An Algorithm for Converting a Degree Gröbner basis to a Lexicographic Gröbner basis*

Deepak Kapur and Tushar Saxena
Institute for Programming and Logics
Department of Computer Science
State University of New York at Albany
Albany, NY 12222

Rough Draft

Abstract

An algorithm for converting a Gröbner basis, G_1 of an arbitrary ideal from one ordering $<_1$ to a Gröbner basis G_2 with respect to another target ordering $<_2$ is presented. Its behavior does not depend upon the dimension of the input ideal and it works for any positive or zero dimensional ideal. The target ordering can be any ordering including elimination orderings such as pure lexicographic.

The algorithm is similar to the *FGLM* algorithm for basis conversion of zero dimensional ideals in the following sense: it incrementally constructs (i) a set T of terms, (ii) a set G of polynomials by computing the normal forms of the terms in T with respect to $<_1$ and solving some linear equations similar to those in *FGLM*. However the order in which terms are generated in T is not completely determined by the target ordering $<_2$. If $<_2$ does not have the property that for every term t, there are only finitely many terms smaller than t, then an enumeration ordering $<_e$ is built using $<_1$ and $<_2$ such that it has that property. For termination condition, it is first checked whether all the polynomials in G_1 reduce to 0 modulo G, and if so, then check if G is a Gröbner basis. We prove that there is a T which will be generated within a finite number of steps for which the termination condition would succeed.

The algorithm has been implemented in GEOMETER and has been successfully tried on a number of examples. It seems to work better than computing a Gröbner basis directly for ideals with small dimensions. Planned extensions include development of heuristics for developing good enumeration orderings as well as an efficient termination check. For the latter, the use of a Hilbert function of an ideal is being investigated.

1 Introduction

A Gröbner basis algorithm is considered as one of the most efficient elimination algorithms in practice for solving many problems related to polynomial ideals as well as the algebraic varieties defined by polynomial ideals. There has been a great deal of interest to obtain efficient algorithms for computing Gröbner bases. This is especially true for computations under an

*Supported in part by a grant from United States Air Force Office of Scientific Research AFOSR-91-0361.

elimination ordering of the variables as a Gröbner basis with respect to an elimination ordering is the most useful computational object for many problems. It is generally agreed that Gröbner basis computations are the hardest for elimination orderings of the variables like pure lexicographic ordering. For zero-dimensional polynomial ideals, this problem was tackled in a novel manner in [1]. Instead of computing a Gröbner basis with respect to an elimination ordering directly, Faugere et al computed a Gröbner basis w.r.t. some other ordering, like graded lexicographic (also popularly known as the degree ordering) or graded reverse lexicographic ordering, which is easy to compute, and the basis was transformed into a basis w.r.t. the desired elimination ordering. This transformation was performed using linear algebra techniques. Impressive results were reported using this technique of computing Gröbner bases of ideals w.r.t. elimination orderings in [1]. We will refer to it as the FGLM algorithm.

It has not been clear how the FGLM algorithm can be generalized to positive dimensional ideals. If the algorithm is applied on a Gröbner basis of a positive dimensional ideal, then it does not terminate. The termination of the algorithm depends upon the property that for a zero-dimensional ideal, the set of all reduced terms with respect to a Gröbner basis in its monomial ideal is finite. For a positive dimensional ideal, this set of reduced terms in its monomial ideal is infinite, consequently the FGLM algorithm goes on indefinitely enumerating terms from the reduced set forever.

We present an algorithm whose termination does not depend upon the finiteness property of the set of reduced terms with respect to a Gröbner basis in a monomial ideal. Our algorithm, instead, depends on the property that a Gröbner basis of a polynomial ideal with respect to any admissible ordering is finite. Hence the algorithm works for ideals of any dimension. This algorithm can be viewed a generalization of the algorithm developed in [1] for zero dimensional ideals.

The paper is organized as follows. The second section first reviews the algorithm for zero dimensional ideals as developed in [1]. We introduce the main new idea, using which the FGLM algorithm can be generalized so that it no longer depends on the finiteness of the set of reduced terms with respect to the target Gröbner basis in a monomial ideal. The third section discusses a naive, simple basis conversion algorithm which works for arbitrary ideals. In the fourth section, an efficient version of the algorithm is discussed. The fifth section gives the correctness argument for the algorithm. In the sixth section, heuristics are discussed to improve the performance of an implementation of the algorithm. In section seven, we present two simple examples illustrating how the algorithm transforms Gröbner bases w.r.t. graded lexicographic orders (total degree orders) to the bases for pure lexicographic orders. Finally, in section eight, some preliminary empirical results are reported.

2 A Generalization of the Zero Dimensional Algorithm

The basis conversion algorithm for zero dimensional ideals in [1] mainly relied on the fact that there are only a finite number of reduced terms w.r.t. any Gröbner basis of a zero dimensional ideal. This gives rise to a staircase like volume which encloses all the reduced terms. We give below the main steps of the algorithm given in [1]. The algorithm is called $FGLM$ and it takes as input, a Gröbner basis G_1 of the ideal with respect to an ordering $<_1$. Its output is G_2, the reduced Gröbner basis w.r.t. a target ordering $<_2$. In the following, the subscript of $<_i$ means that the comparision is done using the ordering $<_i$. For example, larger$_{<_2}$ means that larger with respect to $<_2$. Also, $\mathcal{NF}_{<_1}(t)$ is a function that computes the normal form of a term t modulo $G_1 and <_1$.

FGLM(G_1)

Input : A Gröbner basis w.r.t. $<_1$.

Output : The reduced Gröbner basis G_2 w.r.t. $<_2$.

Algorithm :

begin
$B := \phi$
$InitialTerms := \phi$
$t_0 := 1$
$i := 1$
while there exists a term larger$_{<_2}$ than t_{i-1} which is not
divisible by any element of $InitialTerms$ do
    $t_i :=$ the smallest$_{<_2}$ term larger$_{<_2}$ than t_{i-1} which is
    not divisible by any element of $InitialTerms$
    solve the equation $\mathcal{NF}_{<_1}(t_i) + \sum_{j=1}^{j=i-1} \lambda_j \mathcal{NF}_{<_1}(t_j) = 0$.
    if solution exists
    then
        let $\lambda_j' s$ be the solutions
        $B := B \cup \{t_i + \sum_{j=1}^{j=i-1} \lambda_j t_j\}$
        $InitialTerms := InitialTerms \cup \{t_i\}$
    else
        $i := i + 1$
    fi
od
return(B)
end.

The algorithm is guaranteed to terminate because for zero-dimensional ideals, the number of reduced terms for its Gröbner basis w.r.t. any admissible ordering is finite.

This algorithm cannot be used for positive dimensional ideals because in general, it does not terminate; there could potentially be infinitely many reduced terms with respect to a Gröbner basis of a positive dimensional ideal. The enumeration process could go on for ever and we may never get a Gröbner basis. For example, consider the case when the reduced Gröbner basis w.r.t. the pure lexicographic order (with $x > y > z$) is $\{x + yz, y^2z - z\}$. If $FGLM$ was run on a Gröbner basis of the same ideal with a degree ordering, terms purely in z would be enumerated and this enumeration would go on forever since there is no polynomial purely in z in this one dimensional ideal!

We will now present a generalization of the $FGLM$ algorithm and use this generalization to get an algorithm for the positive dimensional case. The main idea is to parameterize the $FGLM$ algorithm by enumerating terms only from a certain finite set T. So the inputs to this algorithm (which we call Solve) are a Gröbner basis w.r.t. $<_1$ and also an additional input which is a set of terms T. We will not specify what the output of this program is, at this point, but we will discuss some properties of it after the description.

Solve(G_1, T)

Input : A Gröbner basis w.r.t. $<_1$ and a set of terms T.

Algorithm :

begin
$B := \phi$
$InitialTerms := \phi$
$t_0 :=$ smallest$_{<_2}$ term in T
$i := 1$
while there exists a term larger$_{<_2}$ than t_{i-1} in T which is not divisible by any element of $InitialTerms$ do
    $t_i :=$ the smallest$_{<_2}$ term larger$_{<_2}$ than t_{i-1} in T which is not divisible by any element of $InitialTerms$
    solve the equation $\mathcal{NF}_{<_1}(t_i) + \sum_{j=1}^{j=i-1} \lambda_j \mathcal{NF}_{<_1}(t_j) = 0$.
    if solution exists
    then
        let $\lambda_j' s$ be the solutions
        $B := B \cup \{t_i + \sum_{j=1}^{j=i-1} \lambda_j t_j\}$
        $InitialTerms := InitialTerms \cup \{t_i\}$
    else
        $i := i + 1$
    fi
od
return(B)
end.

This algorithm is very similar to the *FGLM* algorithm outlined above except that it enumerates *only* those terms which are in a predetermined set T. We refer to this algorithm as *Solve*. Some useful properties of this algorithm are:

- *If* T is the set of all the terms in the reduced Gröbner basis with respect to the target ordering, then when algorithm returns G_2, the desired Gröbner basis w.r.t. $<_2$. Obviously, having T to be a *superset* of all the terms in the reduced Gröbner basis with respect to the target ordering would also suffice.
- In case T does *not* contain all the terms in the reduced Gröbner basis w.r.t. $<_2$, then the algorithm may not give us a Gröbner basis, but B generates a subideal of the ideal of the input Gröbner basis.

These properties of this algorithm enables us to develop an algorithm for positive dimensional ideals in the following sections.

3 The positive dimensional case

As discussed earlier, the *FGLM* algorithm would not terminate for ideals of positive dimensions because its termination relies on the fact that the set of reduced terms with respect to the

Gröbner basis is finite. In fact, the algorithm would not give any useful information for a lexicographic ordering if the ideal did not include purely a polynomial in the lowest variable in the ordering.

On the other hand, the utility of the output as well as the efficiency of the generalized algorithm $Solve$ depends upon the finiteness of T as well as an appropriate choice for T. Since the reduced Gröbner basis is finite for any admissible ordering, if T can be built by enumerating terms in such a way that eventually all terms appearing in the reduced Gröbner basis w.r.t. the desired target ordering can be included in T, $Solve$ would then return a Gröbner basis. So the parameter to $Solve$, the set of terms T may be incrementally generated by an *enumeration ordering*. Instead of requiring that the terms be enumerated in the same order as in the target ordering, we only require that terms are enumerated in such an order that we eventually generate enough terms to include terms appearing in the reduced Gröbner basis with respect to the target ordering.

A good *enumeration ordering*, $<_e$ guarantees that eventually, T will be a superset of all the terms in the reduced Gröbner basis w.r.t. $<_2$. But how do we know that we have generated enough terms so that the set of terms that we have is a superset of the set of terms in the reduced Gröbner basis? In other words, how do we reconize that T is *good enough*, even though we are guaranteed that within a finite amount of time, it would be? This is the termination condition. Clearly it cannot be that every term greater than the terms generated so far in the enumeration ordering is reducible (as in 0-dimensional $FGLM$). For positive dimensional ideals, there are infinitely many reduced terms. This is an interesting research issue to be further investigated. Currently, there are two ideas: (i) Run the algorithm $Solve$ each time a term from the enumertation ordering is produced and test if the output of $Solve$ is a Gröbner basis and also if it reduces all the polynomials of G_1 to zero. (ii) the use of Hilbert function (polynomial) for checking the number of reduced terms of various degrees. Below we sketch the algorithm that we have proposed here. A more efficient and detailed algorithm will be given in the next section. Assume that we have a good *enumeration ordering*, $<_e$. Notice that any sequential (or degree compatible) ordering would suffice.

Algorithm(G_1)

1. $T := \{1\}$
2. $B := \phi$
3. while *Gröbner ?*(B) is false do
   (a) $t :=$ the smallest term larger$_{<_e}$ than all the terms in T
   (b) $T := T \cup \{t\}$
   (c) $B := Solve(G_1, T)$
4. return(B)

Here, the predicate *Gröbner ?*(B) returns true if B is a Gröbner basis, else it returns false. Also, $reduces_B(G_1)$ returns true if all the polynomials in G_1 reduce to zero modulo B. We now discuss the two main steps in the general algorithm:

- How to generate T systematically and efficiently based on the input Gröbner basis and the target ordering so that eventually all terms in the output Gröbner basis would be included?

- How can the two tests – whether a given basis generates the same ideal as the input basis, and whether a given basis is a Gröbner basis, be performed efficiently?

Incrementing T

If the target ordering is degree compatible, then it could be used as an enumeration ordering. A degree compatible ordering has the nice property that there are finitely many terms smaller than any term in the ordering. For arbitrary ideals, elimination orders such as lexicographic orderings are not likely to work as enumeration orderings because in an elimination ordering, there can be infinite terms below the smallest head-term of a Gröbner basis with respect to the target order. For the example above, there are infinitely many terms lower than the smallest head term y^2z. In general, it should be possible to make use of the structure of the input Gröbner basis to pick an appropriate ordering which is a mixture of a degree compatible and elimination orderings such as block-orderings. This issue is under further investigation. If the target ordering is degree compatible, then it is used as an enumeration ordering. If the target ordering is not degree compatible but the input Gröbner basis is computed using a degee compatible ordering, then the input ordering is used for enumeration. Otherwise, a degree ordering (with input order to break ties) is used. We call this order $<_e$. So the strategy for incrementing T is as follows:

Start with an empty T and increment it by the terms in an ascending $<_e$ order.

Assume that T_s is the set of all terms which are smaller, w.r.t. $<_e$, than a term s. Let the largest term, w.r.t. $<_e$, which appears in a reduced Gröbner basis with respect to the target ordering be q. Since T_q is finite, it is easy to see that T *will* equal T_q within a finite number of steps. This guarantees that the general algorithm will terminate.

Because of the relaxation of this requirement about an enumeration ordering being different from a target ordering, when a new term is enumerated, it is not necessarily greater in the target ordering than previously enumerated terms. So when we solve for reducible enumerated terms, we may get more than one polynomials to be potential candidates for being elements of a Gröbner basis with respect to the target ordering.

Termination test

Two tests must succeed for the algorithm to terminate. The basis B generated by the general algorithm must generate the same ideal as the input basis, and B should be a Gröbner basis with respect to the target ordering. One way to perform the first test is to check whether all polynomials in the input Gröbner basis reduce to 0 using B; this would ensure that B generates the same ideal as the input Gröbner basis since every element in B is in the ideal of the input Gröbner basis.

For the second test, one possibility is to check whether all nontrivial S-polynomials (critical pairs) of B reduce to 0. This test can be performed incrementally making use of the information from previous unsuccessful attempts of the test.

There are perhaps other ways of testing whether a Gröbner basis of the same ideal has already been generated. One possibility is to use the Hilbert function of an ideal using its Gröbner basis given as the input. The main idea is to count the number of irreducible terms for every degree, and then use this information to decide whether a Gröbner basis with respect to the target ordering has been generated or not. This approach has not yet been fully studied in this framework, but it may have interesting applications. We are particularly concerned about the difficulty in computing a Hilbert function of positive dimensional ideals.

In the next section, we present a more efficient variant of this algorithm. In another section, we present heuristics using which this algorithm can be made faster.

4 Basis conversion algorithm

The following algorithm, *BasisConvert*, generates a Gröbner basis G_2 w.r.t. a target ordering $<_2$, given a Gröbner basis G_1 w.r.t. a term order $<_1$ The *Solve* algorithm has been slightly changed in this section for efficiency considerations.

BasisConvert(G_1)

Input : A Gröbner basis G_1 w.r.t. $<_1$.

Output : A Gröbner basis, G_2 w.r.t. $<_2$ of the ideal generated by G_1.

Algorithm :

begin
$G_2 := \phi$;
$InitialTerms := \phi$;
$ReducedTerms := \phi$;
$t := 1$;
While not($TerminationTest(G_1, G_2)$) do
    $t := NextTerm(t, InitialTerms)$;
    $ReducedTerms := ReducedTerms \cup \{t\}$;
    $G' := Solve(G_1, ReducedTerms)$;
    if $G' \neq \phi$ then
        $G_2 := G_2 \cup G'$;
        $InitialTerms := InitialTerms \cup HeadTerms(G')$;
        $ReducedTerms := RemoveMultiples(ReducedTerms, HeadTerms(G'))$;
    fi;
od;
Return(G_2);
end;

The input/output specifications of some of the procedures used in the above algorithm are given below. Procedure *Solve* is described fully. The enumeration order, $<_e$ is defined to be $<_2$ if $<_2$ is degree compatible, $<_1$ if $<_1$ is degree compatible, else it is defined as follows:

$$a <_e b \ if \ degree(a) < degree(b) \ or (degree(a) = degree(b) \ and \ a <_1 b)$$

NextTerm(t, S)

Input : A term t and a set of terms S.

Output : The smallest term larger than t w.r.t. $<_e$ and not divisible by any term in the set S.

HeadTerms(S)

Input : A set of polynomials S.

Output : The set of headterms of all polynomials in S w.r.t. $<_2$.

RemoveMultiples(T, S)

Input : A set of terms T and another set of terms S.

Output : The set $T - multiples(S)$ where $multiples(S)$ is the set of all multiples of the terms in S.

Solve(G_1, T)

Input : A set of terms T and any term order $<$.

Output : The set B of all polynomials p which satisfy the following properties:

1. p is in the ideal generated by G_1,
2. p is a linear combination of the elements of T and
3. There is no polynomial, p' which is in the ideal generated by G_1 and is a linear combination of terms in T, but $HeadTerm(p')$ divides some term of p.

Algorithm :

begin
$B := \phi$
$t_0 := 1$
$i := 1$
while there exists a term larger than t_{i-1} in T w.r.t. $<$ do
    $t_i :=$ the smallest term larger than t_{i-1} in T (w.r.t. $<$)
    solve the equation $\mathcal{NF}_{<_1}(t_i) + \sum_{j=1}^{j=i-1} \lambda_j \mathcal{NF}_{<_1}(t_j) = 0$.
    if solution exists
    then
        let $\lambda_j's$ be the solutions
        $S := S \cup \{t_i + \sum_{j=1}^{j=i-1} \lambda_j t_j\}$
        $T := RemoveMultiples(T, t_i)$
    else
        $i := i + 1$
    fi
od
return(S)
end.

TerminationTest(G_1, G_2)

Input : A Gröbner basis w.r.t. $<_1$ viz. G_1 and the set of polynomials G_2.

Output : *True* if $G_{<_2}$ is a Gröbner basis of the ideal generated by $G_{<_1}$ w.r.t. $<_2$, else *False*.

Algorithm : Test if every polynomial in $G_{<_1}$ normalizes to zero w.r.t. $G_{<_2}$. If they don't then return false else test if $G_{<_2}$ is a Gröbner basis. More details and heuristics for this procedure will be outlined in section 6.

5 Proof of correctness of $BasisConvert$

5.1 Proof of correctness of $Solve$

For a fixed T and an ordering $<_2$, the set of all polynomials which satisfy all the conditions in the output of $Solve$ is unique (call it $P_{T,<_2}$). Let $P_{T,<_2} = \{p_1, p_2, ..., p_n\}$. Also let this be an ordered set, i.e., $ht_{<_2}(p_i) <_2 ht_{<_2}(p_j)$ if $i < j$. The following lemma on $P_{T,<_2}$ proves the correctness of procedure $Solve$:

Theorem 5.1 *For a given T and an ordering $<$, $Solve(G_1, T)$ produces all the polynomials in $P_{T,<_2}$, in that order.*

Proof: We use mathematical induction on i (where p_i is the i^{th} polynomial in $P_{T,<_2}$) to prove this lemma.

Base case Let q_1 be the first polynomial that $Solve$ produces. q_1 satisfies the first requirement in the output of $Solve$ because of the fact that the linear equation which resulted in q_1 was solvable implies that q_1 is in the ideal generated by G_1. The second condition is also satisfied since the linear equation which resulted in q_1 was only in terms of the normal-forms of the terms in T. The third condition is satisfied because in the procedure $Solve$ we are enumerating terms from T in $<_2$ order. Hence if there was any polynomial smaller than q_1 which was in the ideal and could be formed using terms in T, it's corresponding linear equation would have been solvable so, we would already have it and q_1 wouldn't be the first one. For any polynomial's head term to divide some term in q_1, that polynomial must be smaller than q_1. Since we don't *have* any polynomial with smaller headterm w.r.t. $<$, the third condition is also satisfied. Hence q_1 is the smallest polynomial which satisfies all the three conditions in the output specification of $Solve$ so $q_1 = p_1$.

Induction hypothesis Let us assume that the first i polynomials produced by the algorithm are $p_1 \cdots p_i$.

We now prove that the $(i+1)^{th}$ polynomial produced by $Solve$ (q_{i+1}) is the same as the $(i+1)^{th}$ polynomial of $P_{T,<_2}$ i.e., p_{i+1}. This can be proven in the same way as the base case by replacing T by T_i and $P_{T,<_2}$ by $P_{T_i,<_2}$ in the proof, where

$$T_i = T - \{multiples(ht_{<_2}(p_1)), multiples(ht_{<_2}(p_2)), \cdots, multiples(ht_{<_2}(p_i))\}$$

and realizing the simple fact that

$$P_{T_i,<_2} = \{p_{i+1}, p_{i+2}, \cdots, p_n\}$$

□

This concludes the proof of correctness of $Solve$.

5.2 Proof of correctness of $BasisConvert$

We will now prove that within a finite number of steps, $BasisConvert$ will return a Gröbner basis of the ideal w.r.t. the target ordering. We prove a lemma which establishes that within a finite number of steps, T will contain all the terms of the reduced Gröbner basis and then the Gröbner basis will be computed by invoking $Solve$ at this point.

Lemma 5.2 *Let s be a term. In a finite number of steps of the algorithm, all the terms which are smaller than s w.r.t. $<_e$ will either be in the set $ReducedTerms$ or be divisible by some term in the set $InitialTerms$.*

Proof: We say that a term is being considered by the procedure $NextTerm$ when it is either produced by $NextTerm$ or it is skipped by $NextTerm$. Notice that a term is said to have been produced by $NextTerm$ if it is actually returned by the procedure.

Since the enumeration is being done in $<_e$ order which is degree compatible, s will be considered in a finite number of steps. At that point, all the terms smaller that s would already have been considered by $NextTerm$. For each smaller term s_i (all of which were considered), we can do the following analysis:

1. s_i was not produced. Since it was not produced, it must be divisible by some term in $InitialTerms$, and will continue to be so since nothing is ever removed from $InitialTerms$.
2. s_i was produced, hence one of the following will be true:
   (a) s_i was put in the set $InitialTerms$ (and then will continue to be in that set) or
   (b) s_i was put in $ReducedTerms$. Hence, one of the following will be true:
      i. s_i will continue to be in $ReducedTerms$, or
      ii. s_i will become divisible by some term in $InitialTerms$ at a later point of time, and be removed from $ReducedTerms$.

Hence, when s is being considered, all terms smaller than s would either be in $ReducedTerms$ or be divisible by some element of $InitialTerms$. $\Box$

We will denote the set of all terms smaller than or equal to a term s w.r.t. $<_e$ to be T_s. Also $RG_<$ denotes the reduced Gröbner basis of an ideal w.r.t. the ordering $<$. If S is a set of polynomials, then $tail(S)$ is the difference of the set of all terms in all the polynomails of S and the set of head terms of all polynomials in S..

Lemma 5.3 *Let $T(RG_{<_2})$ be the set of all terms in the reduced Gröbner basis of the ideal generated by the original Gröbner basis G_1 w.r.t. the target term ordering. In a finite number of steps of the algorithm, all the following become true together:*

1. $T(RG_{<_2}) \subseteq ReducedTerms \cup InitialTerms$
2. $tail(RG_{<_2}) \subseteq ReducedTerms$
3. $\forall p \epsilon RG_{<_2}, ht(p) \epsilon InitialTerms \Rightarrow \exists q \epsilon G_2 ht(p) = ht(q)$

4. $\forall p \epsilon RG_{<_2}, ht(p) \epsilon ReducedTerms \Rightarrow p \epsilon Solve(G_1, ReducedTerms)$

Proof: From the properties of $Solve$, we can assume that at all the polynomials in G_2 are from the ideal generated by G_1.

We first prove that within a finite number of steps, the first condition will be satisfied. Then we prove that the first condition implies all the remaining conditions, hence proving the lemma.

1. First condition becomes true in a finite number of steps: Let the largest term in $T(RG_{<_2})$ w.r.t. $<_e$ be s. Then by previous lemma, within a finite number of steps, every term of $T_s(\supseteq T(RG_{<_2}))$ will either be in $ReducedTerms$ or be divisible by some term in $InitialTerms$. But all the terms in $InitialTerms$ are head terms of some polynomial in the ideal. Since there is no term $t \,\epsilon\, T(RG_{<_2})$ which can be divisible by the head term (say h) of another polynomial in the ideal unless $h = t$, all the terms of $T(RG_{<_2})$ must either be in $ReducedTerms$ or be in $InitialTerms$ (as opposed to simply being divisible by some term in $InitialTerms$).

2. First condition implies the second condition: Previous condition implies that all the terms in the tail of every polynomial in $RG_{<_2}$ must either be in $ReducedTerms$ or be in $InitialTerms$. But any term in the tail of a polynomial in the reduced Gröbner basis can never be the head term of another polynomial in the ideal (since this contradicts the fact that $RG_{<_2}$ is reduced). Hence all the terms in the tail of all the polynomials of $RG_{<_2}$ must be in the set $ReducedTerms$.

3. Third condition is always true: The third condition is an invariant of the main loop of the algorithm. If any term is in $InitialTerms$, it is clear from the algorithm that it must be the case that there is a polynomial in G_2 whose head term is the same as this term (since we only put head terms in $InitialTerms$).

4. Previous conditions imply the fourth condition: If the head term of any polynomial in $RG_{<_2}$ is in $ReducedTerms$, then, since the tail of this polynomial must be in $ReducedTerms$ (by the second condition), this polynomial would satisfy the second condition in the output of $Solve$. It will also satisfy the first condition of the output of $Solve$ since it is obviously in the ideal. Moreover, since this polynomial is a member of the reduced Gröbner basis, there is no polynomial at all in the ideal whose head term divides any term in this polynomial (unless it is larger than this polynomial w.r.t. $<_2$), it also satisfies the third condition of the output of $Solve$. Since it satisfies all the three conditions of the output of $Solve$, it must be produced by $Solve$ when invoked with $ReducedTerms$.

□

Theorem 5.4 *In a finite number of steps, G_2 would be a Gröbner basis for the ideal generated by G_1.*

Proof : By the previous corollary, after a finite number of steps, the head terms of all polynomials in the reduced Gröbner basis are either in $ReducedTerms$, and then by the fourth condition of the previous corollary they will be produced (and hence be put in G_2), or in

$InitialTerms$, in which case, by the third of the previous corollary, there already exists a polynomial in G_2 which has the same head term as this one. So for all polynomials in $RG_{<_2}$, there exists a polynomial in G_2 which has the the same head term as this one. Hence by Proposition 5.38 of [2], G_2 must be a Gröbner basis. □

This last theorem proves the correctness of $BasisConvert$. Notice that $BasisConvert$ *does not* necessarily produce the reduced Gröbner basis w.r.t. $<_2$. All that we have proved is that it produces a Gröbner basis. The reduced Gröbner basis can easily be obtained from this Gröbner basis by deleting all polynomial whose head terms are multiples of the head terms of other polynomials and then interreducing them.

6 Optimizations

The most time-consuming steps in the algorithm are:

- normal form computations of terms,
- solving linear equations to generate polynomials expressed using (possibly) reduced terms with respect to G_2 in the ideal,
- testing for ideal equality, and
- testing for the output to be a Gröbner basis.

6.1 Generating normal forms of terms

Computing the normal form of a term t is done in two steps.

1. Generate a polynomial (which we call a *subnormal form*) whose normal form is the same as that of t. An objective here is to use normal forms already computed and saved in a hash-table.
2. Compute the normal form of a subnormal form.

- **Generating a subnormal form.** Since terms are being generated in a degree compatible order by the procedure $NextTerm$, when a term t is generated, the set $ReducedTerms$ contains terms, all of which are smaller than t w.r.t. $<_e$. For every variable x_i which divides t, it must be the case that the term $\frac{t}{x_i} \epsilon$ $ReducedTerms$ (as otherwise , there must exist a variable x_k such that $\frac{t}{x_k}$ is divisible by some term in $InitialTerms$, and hence, so is t so t should not have been generated anyway). Let x_s be the smallest variable which divides the term t. The normal form of t, $\mathcal{NF}(t)$, is computed as follows:

$$\mathcal{NF}(t) = Normalize(x_s \times \mathcal{NF}(\frac{t}{x_s}))$$

The polynomial $x_s \times \mathcal{NF}(\frac{t}{x_s})$ is called a *subnormal form* of t.

- **Computing the normal form of a subnormal form.** The procedure $Normalize$ is invoked on a polynomial such that all the terms in it are smaller than t w.r.t. $<_1$. Since the summation of the normal forms of terms gives the normal form of the whole polynomial, some of the terms of this polynomial which are smaller than t w.r.t $<_e$ can be replaced by their normal forms, which have already been computed and stored. It turns out that in practice, if the normal forms of the terms *not* smaller than t w.r.t $<_e$ are also computed and stored, they can be used in subsequent computations making the normalization process more efficient. If $<_1$ coincides with the enumeration term order $<_e$, i.e. $<_1$ is degree compatible, then we will have normal forms of all the terms of the polynomial which are not divisible by any term in $InitialTerms$, and this further simplifies the computation of normal forms.

6.2 Solving Linear Equations

Instead of invoking $Solve$ after every term is generated, it is more efficient to invoke only after all the terms of a certain degree have been generated. For every degree, we invoke $Solve$, and if new polynomials are generated to be included in the new basis, normalize G_1 w.r.t. the new basis collected till that point, and if all the polynomials in G_1 reduce to zero, test the new basis for a Gröbner basis. For solving a system of equations, the optimization used for the zero-dimensional case ([1]) can be used.

6.3 Normalizing G_1 and testing G_2 for Gröbner basis

- **Partial normalization of G_1.** As should be obvious, the test for checking whether every polynomial in G_1 reduces to 0 using the new basis is done incrementally when new polynomials are added to the new basis. If during this test, a polynomial in G_1 does not reduce to 0, $false$ is returned. Next time the same test is invoked, there is no need to check again the polynomials in G_1 which earlier reduced to 0 since polynomials are not being deleted from the new basis. The test can resume from a normal form of the last polynomial in G_1 that did not reduce to 0.

- **Testing G_1 for Gröbner Basis.** This test is also done incrementally. Firstly we need to consider only those polynomials in the new basis whose head-terms do not reduce by the other polynomials in the new basis. For any distinct pair of such polynomials in the new basis, if the associated S-polynomial (critical pair) does not reduce to 0, $false$ is returned. Next time this test is invoked, we can resume from this S-polynomial itself (assuming the head term of any of the polynomials in the new basis from which the S-polynomial was generated does not reduce by any new polynomial added to the new basis). Finally, the criteria (like Buchberger's two criteria [4]) which minimize the number of S-polynomials needed to be reduced in the classical algorithm can be used to optimize the number of S-polynomials which must be considered for a Gröbner basis test.

Because of these optimizations, polynomials in G_1 are reduced exactly once.

7 Examples

Below we illustrate the main steps of the general algorithm on a couple of very simple examples.

Example 1: Consider the following Gröbner basis of a one dimensional ideal w.r.t. the total degree order with variables $x > y > z$.

$$G_1 = \{yz + x, xy + z, x^2 - z^2\}.$$

The general algorithm can be used to transform G_1 into G_2 where $<_2$ is the pure lexicographic order with $x > y > z$. In the following trace, we present the snapshots of the values of of the sets *ReducedTerms*, *InitialTerms*, G_2 after all the terms of a certain degree have been generated and the call to *Solve* has been made. Finally it is indicated how much of G_1 has been reduced by the current G_2. In the last case, we start with G_1 at degree 1 and keep reducing at each step until this set reduces to ϕ and then test G_2 for Gröbner basis.

After all the terms of degree one have been produced

$$ReducedTerms = \{z, y, x\},$$
$$InitialTerms = \phi,$$
$$G_2 = \phi,$$
$$G_1 = \{yz + x, xy + z, x^2 - z^2\} \textit{ reduced to } \{yz + x, xy + z, x^2 - z^2\}.$$

After all the terms of degree two have been produced

$$ReducedTerms = \{z, y, z^2, yz, y^2\},$$
$$InitialTerms = \{x\},$$
$$G_2 = \{x + yz\},$$
$$G_1 = \{yz + x, xy + z, x^2 - z^2\} \textit{ reduced to } \{y^2 z - z, x^2 - z^2\}.$$

After all the terms of degree three have been produced

$$ReducedTerms = \{z, y, z^2, yz, y^2, z^3, yz^2, y^2 z, y^3\},$$
$$InitialTerms = \{x, y^2 z\},$$
$$G_2 = \{x + yz, y^2 z - z\},,$$
$$\{y^2 z - z, x^2 - z^2\} \textit{ reduced to } \phi.$$

Since, at this point, every polynomial from the original Gröbner basis was reduced to zero, G_2 is tested for a Gröbner basis. The test succeeds. Hence G_2 is the Gröbner basis of the ideal generated by G_1 w.r.t. the pure lexicographic order $<_2$.

Example 2: Consider another Gröbner basis of a one dimensional ideal w.r.t. the total degree order with variables $x > y > z$:

$$G_1 = \{xy - x, x^2 + xz, y^2z + x\}.$$

The goal is to transform G_1 into G_2 where $<_2$ is the pure lexicographic order with $x > y > z$. Notice that in this example, when the polynomials in the original Gröbner basis G_1 reduce to 0, G_2 is still not a Gröbner basis. One more step is needed. Also, after degree 2, some polynomials are obtained, which belong to the ideal, but later other polynomials are produced whose headterms divide the headterms of these polynomials.

After all the terms of degree one have been produced

$$ReducedTerms = \{z, y, x\},$$

$$InitialTerms = \phi,$$

$$G_2 = \phi,$$

$$G_1 = \{xy - x, x^2 + xz, y^2z + x\} \textit{ reduced to } \{xy - x, x^2 + xz, y^2z + x\}.$$

After all the terms of degree two have been produced

$$ReducedTerms = \{z, y, x, z^2, yz, y^2, xz\},$$

$$InitialTerms = \{xy, x^2\},$$

$$G_2 = \{xy - x, x^2 + xz\},$$

$$G_1 = \{xy - x, x^2 + xz, y^2z + x\} \textit{ reduced to } \{y^2z + x\}.$$

After all the terms of degree three have been produced

$$ReducedTerms = \{z, y, z^2, yz, y^2, z^3, yz^2, y^2z, y^3\},$$

$$InitialTerms = \{x\},$$

$$G_2 = \{x + y^2z, xy - x, x^2 + xz\},$$

$$\{y^2z + x\} \textit{ reduced to } \phi.$$

At this point, since all polynomials in G_1 reduce to 0, G_2 must be tested for a Gröbner basis. This test fails because the S-polynomial of $\{x + y^2z, xy - x\}$ does not reduce to 0. So we enumerate terms of degree 4.

After all the terms of degree four have been produced

$$ReducedTerms = \{z, y, z^2, yz, y^2, z^3, yz^2, y^2z, y^3, z^4, yz^3, y^2z^2, y^4\},$$

$$InitialTerms = \{x, y^3z\},$$

$$G_2 = \{x + y^2z, y^3z - y^2z, xy - x, x^2 + xz\}.$$

Again the test for a Gröbner basis is performed and this time it succeeds. Hence G_2 is a Gröbner basis for the ideal generated by G_1. The Gröbner basis from G_2 is obtained by deleting polynomials whose head terms can be reduced by the head terms of other polynomials.

8 Empirical results

The general algorithm has been implemented in GEOMETER system [3]. The heuristics outlined in section 6 have been tried. A degree ordering was used for enumeration of terms. Polynomials in the input Gröbner basis were incrementally reduced to 0 to ensure that the output generated the same ideal. At the end, a Gröbner basis test was incrementally run in which S-polynomials are reduced to 0. The performance of the general basis conversion algorithm is compared with the classical Buchberger's Algorithm using the normal selection strategy [4]. Some of the examples tried for comparisons were given below. The polynomials for each set are written in the infix notation and a Gröbner basis was computed w.r.t. the term order $A > B > C > D > E > F$.

1. ```
   (+ A B C D E F)
   (+ (* A B) (* B C) (* C D) (* D E))
   (+ (* A B C) (* B C D) (* C D E))
   (+ (* A B C D) (* C B D E))
   (+ (* A B C D E) -1)
   ```

2. ```
   (+ A B C D E F)
   (+ (* A B) (* B C) (* C D) (* D E))
   (+ (* A B C) (* B C D) (* C D E))
   (+ (* A B C D) (* B C D E))
   (+ (* A B C D E F) -1)
   ```

3. ```
   (+ A B C D E F)
   (+ (* A B) (* B C) (* C D) (* D E) (* E F))
   (+ (* A B C) (* B C D) (* C D E))
   (+ (* A B C D) (* B C D E))
   (+ (* A B C D E F) -1)
   ```

4. ```
   (+ A B C D E F -1)
   (+ (* A B) (* B C) (* C D) (* D E) -1)
   (+ (* A B C) (* B C D) (* C D E))
   (+ (* A B C D) (* B C D E))
   (+ (* A B C D E F) -1)
   ```

In Figure 1 above, timings are given for computing a degree basis on GeoMeter (a), converting to a lexicographic basis from the degree basis on GeoMeter (b), total time taken to obtain the lexicographic basis by conversion (c) (= a + b), obtaining the lexicographic basis directly on GeoMeter (d) and finally, for obtaining the lexicographic basis directly on Maple (e). A star (*) entry indicates that either the program ran out of space or it ran for more than 24 hours.

It was found that for most nontrivial examples, *BasisConvert* performed better than the classical algorithm as implemented in GEOMETER and MAPLE. Comparisons with other systems are difficult to make because their implementations and handling of data structures could possibly be totally different.

There is considerable room for improving the performance of the *BasisConvert* algorithm. We earlier mentioned issues related to enumeration orderings as well as tests for termination.

Example	a	b	c	d	e
1	16	221	237	570	2010
2	27	365	382	811	*
3	35	3850	3885	8894	*
4	368	19213	19581	*	*

Figure 1: Comparison of $BasisConvert$ with direct computation of Gröbner basis

Further, currently we are using a naive Gaussian elimination procedure in $Solve$ using the *kcl* infinite precision rational arithmetic. A better implementation of a Gaussian elimination method is likely to result in much better timings.

References

[1] J.C. Faugere, P. Gianni, D. Lazard and T. Mora Efficient computation of zero-dimensional Gröbner bases by change of ordering. *To appear in: J. Symb. Comp.* 1990.

[2] T. Becker, V. Weispfenning in cooperation with H. Kredel Gröbner Bases - A Computational Approach to Commutative Algebra. *Springer Verlag* 1993.

[3] C.I. Connolly, D. Kapur, J.L. Mundy and R. Weiss GeoMeter: A System for Modeling and Algebraic Manipulation. *Proc. of DARPA Workshop on Image Understanding.* Palo Alto, CA, pp 797-804 1989.

[4] B. Buchberger, Gröbner bases: An algorithmic method in polynomial ideal theory. In: (ed. N.K. Bose) *Multidimensional System Theory,* D. Reidel Publishing Co, 1985, 184-232.