), then the resulting expression represents the elimination ideal.

This function is exactly the same as GbIdeal, except that it *assumes without checking* that the input polynomial system $\{p_1, p_2, \dots\}$ is already a Gröbner basis. If this is not the case, subsequent applications of functions like GbDegree, GbDimension, etc. may give arbitrary results or errors.

Options: the usual options, where assingments to Algorithm, InFileName, OutFileName, TimeConstraint are ignored.

See also: GBasis

GbTime — returns the time needed to construct an Ideal object

GbTime[*ideal*]

Description: Returns the time spent by Gb on computation of the given ideal object, according to Gb's log files. This is pure computation time; time spent by the interface package in Mathematica as well as time spent for I/O is not taken into account.

The GbTime function should offer an alternative to calls like Timing[GbIdeal[...]] which will return missleading information, because they ignore cpu time which was spent by external processes like the Gb process.

From Mathematica 5 on, there exists a function AbsoluteTiming which offeres another substitute for the Timing function, measuring the absolute time needed to complete the computation. (But this does, of course, also count background processes different from Gb).

GbTriangular — computes a list of triangular sets

GbTriangular[*ideal*]

Description: Computes a list of triangular sets from a given zero dimensional ideal wrt. lexicographic ordering. The function returns a list of lists of polynomials, or the value $Failed if something went wrong.

Options: Algorithm, InFileName, OutFileName, and TimeConstraint from the usual options.

See also: GbFGLM

GbVariables — extracts the list of variables from an ideal object

GbVariables[*ideal*]

Description: Returns the list of variables by which the ideal was computed. In case of an elimination ordering, the function returns a list of two lists of variables, the first list of which contains the "heavier" variables.

The monomial order is defined by the return value of this function together with the return value of GbMonomialOrder

See also: GbMonomialOrder

GbVersion — Prints out the version of Gb

GbVersion[ ]

Description: Prints out information on the version of the underlying Gb system, and exists.

Options: Serveur: string of the serveur (full name) to be used for extracting the information. The default is `"serveur__DMP__Dexp__GF"`, any other serveur should give the same result.

\$PrettyPrintIdeals — Flag to switch pretty printing of ideals on and off

Description: If set to True (this is default), ideal objects will be printed in the form $\langle b_1, b_2, \ldots \rangle$ (with angle brackets) where $\{b_1, b_2, \ldots\}$ is the Gröbner basis of the ideal.

If set to False, the whole ideal expression will be printed in raw form. This might be a more convenient representation for Mathematica programmers. However, note that the internal structure of the ideal expression might be subject to changes in future versions of the interface

4 Implementation Notes

This section provides some technical remarks on the internals of the package which need not be of interest for most users.

4.1 General Tools

The following private auxiliary functions are defined.

- **ExtractVariables[*expr*]** – returns a list of subexpressions of *expr* such that *expr* is a polynomial in these subexpressions. For this purpose, a polynomial is defined recursively as the sum or product of two polynomials, a polynomial raised to the power of a positive integer, a rational number, or a variable.
- **IsPolynomial[*expr*]** – checks whether the given expression is a polynomial. The option Variables specifies a function which checks whether an expression is to be regarded as a variable. By default, only symbols are regarded as variables.
- **CleanVariables[*expr*]** – computes two lists of transformation rules which map the variable subexpressions (see ExtractVariables above) of *expr* to the canonical variables $x1, x2, x3, \ldots$ and back. The option Variables may specify a function which checks the admissiblity of variable subexpressions. Variables which are not admissible cause a warning. By default, anything is admissible as variable.
- **FileExistsQ[*string*]** – checks whether *string* points to an existing file.
- **MakeString[*ex1, ex2, ex3,...*]** – converts the arguments which are not already strings into strings (input form) and returns their concatenation.
- **FreeOfErrorsQ[*list*]** – checks if the output file represented by the given list of strings contains error messages. In this case, the error messages are printed using Print, one per line, and the function returns False. Otherwise it returns True.

4.2 Communication with the Serveurs

For comunication with the serveurs text files are used. (Snobbish Mathematica calls this "unstructured communication".) First, an input file is created. This input file is passed to an appropriate Gb serveur which writes its output to another file. This file is read into mathematica in text form and transformed into the respective Mathematica expressions which are returned.

Write Input Files

Input files are produced by the private function WriteInputFile.

WriteInputFile[*datalist*]

Description: The datalist consists of a set of transformation rules whose left hand sides correspond to the tags in the Gb files as follows.

Variables $\to \{x1, x2, \ldots\}$	$\Longrightarrow$	`#vars [x1, x2, ...]`
Characteristic $\to 0$	$\Longrightarrow$	`#coeffring rational`
Characteristic $\to p$	$\Longrightarrow$	`#coeffring modulo p`
Properties $\to \{$`"p1"`, `"p2"`$, \ldots\}$	$\Longrightarrow$	`#properties [p1, p2, ...]`
List $\to \{\{p_1, p_2, \ldots\}, \ldots\}$	$\Longrightarrow$	`#list [[p1, p2, ...], ...]`
Error $\to$ `"message"`	$\Longrightarrow$	`#error message`

A single list List $\to \{\{p_1, p_2, \ldots\}\}$ may also be written List $\to \{p_1, p_2, \ldots\}$.

Options: InFileName: string containing the desired file name. If InFileName points to a non string expression, then a temp file is used. Further option specifications are ignored.

In any case, the function returns the file name of the input file which has been written.

Run Serveur

Serveur calls are supported by the ServeurCall function

ServeurCall[*serveur*]

where *serveur* is a string such that GbHome <> `"/"` <> *serveur* yields the file name of the executable to be called.

The function returns the file name of the file to which the output was piped. If an error occurs, then the value $Failed is returned instead. An output file may or may not exist in this case.

Options: The following options specify the serveur call further

- InFileName: name of the input file from which to read the input for Gb
- OutFileName: name of the output file to which to write the output of Gb. If this option is not given (or it points to something which is not a string), then a temp file will be used.
- TimeConstraint: positive integer or Infinity for specifying timeout. If this option does not point to Infinity but to a positive integer, then the Gb call will be wrapped by a simple bash script which kills Gb when a timeout occurs.
- Flags: a list of strings specifying flags of Gb such as `"-pretend"`. The `"-read"` flag is automatically added.

Additional option specifications are ignored.

Assumptions on the format of the output file

The output file of Gb may have different shapes, depending on the flags by which Gb was called. We assume that output files by which Gröbner bases are computed have the following shape. Different assumptions may be made for reading output files of special algorithms like FGLM or triangular sets. These special assumptions can be found at the respective place in the source code of the package.

1. The first line consisting only of numbers and the symbols `"+"`, `"-"`, `"*"`, `"^"`, `"T"` and `"/"` declares the first Hilbert polynomial
2. the second line of this shape declares the second Hilbert polynomial.
3. A line starting with `"Dim : "` contains the dimension of the ideal
4. A line starting with `"Deg : "` contains the degree of the ideal
5. A line starting with `"Output: "` contains the polynomial list of the resulting Groebner basis.
6. A line starting with `"#C Time"` contains the number of seconds needed to complete the computation, and this line has the form

   `#C Time * sec *`

   where the first $*$ is a floating point number in MMA syntax and the second $*$ is arbitrary material.
7. Lines starting with `"#error"` contain errors that are reported to the user. If such a line exists, we return the value $Failed.

4.3 Structure of the ideal expression

The Ideal symbol wraps a list of transformation rules:

In[13]:= GbIdeal$[\{x+y, xy-1\}, \{x,y\}]$ // InputForm

Gb::noorder : No monomial ordering specified. Taking DRL.

Out[13]= Ideal[{MonomialOrder → DRL, Characteristic → 0, Gb‘Private‘Variables → $\{x,y\}$, HilbertPolynomial[1] → $1-T-T^2+T^3$, HilbertPolynomial[2] → $1+T$, Dimension → 0, Degree → 2, GBasis → $\{1+y^2, x+y\}$, Time → 0.Second}]

The functions that take only an ideal as input are defined such as to extract their result from this list of transformation rules. Mandatory for every ideal object are transformation rules with the following left hand sides

- Characteristic
- MonomialOrder   Lex or DRL
- Variables   (package private symbol) a list of variables or a list of two lists of variables in case of a block ordering
- GBasis   containing the Gröbner basis

In addition, an ideal may have the following entries

- Dimension
- Degree
- Time
- HilbertPolynomial[1]   the first polynomial mentioned in the output file, T is used as a variable.
- HilbertPolynomial[2]   the second polynomial mentioned in the output file, T is used as a variable.

Note that the structure of ideal objects is not considered fixed and may be subject to changes in future versions of the interface.

4.4 Warnings and Errors gallery

We list the messages which are used by the package. We distinguish between warnings and errors. The latter are fatal and cause the function to return the value $Failed while the former only indicate an unusual situation in which the computation can still proceed.

These are the warnings:

- Gb::newvars "New variables detected"
- Gb::noorder "No monomial ordering specified."
- Gb::novars "No variable order specified." (this is switched off by default)
- Gb::nochar "No characteristic specified." (this is switched off by default)
- Gb::noalgo "No algorithm specified." (this is switched off by default)
- Gb::bogusopt "Unknown Option(s) encountered"
- GbFGLM::bogusorder "Illegal order"
- GbFGLM::bogusdim "Illegal dimension"
- GbTri::bogusorder "Illegal order"
- GbTri::bogusdim "Illegal dimension"

These are the errors:

- Gb::boguselim "Eliminate → True makes only sense on elimination orderings."
- Gb::bogusfglm "You cannot specify an elimination ordering for FGLM."
- Gb::boguschar "Modulus illegal. Prime <65522 or 0 expected."
- Gb::bogusord "Termorder unknown or no termorder specified."
- Gb::timeout "Computation aborted."
- Gb::bogusalgo "Algorithm unknown or binary file not found. (Expecting, e.g., `"serveur"`, `"serveur_t"`, or `"accel_serveur"`)"
- Gb::bogusfield "Field not supported or incompatible to specified modulus"
- Gb::nyi "Not yet implemented"
- Gb::bogustiming "Timing specification not understood. Expecting a positive integer or Infinity."