Commutativity Theorems

Examples in Search of Algorithms

John J Wavrik
Department of Mathematics
University of California - San Diego

dedicated to the memory of John Hunter

Introduction

Commutativity theorems are part of the study of polynomial identities in non-commutative rings. They are theorems which assert that, under certain conditions, the ring at hand must be commutative. The proofs of theorems of this sort in their general form require the structure theory for non-commutative rings. Instances of these theorems have a strongly computational flavor. They provide interesting test examples for algorithms which use re-write rules and reduction theory for polynomial rings in non-commuting variables. This paper presents several examples of commutativity theorems with solutions. The solutions were obtained using a reduction process for non-commutative polynomials with integer coefficients. The reduction process blends a treatment of integer coefficients due to Buchberger with handling of non-commutative polynomials due to Mora. Some comparisons are made between automated solutions and solutions "by hand".

Background

$Z<x_1,..,x_n>$ will denote the ring of polynomials over the integers in non-commuting variables $x_1,..,x_n$. Let R be a ring (not necessarily unital, not necessarily commutative). We say that $f\in Z<x_1,..,x_n>$ is a polynomial identity on R if $f(a_1,..,a_n) = 0\ \forall a_i\in R$. R is called a PI ring if it satisfies a non-trivial polynomial identity. Commutative rings, for example, all satisfy the polynomial identity xy-yx. The ring of 2×2 matrices over a field satisfies the identity $[[x,y]^2,z]$ where $[x,y] = xy-yx$.

Herstein [Her] discusses a collection of results designated Commutativity Theorems. These theorems provide conditions on a ring R which can be shown to imply that R is commutative. An example is a Theorem of Jacobson:

Let R be a ring for which $\forall a\in R\ \exists$ integer $n(a)>1$ with $a^{n(a)}=a$, then R is commutative.

Proofs of these theorems in general form use structure theory for non-commutative rings. Instances of these general theorems can be examined from a computational point of view. In this paper we will look at some commutativity theorems both as an application of reduction processes to automated theorem proving, and as a source of test examples for this type of machinery.

Many of the commutativity theorems discussed in [Her] assert that if R satisfies a certain polynomial identity then R is commutative. The set of polynomial identities on R, $I_R \subset Z<x_1,..,x_n>$, is a two sided ideal. In addition to the use of ideal operations, we may obtain new polynomial identities by making substitutions $t_i = g_i(x_1,..,x_n)$ for the variables in a known identity $f(t_1,..,t_m)$. The matter is to show that the identity xy-yx is a consequence of the hypothesis identities: that it can be obtained by a succession of substitutions and ideal operations.

Simple Examples

The simplest example is often found as an exercise in undergraduate textbooks in abstract algebra [e.g. Her2]:

Theorem 1: If $a^2 = a\ \forall a \in R$ then R is commutative.

proof: Assume $a^2 - a = 0\ \ \forall a \in R$.
We have $0 = (x + y)^2 - (x + y) = x^2 + xy + yx + y^2 - (x + y)$
$= xy + yx$
so $xy = -yx\ \ \forall\ x,y \in R$
Also $a = (a)^2 = (-a)^2 = -a$
So $xy = yx$

Here are more sophisticated examples:

Theorem 2: If R is a ring with 1 for which $(ab)^2 = a^2b^2\ \forall a,b \in R$ then R is commutative.

proof: We make substitutions in abab - aabb:

a=x, b=y	xyxy - xxyy
a=1+x, b=y	xyxy - xxyy + yxy - xyy = yxy - xyy
a=x, b=1+y	xyxy - xxyy + xyx - xxy = xyx - xxy
a=1+x, b=1+y	xyxy - xxyy + yxy - xyy + xyx - xxy + yx - xy = yx - xy

Theorem 3: If R is a ring with no nilpotents for which $(ab)^2 = a^2b^2\ \forall a,b \in R$ then R is commutative.

proof: We again make substitutions in abab - aabb to obtain a sequence of identities:

(1)	a=x, b=y	xyxy - xxyy
(2)	a=x+y, b=x	-yyxx + yxyx - xyxx + xxyx
(3)	a=x+y, b=y	yyxy - yxyy + xyxy - xxyy
(4)	a=x, b=x+y	xyxy + xyxx - xxyy - xxyx
(5)	a=y, b=x+y	-yyxy - yyxx + yxyy + yxyx

Each of these identities is now reduced by others (we will discuss this in more detail later) The original set of identities reduces to:

(6)	-xyxx + xxyx	= (1) - (4)
(7)	- yyxx + yxyx	= (2) - (6)
(8)	yyxy - yxyy	= (3) - (1)
(9)	xyxy - xxyy	= (4) + (6)

(5) reduces to 0: (5) = -(3) + (2) + (4)

Now let f = xy - yx. We find that $f^3 = -y(9)x - y(6)y + x(8)x + x(7)y$. Thus $(xy-yx)^3 = 0$. Since R has no nilpotent elements we have xy-yx = 0, so R is commutative.

Note that the identity $(ab)^2 = a^2b^2$ alone, without additional conditions, does not guarantee commutativity. The ring, R, of strictly upper triangular matrices in $M_3(\mathbb{Q})$ is a counter-example.

Theorem 4: Let R be a ring with 1 for which $2x = 0 \Rightarrow x=0$. If $(ab)^2 = (ba)^2$ $\forall a,b \in R$ then R is commutative.

proof: Let $F(a,b) = abab - baba$.
Then $F(1+x,1+y) - F(1+x,y) - F(x,1+y) + F(x,y) = -2\,yx + 2\,xy$

Harder Examples with machine solutions

The previous proofs are easily done by hand. They show a typical strategy used to solve these problems: (1) preliminary simple substitutions are made in the initial identities; (2) new identities are used for mutual reduction (simplification) to obtain a basis for a subideal of I_R; (3) the process is continued until xy-yx can be deduced.

For machine computation, the choice of starting substitutions is made by hand. A critical pairs reduction process is applied (like that used to compute Gröbner Bases) to produce a new basis from the starting identities.

In the following theorems we give the initial identities and a set of substitutions that was found to lead to a proof. The algorithm for computing an R-Basis is applied. The identity xy-yx is either a member of the R-Basis or commutativity is obtained by a final step which will be described.

Theorem 5: Let R be a ring with no nilpotents for which $(ab)^2 = (ba)^2$ $\forall a,b \in R$. Show that R is commutative.

proof: Let $F(a,b) = abab - baba$
The Basis Procedure is started with F(x,y), F(y+x,x), F(y+x,y), F(-y+x,x), F(-y+x,y), F(yx,yy), F(xy,xx), F(xx,yy), F(yx,xy), F(xx+x,y), F(yy+y,x).
An R-Basis is produced with 15 elements (all consequences of baba=abab):

$R[1] = x^2y^2x^2y^2x - x^5y^4$
$R[2] = yx^3 - xyx^2 + x^2yx - x^3y$
$R[3] = -yx^2y^2x^2y + xy^2x^2y^2x$
$R[4] = y^2x^2y^2x^2 - x^2y^2x^2y^2$
$R[5] = -x^3y^2x + x^4y^2$
$R[6] = yx^2y^3 - x^2y^4$
$R[7] = yxyx - xyxy$
$R[8] = y^3x - y^2xy + yxy^2 - xy^3$
$R[9] = yxy^2x - xy^2xy$
$R[10] = -yx^2yx + xyx^2y$
$R[11] = -yx^2y^2xy - x^2yxy^3 + 2x^3y^4$
$R[12] = 2x^2yxy^3 - 2x^3y^4$
$R[13] = xyx^2y^2 - x^2y^2xy$
$R[14] = -2x^3yxy^2 + 2x^4y^3$
$R[15] = x^2y^2xy^2 - x^3y^4$

The polynomial $(xy-yx)^5$ reduces to zero in 89 reduction steps. Since we have assumed that R has no nilpotent elements, we have xy-yx = 0.

Theorem 6: Let R be a ring in which $a^3 = a\ \forall a \in R$. Show that R is commutative

proof: Let F(a) = aaa-a. Notice that $(2a)^3 = 2a$ so that 6a is a consequence of the identity. The following are used to initiate the R-Basis procedure: 6x, 6y, F(x), F(y), F(y+x), F(-y+x), F(xx+x), F(yy+y), F(-yx+xy).

In this case the procedure produces a finite R-Basis which includes the identity xy - yx:

$R[1] = 6x$
$R[2] = 6y$
$R[3] = x^3 - x$
$R[4] = yx - xy$
$R[5] = -3y^2 - 3y$
$R[6] = -y^3 + y$
$R[7] = -3x^2 - 3x$

In this case we obtain the Gröbner bases for the x and y variables together with the commutativity relation.

Theorem 7: Let R be a ring in which $a^4 = a\ \forall a \in R$. Show that R is commutative

proof: Let F(a) = aaaa-a. Notice that $-a = (-a)^4 = a^4 = a$ so that 2a is a consequence of the identity. As a result we can either use the R-Basis algorithm or Mora's algorithm over the field Z_2. Both yield the same results and behave similarly. We also use the lemma below to deduce the relation a^3b-ba^3. The following are used to initiate the R-Basis procedure: 2x, 2y, F(x), F(y), F(y+x), F(yx+x), F(xy+y), F(xy), F(yx), F(xy+x), F(yx+y), yyyx-xyyy, -yxxx+xxxy.

A finite R-Basis is produced:

$R[1] = -yx^3 - x^3y$
$R[2] = -yx^2y - x^2y^2 - yxy - yx^2 - xy^2 - xyx$
$R[3] = -xyx^2 - x^3y - yxy - yx^2 - xy^2 - x^2y - yx - xy$
$R[4] = -yxy^2 - xy^3 - xyx - x^2y$
$R[5] = -(yx)^2 - x^2y^2$
$R[6] = -y^2x - yx^2 - xy^2 - x^2y - yx - xy$
$R[7] = -y^4 - y$
$R[8] = -(xy)^2 - x^2y^2 - yx - xy$
$R[9] = -x^4 - x$
$R[10] = -x^2yx - x^3y - yxy - xy^2$
$R[11] = 2y$
$R[12] = 2x$

With respect to this basis, $(xy-yx)^4 - (xy-yx) = F(xy-yx)$ reduces to xy-yx in 118 reduction steps (it can be written as a sum, having 118 terms, in the polynomials above).

Lemma: If $a^n = a\ \forall a \in R$ ($n > 1$) then $a^{n-1}b - ba^{n-1}\ \forall a,b \in R$

proof: Notice first that if $zw = 0$ for some z,w then also $wz = 0$ because $wz = (wz)^n = wzwz...wz = 0$. Now $a(a^{n-1}b - b) = 0$, so we have $(a^{n-1}b - b)a = a^{n-1}ba - ba = 0$. Thus $a^{n-1}ba^{n-1} = ba^{n-1}$. Similarly from $(ba^{n-1} - b)a = 0$ we deduce $a^{n-1}ba^{n-1} = a^{n-1}b$.

In terms of the size of the input and the size of the resulting R-Basis, this would not seem to be much more difficult than the a^3-a case. In fact, both computationally and by hand a^4-a is a considerably harder problem. The Basis computation in the case of cubes takes a matter of minutes (on a Pentium 135 Mh machine). The corresponding computation for fourth powers takes several hours. In the case of cubes, there were 404 critical pairs generated (398 of which reduced to zero). The longest intermediate polynomial had 6 terms and the largest number of factors for any term (length of largest word) was 5. In the case of fourth powers there were 2307 critical pairs generated (2278 reduced to zero). The longest intermediate polynomial had 46 terms and the largest word size was 10.

Notice that the relation xy-yx does not appear in the R-Basis -- it was obtained by reducing F(xy-yx). Presumably it would appear in the basis if this relation were added initially. When this was tried, the reduction algorithm was still running after two days.

Theorem 8: Let R be a ring which $a^2 = a \in Z(R)\ \forall a \in R$ (where Z(R) is the center of R). Show that R is commutative

proof: Let F(a,b) = (aa-a)b-b(aa-a). We start with F(x,y), F(y,x), F(y+x,x), F(y+x,y). The relation xy-yx is instantly obtained as the sole member of the R-Basis. In fact -yx + xy = F(y+x,y)-F(x,y)+F(y,x) so it is obtained when the starting basis is interreduced.

Theorem 9: Let R be a ring which $a^3 = a \in Z(R)\ \forall a \in R$ (where Z(R) is the center of R). Show that R is commutative

proof: Let F(a,b) = (aaa-a)b-b(aaa-a). We start with F(-yx+xy,x), F(-yx+xy,y), F(-xy+x,x), F(-xy+x,y), F(-yx+x,x), F(-yx+x,y), F(-xy+y,x), F(-xy+y,y), F(-yx+y,x), F(-yx+y,y), F(yx,x), F(yx,y), F(xy,x), F(xy,y), F(-y+x,x), F(-y+x,y), F(y+x,x), F(y+x,y), F(y,x), F(y,y), F(x,x), F(x,y). The resulting R-Base is:

$R[1] = 2\,yx - 2\,xy$
$R[2] = -\,yx^2yx - x^2yxy + 2\,x^3y^2 - yx^2y + x^2y^2 - yx + xy$
$R[3] = -\,yx^3 + x^3y - yx + xy$
$R[4] = -\,x^3yx + x^4y - yx^2 - xyx + 2\,x^2y - yx + xy$
$R[5] = -\,xyx^2 - x^2yx + 2\,x^3y - yx^2 + x^2y$
$R[6] = -\,y^2x - yx^2 + xy^2 + x^2y - yx + xy$
$R[7] = -\,yx^2y^2 + x^2y^3 - yxy + xy^2 - yx + xy$
$R[8] = -\,yxy^2 - yx^2y + xy^3 + x^2y^2 - yxy - yx^2 + xy^2 + x^2y - yx + xy$
$R[9] = -\,(yx)^2 - (xy)^2 + 2\,x^2y^2$

Notice that xy - yx is not in this basis, although 2xy - 2yx is. The polynomial $F(x^2-y^2,x) - F(x^2-y^2,y)$ reduces to xy - yx and so the theorem is proved.

As in the case of a^4-a, we obtain a proof by reducing additional ideal elements after computing an R-Basis. In the current proof, the polynomials F(-yy+xx,x) and F(-yy+xx,y) were reduced by this basis to obtain the result. It is possible to compute an R-Basis with these two polynomials added to the starting set. When this is done the R-basis reduces to xy-yx but the time required for the computation increases dramatically. The above computation took about 17 minutes, produced only 9 new polynomials, the longest of which has 34 terms. When the starting basis is augmented with F(-yy+xx,x) and F(-yy+xx,y) the computation takes over a day, 23 new polynomials are produced, the longest having 77 terms. It has been found that adding more elements to a starting basis can dramatically increase the computation time. As a rule of thumb, the starting basis should be kept as small as possible.

R-Basis Algorithm

The algorithm used for computing R-Bases was obtained by modifying Mora's algorithm [Mora] (for polynomials in non-commuting variables with coefficients in a field). Our treatment of the integer coefficients follows the algorithm given by Buchberger [Buch] for the case of commuting variables.

We place a total ordering, $\prec$, on the integers so that $0 \prec -1 \prec 1 \prec -2 \prec 2$ ... We define the quotient in the Euclidean division process so that the remainder is smallest in this ordering.

Definition 1: If $a, b \in \mathbb{Z}$ let
$Q(a,b) = 0$ if $b = 0$ and
$Q(a,b) = q$ for which $r = a - bq$ is smallest in the ordering $\prec$

In any particular computer implementation, integer division is either floored or symmetric. Q(a,b) can be defined in terms of whatever quotient is provided by the language implementation. Let q be the (implementation dependent) value given by a / b. If a and b have the same sign then Q(a,b) will either be q or q-1. If a and b have opposite signs then Q(a,b) will be q or q+1.

Definition 2: (Buchberger) Let $a, b \in \mathbb{Z}$. The least common reducible of a and b, denoted LCR(a,b), is the smallest integer n (in the ordering $\prec$) so that Q(n,a) and Q(n,b) are non-zero.

Let L be the function which maps 0,1,2,3,... to 0,-1,1,-2,2,.. We have L(a)=a/2 if a is even and = -(a+1)/2 if a is odd. The numbers x so that Q(x,a)=0 are those which are less than L(a) in the ordering $\prec$. Thus LCR(a,b) = max(L(a),L(b)) where the maximum is taken using $\prec$.

In the ring $\mathbb{Z}<x_1,..,x_n>$ we use the graded lexicographic ordering on the words in the variables (monomials) with the ordering $x_1 < x_2 < .. < x_n$. This ordering combined with the ordering $\prec$ produces a well-ordering on the terms of polynomials: $\alpha X < \beta Y$ if either $X < Y$ in the graded lexicographic ordering of the monomials or if $X = Y$ and $\alpha \prec \beta$. We will always assume that polynomials are written so that all terms with the same monomial are added together. Thus every non-zero polynomial, f, has a unique term of highest order.

Definition 3: If $f \neq 0$ we let LT(f) denote the term of highest order. LC(f) will denote the coefficient and LM(f) the monomial part of LT(f).

Definition 4: We say that the word Y divides the word X (written Y | X) if there are words S and T so that X = S·Y·T.

Definition 5: Let f,g,p ∈ Z<x_1,..,x_n>. We say that f is reducible to g modulo p, and we write $f \xrightarrow[p]{} g$ if g is obtained by reducing a term of f. This means that there is a term αX of f so that LM(p) | X, and g = f - γSpT where X = S·LM(p)·T and where γ = Q(α,LC(p)) ≠ 0.

This is what Becker and Weispfenning [Beck] refer to as E-Reduction. Notice that reduction may not eliminate the term αX, but it does replace it by terms which are lower in order. As a result, reduction is Noetherian.

Examples: $f \xrightarrow[p]{} g$

	f	**p**	**g**
1	5y	2y	-y
2	yyy + xyx	2x + 1	yyy - xyx - yx
3	yyy + xyx	2x + 1	yyy - xyx - xy
4	yyy - xyx	2x + 1	yyy - xyx

Notice that example 2 gives a non-trivial reduction because Q(1,2) = 1 while example 4 produces a trivial reduction since Q(-1,2) = 0. The two possibilities given in examples 2 and 3 arise from the two positions in which LM(p) occurs in the second term. Our algorithms have been implemented to search the terms of f in descending order and, within a term, to look for a factor starting from the left. Thus our software produces example 2 as the result of a one step reduction of f by p.

Definition 6: Let P be a set of polynomials.
$f \xrightarrow[P]{} g$ will mean $f \xrightarrow[p]{} g$ for some p∈P.

For computational purposes we will usually apply reduction repeatedly until we obtain an irreducible normal form. Since reduction always replaces a term in f by terms of smaller order, repeated reduction always leads to a polynomial which is irreducible (no further non-trivial reductions can be applied). The set P will usually be understood from the context. Thus in the descriptions of algorithms, we will write f → g to indicate that g is obtained by a sequence of reductions from f . We will write g = NForm(f,P) if f → g and g is irreducible (with respect to the given set P). In general NForm(f,P) is not unique but an algorithm for calculating it yields a deterministic (implementation-dependent) result.

In the classical theory, given polynomials f_1 and f_2 we form a critical pair (p_1,p_2) by reducing some monomial X by f_1 and f_2. We then add p_1-p_2 to the basis under construction if it is non-zero. If the polynomial variables commute and the coefficients lie in a field, the monomial part of X is the least common multiple of LM(f_1) and LM(f_2) and the coefficient of X may be taken to be 1. If the variables commute and the coefficients are integers, Buchburger chooses the coefficient for X to be the smallest integer which has non-zero quotients upon division by both LC(f_1) and LC(f_2). If the variables do not commute, there is no unique least common multiple of monomials. Mora shows how to choose several candidates for the monomial part of X based on the concept of "matches". In the algorithm discussed here, the monomial part of X is chosen for each match (following Mora's Algorithm) and the coefficient is chosen as the “least common reducible” (following Buchberger’s algorithm for commuting variables).

Definition 7: (Mora) Let M_1 and M_2 be words in the polynomial variables. A (non-trivial) match for (M_1,M_2) is a 4-tuple of words (L_1,L_2,R_1,R_2) which satisfy one of the following conditions:

(1) $L_1 = R_1 = 1$, $M_1 = L_2M_2R_2$
(2) $L_2 = R_2 = 1$, $M_2 = L_1M_1R_1$
(3) $L_1 = R_2 = 1$, $L_2 \neq 1$, $R_1 \neq 1$, there is a $W \neq 1$ with $M_1 = L_2W$, $M_2 = WR_1$
(4) $L_2 = R_1 = 1$, $L_1 \neq 1$, $R_2 \neq 1$, there is a $W \neq 1$ with $M_1 = WR_2$, $M_2 = L_1W$

Mora has shown that it is sufficient to use non-trivial matches in his algorithm for computing Gröbner bases for non-commutative polynomials with coefficients in a field. For integer coefficients, Pritchard [Prit] has pointed out that non-trivial overlaps are not enough (see discussion below). We will therefore also allow the trivial match:

(5) $L_1 = R_2 = 1$, $L_2 = M_1$, $R_1 = M_2$

These conditions make $X = L_1M_1R_1 = L_2M_2R_2$ a common multiple of M_1 and M_2 which is minimal in some sense. For a non-trivial match to exists either one of the two words is a sub-word of the other, or there is a non-trivial overlap of the start of one word with the end of the other. Here are some examples of matches:

M_1	M_2	L_1	R_1	L_2	R_2	
x	y	1	y	x	1	(5)
xyxx	xxyx	x	1	1	x	(4)
xyxx	xxyx	xxy	1	1	yxx	(4)
xyxx	xxyx	1	yx	xy	1	(3)
xyxx	xxyx	1	xyx	xyx	1	(3)
xyxx	xxyx	1	xxyx	xyxx	1	(5)
xx	xx	x	1	1	x	(4)
xx	xx	1	x	x	1	(3)
xyx	y	1	1	x	x	(1)

Let $m = (f_1,f_2, L_1,L_2,R_1,R_2)$ where (L_1,L_2,R_1,R_2) is a match for $(LM(f_1),LM(f_2))$.
Set $X = L_1LM(f_1)R_1 = L_2LM(f_2)R_2$ and $\alpha = LCR(LC(f_1),LC(f_2))$. We notice that αX can be reduced by f_1 and f_2.

Definition 8: We define $CritPair(m) = (p_1,p_2)$ as the pair of reductions of αX by f_1 and f_2. Specifically $p_i = \alpha X - Q(\alpha,LC(f_i))L_if_iR_i$

Procedure 1: Interreduce
Input: A finite set, G, of polynomials
Output: An interreduced set, F, generating the same ideal

```
Changed? := true;  F := G
```

```
WHILE  Changed?  DO
      Changed? := false;  G := F;  F := $\emptyset$
      WHILE  not Empty(G)  DO
            Select $g \in G$
            $G := G - \{g\}$
            $f := \mathrm{NForm}(g, F \cup G)$
            IF  $f \neq g$ THEN  Changed? := true
            IF  $f \neq 0$ THEN  $F := F \cup \{f\}$
```

Procedure 2: R-Basis Algorithm

Input: A finite set, G, of polynomials

Output: (If the algorithm terminates) an R-Basis for the ideal generated by G.

```
H := G
WHILE  not Empty(H)   DO
        $B := \{ (f_1,f_2,l_1,r_1,l_2,r_2) \mid f_1 \in G, f_2 \in H;$
                        $(l_1,r_1,l_2,r_2)$ a match for $(LM(f_1),LM(f_2)) \}$
        $H := \emptyset$
        WHILE   not Empty(B)      DO
                select  $m \in B$;     $B := B - \{m\}$
                $(p_1,p_2) := \mathrm{CritPair}(m)$
                $p_1 := \mathrm{NForm}(p_1, G \cup H)$;  $p_2 := \mathrm{NForm}(p_2, G \cup H)$
                $f := p_1 - p_2$
                IF   $f \neq 0$  THEN   $H := H \cup \{f\}$
        $G := G \cup H$
G := Interreduce(G)
```

Since R-Bases can be infinite in the non-commutative setting, this algorithm follows the Mora algorithm [Mora] in making sure that every match is ultimately processed: matches are not formed with newly generated polynomials (saved in list H) until the B-list is empty. Any newly generated polynomial is reduced by the current basis. In this version of the algorithm, however, previous basis elements are not reduced by the newly generated ones until the interreduction step at the end. This is done to make sure that the information in the B-list still correctly corresponds to the state when it was generated.

Performance Modifications

In this section we make some observations based on profiling the behavior of these procedures when used for the examples given above. As in the case of coefficients in a field, the non-commutative version of the basis algorithm need not terminate in general. In all of our examples, however, the algorithm did terminate leaving a finite basis. Our examples exhibited "intermediate expressions swell": polynomials often appeared in the midst of computation having a large number of terms -- only to be subsequently reduced to smaller polynomials. On average, the number of steps needed to reduce a polynomial to normal form appears to be much larger than what we have typically seen when using coefficients in a field.

The procedure actually used for the examples has one modification from the version presented above: we applied interreduction after each new basis element is generated. This appeared to shorten the execution time by decreasing the size of intermediate polynomials and eliminating some earlier polynomials. Since the bases for these examples proved to be finite, we did not attempt to modify the B-list after interreduction -- but instead we started the procedure again at the top. Note that this should not be done in cases where the R-basis is infinite: it may result in some matches not being processed.

Is an R-Basis a Gröbner Basis?

Mora [Mora] proved that for the case of non-commutative polynomials with coefficients in a field, it is enough to consider non-trivial matches to obtain a Gröbner Basis. Pritchard [Prit] observes that non-trivial matches may not be sufficient to produce a Gröbner Basis in the case of integer coefficients. Here are a simplified versions of Pritchard's example:

Let $f_1 = 2x - a$, $f_2 = 2y - b$ where we assume graded lexicographic ordering with the letters a,b,c,.... in increasing order. Neither of these polynomials can be reduced with respect to the other. There are no non-trivial matches of the leading monomials. If the Basis algorithm uses only non-trivial matches it would terminate leaving $\{f_1,f_2\}$ as the basis. However $f_1y - xf_2 = xb - ay$ is in the ideal and is irreducible with respect to this basis. The problem is that x and y cannot be reduced by the leading terms 2x and 2y as would be the case in a field of characteristic $\neq 2$.

This particular example can be handled by allowing the trivial match in the R-Basis procedure. The R-Basis is $f_1 = 2x\text{-}a$, $f_2 = 2y\text{-}b$, $f_3 = -xa-ax+a^2$, $f_4 = -xb-ay+ab$, $f_5 = -ya-bx+ba$, $f_6 = -yb-by+b^2$. If is not clear if $\{f_1,..,f_6\}$ is a Gröbner Basis, but it does appear to reduce polynomials of the form $f_1Ty - xTf_2$ where T is a term in x,y,a,b.

For $f_1 = 9x\text{-}a$, $f_2 = 15y\text{-}b$ we obtain an infinite R-basis which does seem to reduce tested polynomials of the form $5f_1Ty - 3xTf_2$. Pritchard's example is $f_1 = 9xww\text{-}u$, $f_2 = 15zyy\text{-}v$. In this case we also obtain an infinite R-Basis. There are, however, some instances of $h = 3xwwTf_2 - 5f_1Tzyy$ which appear not to be reduced to zero (at least not by the truncation of this basis which we tested).

We do know that some of our R-bases are in fact Gröbner Bases because they reduce the situation to either the commutative case (in which case Buchberger's work applies) or to coefficients over a field (in which case Mora's work applies). The R-Basis procedure does, in fact, provide a condition for ideal membership sufficient for the examples we considered. For these examples the addition of a trivial match (case (5)) was also found unnecessary.

In Search of Algorithms

The context of polynomial rings with non-commuting variables problems provide difficulties which do not arise in the commutative case. The ideal membership problem is related to the "word problem" and is therefore undecidable in general. Gröbner Bases can be infinite. Nevertheless, the use of the machinery of reduction and rewrite rules in the non-commutative case can be useful. In [Wav] and [HWS] we show how even infinite Gröbner Bases can be useful in the automatic simplification of some of the complex matrix and operator expressions which arise in engineering mathematics. In this paper we have show how this technology, using polynomials with integer coefficients, can be useful in the study of polynomial identities in non-commutative rings. It should be noted that the use of integer coefficients is essential for this work.

The speed of the R-Basis algorithm appears to depend on many factors such as the order in which basis polynomials are used in reduction; and when and how to reduce existing basis polynomials by newly generated ones. The R-Basis procedure in its current form was observed to produce a very large proportion of critical pairs which reduce to zero. There is a need, therefore, for more efficient versions of algorithms which manipulate polynomials with integer coefficients.

The ideal of consequences of an identity requires the production of a potentially infinite collection of polynomials by making substitutions for the variables in the initial identity. In our proofs a finite set of starting substitutions were chosen ad hoc. Substitutions were used which seemed to be used in "hand" proofs or which trial runs showed might simplify some of the basis elements which were appearing. As noted when discussing the proofs, it is not a good strategy to add an overabundance of starting polynomials. There is a need for heuristics to automate the selection of starting substitutions for problems of this type.

Machine proofs using rewrite rules differ from "hand" proofs even though both the machine and human are performing essentially the same type of manipulations and using the same set of rules. For automated computation with reduction rules, each rule is given a "handed-ness" by the choice of ordering. Thus, for example, a human has the flexibility of applying a rule like $y^2x = xy^2$ in whichever direction seems appropriate to the goal. By contrast, the automated procedures will (given the ordering used in this paper) only use $y^2x \to xy^2$. The mathematician can also choose where, within a term, to apply a rule. In the automated computation this choice is part of the implementation design (in our implementation a rule is applied in the first position where it is applicable, scanning left to right). A human can choose to only manipulate expressions which are relevant to the goal. The machine computation may automatically process a large number of expressions that are not really needed for the proof. There is a need for algorithms which allow the computer to use these tools more effectively and efficiently.

To illustrate the difference between machine and human proofs, here is the most elegant proof I have seen of the $a^4 = a$ theorem. It should be compared with the proof of Theorem 7 above (which took several hours of machine computation and processed several thousand critical pairs):

Theorem: If R is a ring so that $a^4 = a \ \forall a \in R$, then R is commutative

proof: (due to John Hunter and David Ferguson)

1. *If $xy = 0$ then $yx = xy$* because $yx = (yx)^4 = 0$
2. *Cubes are in the center*
   $y(y^3x-x) = 0$ so, by (1), $(y^3x-x)y = 0$
   thus $y^3xy = xy$ and $y^3xy^3 = xy^3$
   Similarly, $(xy^3-x)y = 0$ so $yxy^3 = yx$
   and $y^3xy^3 = y^3x$. Thus we find that $xy^3 = y^3x$
3. *$a = -a$* because $a = a^4 = (-a)^4 = -a$
4. *$(a+a^2)^n = a+a^2$* (we will use this for n=3)
   $(a+a^2)^2 = a^2 + 2a^3 + a = a+a^2$ by (3)
   the general result is by induction
5. *$(a+b^2) + (a+b^2)^2$ is a cube by (4)*
   so $a[(a+b^2) + (a+b^2)^2] = [(a+b^2) + (a+b^2)^2]a$ using (2)
   expand $ab^2 + a^2b^2 + ab = b^2a + b^2a^2 + ba$

or $(a+a^2)b^2 + ab = b^2(a+a^2) + ba$

However $a+a^2$ is a cube, so it commutes with b^2.

Thus $ab = ba$

In contrast to the automated proof, the number of algebraic steps is far less and the size of "intermediate" polynomials much smaller. This proof depends, however, on making some observations (using the fact that cubes are in the center and making clever note of certain expressions which are cubes).

Footnote: John Hunter became interested in this problem while an undergraduate student and his first proof was a dozen pages long. He sent this shorter proof to me about 6 years later when he was an Assistant Professor [Hunt]. He said that he had worked on the problem from time to time and that this was the latest in a sequence of successively shorter proofs. The processing time for this proof was, therefore, fairly long.

References

[Beck] *Gröbner Bases: A Computational Approach to Commutative Algebra* by Thomas Becker and Volker Weispfenning, Springer-Verlag 1993

[Buch] *Gröbner Bases: an Algorithmic Method in Polynomial Ideal Theory* by B. Buchberger, Recent Trends in Multidimensional System Theory, Reidel 1985, pp. 184-232

[Her] *Noncommutative Rings*, by I. N. Herstein. Carus Mathematical Monographs Number 15, Amer Math Society 1968

[Her2] *Topics in Algebra 2nd Ed*, by I. N. Herstein, Xerox 1975

[Hunt] *A new proof for $a^4=a$*, by J. Hunter and D. Ferguson private communication, 1982

[HWS] *Computer Simplification of Formulas in Linear Systems Theory*, by J. W. Helton, M. Stankus and J. J. Wavrik, IEEE Transactions on Automatic Control 43 (1998) pp. 302-314

[Mora] *Groebner Bases for Non-Commutative Polynomial Rings*, by F. Mora, Lecture Notes in Computer Science number 229 (1986) pp. 353-362

[Prit] *The Ideal Membership Problem in Non-Commutative Polynomial Rings*, by F. Leon Pritchard, J Symbolic Computation 22 (1996) pp. 27-48

[Wav] *Rewrite Rules and Simplification of Matrix Expressions*, by J. J. Wavrik Computer Science Journal of Moldova 4 (1996), pp. 360-398

John J Wavrik
Department of Mathematics - 0112
Univ. of California - San Diego
La Jolla, CA 92093-0112

jjwavrik@ucsd.edu
http://math.ucsd.edu/~jwavrik