

NON-COMMUTATIVE POLYNOMIAL COMPUTATIONS

ARJEH M. COHEN

This report describes the theory behind the key algorithms in the package GBNP for non-commutative polynomial rings. In Section 1, we deal with non-commutative Gröbner bases. This enables us to construct quotient algebras of free algebras on a finite number of generators. In Section 2, based on work contributed by Jan Willem Knopper, we show how to use this set-up to construct modules for such quotient algebras. Finally, in Section 3, based on work contributed by Chris Krook, we demonstrate how to compute whether a quotient algebra is finite-dimensional when given a Gröbner basis. In the infinite-dimensional case, it is also shown how to analyze the growth of the quotient algebra.

1. GRÖBNER BASES

In [6] Mora described an algorithm which, if it terminates, returns a non-commutative Gröbner basis. Here we follow that paper to prove the correctness of the algorithm as implemented by Dié Gijssbers and the author in GAP, cf. [1]. The algorithm is the core algorithm of the GAP package GBNP for computing with non-commutative polynomials. Earlier versions of this section were written with the help of Dié Gijssbers.

1.1. Basic notions. We start with some definitions and lemmas inspired by [6, 2]. We work with finitely presented algebras with a unit element 1 over a field k . Let T be the free monoid on n generators, which will be denoted by x_1, \dots, x_n . Here n is a natural number. We consider the monoid ring $k\langle T \rangle$, which has an obvious k -vector space structure with basis T . The multiplication of the monoid ring is a bilinear extension of the multiplication in T , so T is a monomial basis. The elements of T are also called the *monomials* of $k\langle T \rangle$. The elements of $k\langle T \rangle$ are called *non-commutative polynomials*, and often just *polynomials*. The finitely presented algebras we mentioned earlier are quotients of $k\langle T \rangle$ by two-sided ideals that are finitely generated.

As usual in Gröbner basis theory, we need an appropriate ordering ' $<$ ' on the set T of monomials.

Definition 1. An ordering $<$ on T is called a *reduction ordering* if for all $t_1, t_2, l, r \in T$ with $t_1 < t_2$ we have $1 \leq lt_1r < lt_2r$.

An example of a reduction ordering is *deglex*, that is, 'total degree first, then lexicographic'. A very useful property of a reduction ordering is that if $a, b \in T$ are such that a divides b , denoted by $a \mid b$ (that is, there are $l, r \in T$ with $b = lar$), then $a \leq b$ (for, $1 \leq l$ and $1 \leq r$ imply $a = 1a \leq la = la1 \leq lar$).

Date: 18 November 2009.

With contributions by Dié Gijssbers, Chris Krook, and Jan Willem Knopper.

Definition 2. Given an ordering on T we can write each polynomial $f \in k\langle T \rangle$ in a unique way as a linear combination of monomials t_i :

$$f = \sum_{i=1}^s c_i t_i \quad \text{with } c_i \in k \setminus \{0\} \text{ and } t_i \in T, \text{ such that } t_1 > \cdots > t_s.$$

We call this decomposition the *ordered form* of f . The polynomial f is called *monic* if $c_1 = 1$.

For such an ordered form of f , we denote t_1 , the largest monomial with non-zero coefficient of f , by $L(f)$. We call it the *leading monomial* of f . By $lc(f)$ we denote the *leading coefficient* of f , that is, c_1 . The polynomial $c_1^{-1}f \in k\langle T \rangle$ is called the *monic version* of f .

Gröbner bases deal with finitely generated (two-sided) ideals in $k\langle T \rangle$. A generating set of an ideal is called a *basis* of that ideal. The ‘best approximation’ of the remainder of a division of a polynomial f by elements of a finite set G of elements of $k\langle T \rangle$ is a crucial notion, usually referred to as a normal form of f with respect to G .

Definition 3. Let $G \subset k\langle T \rangle$, and denote by I the ideal generated by G . A *normal form* of $f \in k\langle T \rangle$ with respect to G is an element $h \in k\langle T \rangle$ such that $f = h + \sum_{i=1}^t c_i l_i g_i r_i$ for certain $g_i \in G$, $c_i \in k$ and $l_i, r_i \in T$ ($i = 1, \dots, t$) with $l_i L(g_i) r_i \leq L(f)$, and either $h = 0$ or $L(g) \not\leq L(h)$ for all $g \in G$. We also say that f can be *reduced* to h with respect to G . If f is a normal form of itself with respect to G , then f is sometimes called *reduced* with respect to G .

Moreover, G is a *Gröbner basis* of I if G is a basis of I and if the leading monomial of each non-zero element of I is a multiple of the leading monomial of an element of G .

A normal form of a fixed polynomial in $k\langle T \rangle$ with respect G need not be unique. If G is a Gröbner basis, then each polynomial in $k\langle T \rangle$ has a unique normal form with respect to G .

The Gröbner basis of a fixed ideal depends on the choice of a reduction ordering. Taking the reduction ordering on T to be deglex, we have a well-founded ordering and hence a guarantee of termination of the normal form computation. We can think of `NormalForm(f, G)` as a deterministic algorithm if we view G as a list and, at every step, take the first element from the list whose leading monomial divides $L(f)$.

In the commutative case, S-polynomials are used for the Gröbner basis computation. In order to define their non-commutative analogues, we use obstructions.

Definition 4. Let $G = (g_i)_{1 \leq i \leq q}$ be a list of monic polynomials. An *obstruction* of G is a six-tuple $(l, i, r; \lambda, j, \rho)$ with $i, j \in \{1, \dots, q\}$ and $l, \lambda, r, \rho \in T$ such that $L(g_i) \leq L(g_j)$ and $lL(g_i)r = \lambda L(g_j)\rho$. For a given obstruction, we define the corresponding *S-polynomial* as

$$s(l, i, r; \lambda, j, \rho) = l g_i r - \lambda g_j \rho.$$

Our notation of an obstruction differs slightly from Mora’s in [6]. The symbols for the six parameters are equal though.

Gröbner basis computations consist mainly of adding to G normal forms of S-polynomials of elements from a basis. The next notion is needed to weed out obstructions whose S-polynomials are not going to contribute to the computation.

Definition 5. Given a list $G = (g_i)_{1 \leq i \leq q}$ of monic polynomials we call a polynomial f weak with respect to G if there are $c_i \in k$, $l_i, r_i \in T$ and $\nu(i)$ ($i = 1, \dots, t$) such that,

$$\text{for each } i \text{ we have } l_i L(g_{\nu(i)}) r_i \leq L(f) \quad \text{and} \quad f = \sum_{i=1}^t c_i l_i g_{\nu(i)} r_i.$$

We call an obstruction $(l, i, r; \lambda, j, \rho)$ of G weak if its S-polynomial $s(l, i, r; \lambda, j, \rho)$ is weak with respect to G .

The S-polynomial of a weak obstruction $(l, i, r; \lambda, j, \rho)$ can be written as follows with $c_h \in k$, $l_h, r_h \in T$, and $\nu(h) \in \{1, \dots, q\}$.

$$s = l g_i r - \lambda g_j \rho = \sum_h c_h l_h g_{\nu(h)} r_h \quad \text{with} \quad l_h L(g_{\nu(h)}) r_h \leq L(s) < l L(g_i) r.$$

Weakness can be tested effectively by solving a finite number of linear equations. If the normal form of f with respect to G is equal to 0, then f is weak with respect to G . The converse is not necessarily true, but see Theorem 1.5.

Observe that infinitely many obstructions can be found. The next lemma makes a lot of them superfluous.

Lemma 1.1. *Let $G = (g_h)_{h=1, \dots, q}$ be a list of monic polynomials in $k\langle T \rangle$ and $(l, i, r; \lambda, j, \rho)$ a weak obstruction of G . Then all obstructions $(l', i, r'; \lambda', j, \rho')$ with $l' = \omega_1 l$, $r' = r \omega_2$, $\lambda' = \omega_1 \lambda$, $\rho' = \rho \omega_2$ and ω_1, ω_2 monomials, are also weak.*

Proof. Denote by s the S-polynomial of $(l, i, r; \lambda, j, \rho)$. By the weak obstruction hypothesis we can write $s = l g_i r - \lambda g_j \rho = \sum_h c_h l_h g_{\nu(h)} r_h$ with $l_h L(g_{\nu(h)}) r_h \leq L(s)$. Now assume $(l', i, r'; \lambda', j, \rho')$ is an obstruction as in the hypotheses and denote by s' its S-polynomial. Then

$$s' = l' g_i r' - \lambda' g_j \rho' = \omega_1 (l g_i r - \lambda g_j \rho) \omega_2 = \omega_1 \sum_h c_h l_h g_{\nu(h)} r_h \omega_2 = \omega_1 s \omega_2.$$

Now $s' = \sum_h c_h l'_h g_{\nu(h)} r'_h$ with $l'_h = \omega_1 l_h$ and $r'_h = r_h \omega_2$. From $l_h L(g_{\nu(h)}) r_h \leq L(s)$ we obtain $l'_h L(g_{\nu(h)}) r'_h = \omega_1 l_h L(g_{\nu(h)}) r_h \omega_2 \leq \omega_1 L(s) \omega_2 = L(s')$ and so the obstruction is weak with respect to G . \square

Definition 6. Let G be a list of monic polynomials in $k\langle T \rangle$ and H a set of polynomials. An S-polynomial s is called *reducible* from H with respect to G if weakness with respect to G of all elements of H implies weakness of s with respect to G .

In particular, if s is weak then it is *reducible* from \emptyset .

If we have obstructions $(l, i, r; \lambda, j, \rho)$ and $(l', i, r'; \lambda', j, \rho')$ as in Lemma 1.1, then $s(l', i, r'; \lambda', j, \rho')$ is reducible from $\{s(l, i, r; \lambda, j, \rho)\}$.

Observe that an obstruction $(l, i, r; \lambda, j, \rho)$ is weak if $L(g_i)$ and $L(g_j)$ have no overlap. This is our next focus (see [6, Lemma 5.4]).

Definition 7. For $b \in T$, two monomials $t_1 \leq t_2$ in T are said to have *overlap* b if there are $a, c \in T$ such that $t_1 = ab$ and $t_2 = bc$ or $t_1 = ba$ and $t_2 = cb$ or $t_1 = b$ and $t_2 = abc$. If 1 is the only overlap between t_1 and t_2 , the two monomials t_1 and t_2 are said to have *no overlap*.

An obstruction $(l, i, r; \lambda, j, \rho)$ is said to have *no overlap* if $L(g_i)$ and $L(g_j)$ do not overlap in $l L(g_i) r$, in the sense that there is a $w \in T$ such that $l L(g_i) r = l L(g_i) w L(g_j) \rho$ or $l L(g_i) r = \lambda L(g_j) w L(g_i) r$.

Clearly, if $L(g_i)$ and $L(g_j)$ have no overlap, then every obstruction $(l, i, r; \lambda, j, \rho)$ with $l, r, \rho, \lambda \in T$ has no overlap. The converse is not true: if $L(g_i) = x_1x_2$ and $L(g_j) = x_2x_3$, then the overlap between these monomials is x_2 whereas the obstruction $(1, i, x_2^a x_3; x_1x_2^a, j, 1)$ has no overlap for each $a > 0$.

Lemma 1.2. *Suppose u, v , and w are monomials in T . If $uw = wv$, then there exist $x, y \in T$ and $a \in \mathbb{N}$ with $u = xy$, $v = yx$ and $w = (xy)^a x$.*

Proof. Induction on the length of w . □

Lemma 1.3. *Suppose, for certain i and j with $i \neq j$ we have $L(g_i) = L(g_j)$. Then, for each k , each S -polynomial $s(l, j, r; \lambda, k, \rho)$ is reducible from the S -polynomials $s(l', i, r'; \lambda', j, \rho')$ and $s(l'', i, r''; \lambda'', k, \rho'')$.*

Proof. Immediate from

$$s(l, j, r; \lambda, k, \rho) = s(l, i, r; \lambda, k, \rho) - s(l, i, r; l, j, r).$$

□

1.2. Reducibility. Throughout this section we take G to be a list $(g_h)_{h=1, \dots, q}$ of monic polynomials in $k\langle T \rangle$.

Lemma 1.4. *Every obstruction without overlap is reducible from an S -polynomial with overlap with respect to G .*

Proof. Let $b = (l, i, r; \lambda, j, \rho)$ be an obstruction and denote by s its S -polynomial. Then $lL(g_i)r = \lambda L(g_j)\rho$. Suppose that b has no overlap. Then there exists $w \in T$ such that either $r = wL(g_j)\rho$ or $l = \lambda L(g_j)w$. Assume the former (the proof for $l = \lambda L(g_j)w$ being quite similar). Then also $\lambda = lL(g_i)w$ and by Lemma 1.1 $b = (l, i, wL(g_j)\rho; lL(g_i)w, j, \rho)$ is reducible from $(1, i, wL(g_j); L(g_i)w, j, 1)$. Therefore, we may and shall assume $l = \rho = 1$.

In the ordered form, the two polynomials g_i, g_j can be written as $g_i = \sum_h c_h t_h$ and $g_j = \sum_p d_p u_p$ for $h = 1, \dots, m_i$ and $p = 1, \dots, m_j$ with $t_h, u_p \in T$ and $c_h, d_p \in k \setminus \{0\}$ such that $t_h > t_{h+1}$ and $u_p > u_{p+1}$. Thus $L(g_i) = t_1$ and $L(g_j) = u_1$. Now

$$\begin{aligned} s &= g_i r - \lambda g_j \\ &= g_i w L(g_j) - L(g_i) w g_j \\ &= g_i w \left(g_j - \sum_{p=2}^{m_j} d_p u_p \right) - \left(g_i - \sum_{h=2}^{m_i} c_h t_h \right) w g_j \\ &= \sum_{h=2}^{m_i} c_h t_h w g_j - \sum_{p=2}^{m_j} d_p g_i w u_p. \end{aligned}$$

To prove that s is weak, it remains to verify that all monomials occurring on the right hand side are less than or equal to $L(s)$. This can only go wrong when the leading monomials, say $t_2 w L(g_j)$ and $L(g_i) w u_2$ of the two summations on the right hand side cancel each other:

$$c_2 t_2 w u_1 = d_2 t_1 w u_2.$$

Since $t_2 < t_1$ and $u_2 < u_1$ this only occurs if $c_2 = d_2$ and there are $v_1, v_2 \in T$ such that $t_1 = t_2 \cdot v_1$ and $u_1 = v_2 \cdot u_2$ with $v_1 w = w v_2$. By Lemma 1.2 the only non-trivial solutions are those with $v_1 = xy$, $v_2 = yx$ and $w = (xy)^a x$ for some monomials x and y and $a \in \mathbb{N}$. Without loss of generality we may assume that $y \neq 1$ (for otherwise the leading monomials will have overlap x).

For brevity we denote the obstruction with $w = (xy)^a x$ by $o(a)$. We expand g_i and g_j and find

$$\begin{aligned} g_i &= t_1 + \sum_{h=2}^{m_i} c_h t_h &= t_2 \cdot xy + c_2 t_2 + \sum_{h=3}^{m_i} c_h t_h \\ g_j &= u_1 + \sum_{p=2}^{m_j} d_p u_p &= yx \cdot u_2 + d_2 u_2 + \sum_{p=3}^{m_j} d_p u_p \end{aligned}$$

Expansion of the summations on the right hand side in the previous expansion of s gives

$$\begin{aligned} s &= \sum_{h=2}^{m_i} c_h t_h w g_j - \sum_{p=2}^{m_j} d_p g_i w u_p \\ &= c_2 t_2 w g_j - d_2 g_i w u_2 + \sum_{h=3}^{m_i} c_h t_h w g_j - \sum_{p=3}^{m_j} d_p g_i w u_p \\ &= c_2 (t_1 (xy)^{a-1} x g_j - g_i (xy)^{a-1} x u_1) + \sum_{h=3}^{m_i} c_h t_h w g_j - \sum_{p=3}^{m_j} d_p g_i w u_p \end{aligned}$$

so, writing $s(a-1)$ for the S-polynomial of $o(a-1)$, we have

$$s = c_2 s(a-1) + \sum_{h=3}^{m_i} c_h t_h w g_j - \sum_{p=3}^{m_j} d_p g_i w u_p.$$

If again the leading terms $t_3 w u_1$ and $t_1 w u_3$ of the summations cancel each other then t_1 and u_1 are not only multiples of t_2 and u_2 but also multiples of t_3 and u_3 . Then their corresponding polynomials can be extracted in the same way as done for t_2 and u_2 . After a finite number m of these extractions we find

$$s = c_2 s(a-1) + c_3 s(a-2) + \dots + \sum_{h=m+2}^{m_i} c_h t_h w g_j - \sum_{p=m+2}^{m_j} d_p g_i w u_p.$$

Now $t_{m+2} w L(g_j)$ or $L(g_i) w u_{m+2}$ with $c_{m+2} \neq 0$ or $d_{m+2} \neq 0$ respectively, is the leading term of the two summations on the right.

Assume that all $s(a-b)$ are weak for $1 \leq b \leq a$. Then we can express every $s(a-b)$ as a sum of polynomials $l g_h r$ with leading term $l L(g_h) r$ less than or equal to $L(s(a-b))$. By the shape of $s(a-b)$ we know that $t_h (xy)^{a-b} x L(g_j) = L(g_i) (xy)^{a-b} x u_p$ for all t_h and t_p that cancelled in the previous summations. So $L(s(a-b))$ is at most $t_{m+2} (xy)^{a-b} x L(g_j)$ or $L(g_i) (xy)^{a-b} x u_{m+2}$.

This means that if all $s(a-b)$ are weak then $t_{m+2} w L(g_j)$ or $L(g_i) w u_{m+2}$ is the leading term of $s = s(a)$ and all monomials on the right hand side are less than or

equal to $L(s)$, proving that $s(a)$ is weak. So $s(a)$ is reducible from $\{s(a-b) \mid 1 \leq b \leq a\}$. By recursion now all $s(a)$ are reducible from $s(0)$.

Since reducibility is a transitive relation, it remains to show that $s(0)$ is reducible from an S-polynomial with overlap. Using the same substitution as before we find $s(0) = g_i x u_2 - t_2 x g_j$. But $t_1 = t_2 \cdot xy$ and $u_1 = yx \cdot u_2$, so the obstruction $o(0)$ has an overlap because $y \neq 1$. \square

As a consequence, if an S-polynomial $s(l, i, r; \lambda, j, \rho)$ is not weak with respect to G , then the leading monomials of the two polynomials involved have an overlap.

Next we show that, although the set of all S-polynomials of G can be infinite, we can restrict our analysis of obstructions to a finite set.

Definition 8. A set H of polynomials is called *basic for G* if every S-polynomial of G is reducible from H with respect to G .

Theorem 1.5. For a finite list G of polynomials generating an ideal I of $k\langle T \rangle$, the following statements are equivalent.

- (i) G is a Gröbner basis.
- (ii) The normal form of each polynomial of I is equal to 0.
- (iii) Each S-polynomial of G is weak with respect to G .
- (iv) The empty set is a basic set for G .

Proof. Write $G = (g_i)_{1 \leq i \leq q}$.

(i) \implies (ii). Suppose G is a Gröbner basis and $f \in I$. If $f = 0$ we are done. Otherwise, let i be the first index $\{1, \dots, q\}$ for which $L(g_i)$ divides $L(f)$. Then there are $a, b \in T$ such that $aL(g_i)b = L(f)$. Now $f - ag_ib \in I$. By induction with respect to the reduction ordering $<$, we have $\text{NormalForm}(f - ag_ib, G) = 0$. As $f - ag_ib$ is the first step of the NormalForm algorithm for f , this implies $\text{NormalForm}(f, G) = 0$, as required.

(ii) \implies (iii). Suppose $s = s(l, i, r; \lambda, j, \rho)$. By (ii), $\text{NormalForm}(s, G) = 0$, so s is weak.

(iii) \Leftrightarrow (iv). This is immediate from the definition of basic set.

(iii) \implies (i). Suppose $f \in I$ and $L(f)$ is not contained in the ideal generated by all $L(g_i)$. Without loss of generality, we may take $L(f)$ to be minimal with these properties as well as the maximum t over all leading monomials in an expression of f as a linear combination of summands $a_h g_{\nu(h)} b_h$ with $a_h, b_h \in k\langle T \rangle$ and $g_{\nu(h)} \in G$. By an argument as in Lemma 1.4, there are at least two indices i and j such that $a_i g_{\nu(i)} b_i$ and $a_j g_{\nu(j)} b_j$ are distinct and occur in an expression of f as a linear combination of multiples of the elements of G and satisfy $t = L(a_i g_{\nu(i)} b_i) = L(a_j g_{\nu(j)} b_j) > L(f)$. Consider the S-polynomial $s = s(L(a_i), \nu(i), L(b_i); L(a_j), \nu(j), L(b_j))$. By (iii), s is weak. Let w be an expression of s as a sum of multiples of g_h whose leading terms are smaller than t and consider the following expression

$$f = \text{lc}(a_i b_i) \text{lc}(a_j b_j)^{-1} a_j g_{\nu(j)} b_j + \text{lc}(a_i b_i) w + \sum_{h \neq i, j} a_h g_{\nu(h)} b_h.$$

This expression for f has fewer summands with leading term equal to t . Continue this way until this number is at most 1. Then we reach a contradiction. We conclude that G is a Gröbner basis. \square

Lemma 1.6. *If G is finite, there is a finite basic set H of S -polynomials of G . Moreover, H can be chosen so that every S -polynomial of G in H corresponds to an obstruction $(l, i, r; \lambda, j, \rho)$ with overlap and with at least one of the two parameters $\{l, \lambda\}$ and one of $\{r, \rho\}$ equal to 1.*

In fact, such an H can be found from the knowledge of the leading monomials of elements of G only.

Proof. Let $(l, i, r; \lambda, j, \rho)$ be an obstruction and write $s = s(l, i, r; \lambda, j, \rho)$ for its corresponding S -polynomial. Write $L(g_i) = m_1 \cdots m_p$ and $L(g_j) = n_1 \cdots n_q$ with m_u and n_v monomials in T of degree 1 (that is, of the form x_l for some $l \in \{1, \dots, n\}$). Recall that $L(g_i) \leq L(g_j)$ so $p \leq q$.

From Lemma 1.4 we know that if s is not weak, then it must have some overlap. In particular, $L(g_i)$ and $L(g_j)$ must have overlap. This can occur in three ways:

$$\begin{aligned} m_1 \cdots m_h &= n_{q-h+1} \cdots n_q & 1 \leq h < p, \\ n_1 \cdots n_h &= m_{p-h+1} \cdots m_p & 1 \leq h < p, \text{ or} \\ m_1 \cdots m_p &= n_{h+1} \cdots n_{h+p} & 1 \leq h < q - p. \end{aligned}$$

In particular, for every two polynomials the number of possible overlaps is finite. We show that H needs to contain at most one S -polynomial for every overlap. This suffices to prove the lemma.

Assume that $L(g_i)$ and $L(g_j)$ have a nontrivial overlap. To satisfy the equation $lL(g_i)r = \lambda L(g_j)\rho$, the factors of $L(g_i)$ that are not in the overlap have to be in l or ρ and, similarly, the parts of $L(g_j)$ that are not in the overlap have to be in l or r . So for every obstruction corresponding to some overlap the monomials $lL(g_i)r$ and $\lambda L(g_j)\rho$ have to be equal to $l'\omega r'$ and $\lambda'\omega\rho'$, respectively, with ω equal to

$$\begin{aligned} \omega &= n_1 \cdots n_{q-h} L(g_i) &= L(g_j) m_{h+1} \cdots m_p \\ \omega &= L(g_i) n_{h+1} \cdots n_q &= m_1 \cdots m_{p-h} L(g_j) \\ \omega &= n_1 \cdots n_h L(g_i) n_{h+p+1} \cdots n_q &= L(g_j) \end{aligned}$$

in the respective cases. Now by Lemma 1.1 these obstructions are weak except when $l' = r' = \lambda' = \rho' = 1$. So for every possible overlap there exists a single S -polynomial such that all other obstructions are reducible from it with respect to $\{g_i, g_j\}$; in the respective cases, the corresponding obstruction is

$$\begin{aligned} &(n_1 \cdots n_{q-h}, i, 1; 1, j, m_{h+1} \cdots m_p) \\ &(1, i, n_{h+1} \cdots n_q; m_1 \cdots m_{p-h}, j, 1) \\ &(n_1 \cdots n_h, i, n_{h+p+1} \cdots n_q; 1, j, 1) \end{aligned}$$

This means that s need only be in H if at least one of the two parameters l and λ is equal to 1 and at least one of the two parameters r and ρ is equal to 1. \square

If, in the above setting, $l = r = 1$, then, as $L(g_i) \leq L(g_j)$, we must have $L(g_i) = L(g_j)$, and so $\lambda = \rho = 1$ as well. We distinguish between three kinds of obstruction.

Definition 9. Let $s = (l, i, r; \lambda, j, \rho)$ be an obstruction of the list $G = (g_i)_{1 \leq i \leq q}$ of monic polynomials in $k\langle T \rangle$.

- If $l = 1$ then we call s a *right obstruction*.

- If $r = 1$ and $l \neq 1$ then s is called a *left obstruction*.
- The remaining obstructions with $\lambda = \rho = 1$ (so $s = (l, i, r; 1, j, 1)$) are called *central obstructions*.

Proposition 1.7. *Let G be a set of polynomials in $k\langle T \rangle$ and let H be the set of all non-zero normal forms of S -polynomials with respect to G corresponding to all left, right, and central obstructions of G . Then H is a basic set for G . If G is finite, then so is H .*

Proof. Follows directly from Lemma 1.6. \square

As H is a finite set if G is finite, Condition (iv) of Theorem 1.5 can be used to verify if G is a Gröbner basis.

Corollary 1.8. *Let G and H be as in Proposition 1.7. Then G is a Gröbner basis if and only if the normal form of each element of H with respect to G is zero.*

Proof. Immediate from Conditions (ii) and (iv) of Theorem 1.5. \square

The set of basic obstructions can be trimmed further.

Lemma 1.9. *Let $(l_1, i, 1; 1, j, \rho_1)$ and $(\omega l_1, i, 1; 1, q, \rho_2)$ be two obstructions with respective S -polynomials s_1 and s_2 . If $\omega \neq 1$, then there exists an S -polynomial $s_3 = s(\omega, j, r_3; 1, q, \rho_3)$ with $r_3 = 1$ or $\rho_3 = 1$, such that s_2 is reducible from the union of the S -polynomials $\{s_1, s_3\}$ and S -polynomials s_p with $L(s_p) < L(s_2)$.*

Proof. We know that $l_1 L(g_i) = L(g_j) \rho_1$ and $\omega l_1 L(g_i) = L(g_q) \rho_2$. Substitution gives $\omega L(g_j) \rho_1 = L(g_q) \rho_2$. This means that there exist obstructions of the form $(\omega, j, \rho_1; 1, q, \rho_2)$. By Lemma 1.6 we can find one obstruction o_3 by reducing ρ_1 and ρ_2 to 1 and some ρ_3 . Here $\rho_{q_1} = \rho_3 \rho_{q_2}$, resulting in a left obstruction if $q_1 = 2$ and $q_2 = 1$ and in a central obstruction if $q_1 = 1$ and $q_2 = 2$.

Now we have two polynomials s_1 and s_3 . Suppose that they are weak. We have to consider two cases. First assume $\rho_2 = \rho_3 \rho_1$ and $s_3 = s(\omega, j, 1; 1, q, \rho_3)$. Now

$$s_2 = \omega l_1 g_i - g_q \rho_3 \rho_1 = \omega(l_1 g_i - g_j \rho_1) + (\omega g_j - g_q \rho_3) \rho_1 = \omega s_1 + s_3 \rho_1$$

The monomials on the right hand side are less than or equal to $L(s_2)$ except possibly when the leading monomials of the weak representations of ωs_1 and $s_3 \rho_3$ cancel each other. In the exceptional case we can write $s_1 = \sum_h c_h l_h g_{\nu(h)} r_h$ with $l_h L(g_{\nu(h)}) r_h \leq L(s_1)$ and $s_3 = \sum_q c_q l_q g_{\mu(q)} r_q$ with $l_q L(g_{\mu(q)}) r_q \leq L(s_3)$. Then there are indices h_0 and q_0 with $l_{h_0} L(g_{\nu(h_0)}) r_{h_0} = L(s_1)$ and $l_{q_0} L(g_{\mu(q_0)}) r_{q_0} = L(s_3)$. This leads to a set of central obstructions $o_p = (l_{h_p}, h_p, r_{h_p}; l_{f_p}, f_p, r_{f_p})$. If all of these obstructions are weak then we can rewrite the right hand side of the equation for s_2 and satisfy the condition that all monomials on the right are less than or equal to $L(s_2)$. Thus s_2 is reducible from a set of S -polynomials containing s_1, s_3 and all S -polynomials corresponding to the o_p and the latter have a leading monomial less than $L(s_2)$.

Finally assume $\rho_1 = \rho_3 \rho_2$ and $s_3 = s(\omega, j, \rho_3; 1, q, 1)$. Now

$$s_2 = \omega l_1 g_i - g_q \rho_2 = \omega(l_1 g_i - g_j \rho_1) + (\omega g_j \rho_3 - g_q) \rho_2 = \omega s_1 + s_3 \rho_2$$

and we can finish by a similar argument. \square

1.3. The algorithm. For our non-commutative Gröbner basis computation we use two sets of polynomials with properties specified in the following definition. Observe that a normal form of a polynomial f with respect to a finite subset of $k\langle T \rangle$ can be computed in much the same way as for the commutative case, so that we may assume the presence of a (terminating) normal form algorithm.

Definition 10. Let I be a two sided ideal of $k\langle T \rangle$ and let G, D be finite subsets of $k\langle T \rangle$. We say that (G, D) is a *partial Gröbner pair* for I if the following properties are satisfied.

- (i) All polynomials in $G \cup D$ are monic.
- (ii) $G \cup D$ is a generating set for I .
- (iii) Every element of D is reduced with respect to G .
- (iv) The set D is basic for G .

Corollary 1.10. *Let I be a two-sided ideal of $k\langle T \rangle$ and let (G, D) be a partial Gröbner pair for I . If D is the empty set, then G is a Gröbner Basis for I .*

Proof. Direct from Definition 10 and Theorem 1.5(iv). □

The Gröbner basis algorithm will start with a finite set G forming a basis of I and with D the basic set determined by Proposition 1.7. The last condition is taken care of by 'cleaning operations'. Observe that (G, D) is indeed a partial Gröbner pair.

The main ingredient of the algorithm is an iteration step that changes the pair (G, D) to another partial Gröbner pair (G', D') such that the ideal generated by the leading monomials of G' strictly contains the ideal generated by the leading monomials of G . As the non-commutative polynomial ring $k\langle T \rangle$ is not Noetherian, we cannot expect the algorithm to terminate in all cases. Termination happens if $D = \emptyset$, in which case G will be a Gröbner basis of I , by Corollary 1.10. In the iteration step one element from D is moved to G and care is taken to let the new pair become partial Gröbner again. In the next theorem we discuss the algorithm.

Theorem 1.11. *Let I be an ideal of $k\langle T \rangle$ and let (G, D) be a partial Gröbner pair for I . The next routine of four steps computes a new partial Gröbner pair (G', D') for I with the property that the leading monomials of G generate an ideal strictly contained in the ideal generated by the leading monomials of G' . If $D' = \emptyset$, the routine leads to a halt and G' is a Gröbner basis for I .*

- (1) Write $G = \{g_1, \dots, g_{N-1}\}$. Move one polynomial f from D to G . Now $G = \{g_1, \dots, g_{N-1}, g_N = f\}$.
- (2) Compute the left, right, and central obstructions of G that involve N (these are of the form $(l, i, r; \lambda, N, \rho)$ or $(l, N, r; \lambda, j, \rho)$ for certain $i, j \in \{1, \dots, N-1\}$ and $l, r, \lambda, \rho \in T$). Add to D the non-zero normal forms of their S-polynomials with respect to G , so that D becomes a basic set for the new G .
- (3) For each $i \in \{1, \dots, N-1\}$ compute the normal form g'_i with respect to $G \setminus \{g_i\}$ of g_i . If $g'_i = 0$ remove g_i from G . Otherwise, if g'_i is distinct from g_i ,
 - (a) replace g_i by the monic version of g'_i ;
 - (b) compute the left, right, and central obstructions of the new G involving g'_i ;

- (c) if the normal form with respect to G of the S-polynomial of such an obstruction is non-zero then add to D the monic version of a normal form of it.
- (4) Replace each $d \in D$ by the monic version of its normal form with respect to G . If this normal form is zero, remove it from D .

Proof. We have to show that the four conditions of Definition 10 hold for the new pair (G, D) obtained at the end of the routine of the theorem.

- (i). Since all new polynomials in G and D are monic versions of non-zero polynomials, they are monic, so condition (i) of Definition 10 is satisfied.
- (ii). In (1), the set $G \cup D$ remains unchanged. In (2) polynomials from I are added to D so $G \cup D$ still generates I . In (3) and (4), elements are replaced by their normal forms with respect to subsets of G , so the generation of I by $G \cup D$ remains intact.
- (iii). Step (4) takes care of these normal form requirements on D .
- (iv). The changes of G , at (1) and (3)(a), are followed by an update of the obstructions, at (2) and at (3)(b) and subsequent additions to D of normal forms of S-polynomials, at (2) and (3)(c), to take care that (iv) is satisfied by Proposition 1.7.

The final assertion is a consequence of Corollary 1.10. □

The normal form of $g_i \in G$ with respect to $G \setminus \{g_i\}$ can change as a result of a replacement of some g_j by an element with smaller leading monomial. To avoid entering a recursive procedure of unknown depth, we observe that if the normal form of g_i alters by changing an element g_j , this happens via the construction of a central obstruction between g_i and g_j . The normal form of the S-polynomial corresponding to this obstruction is in fact the new form for g_i . By adding this S-polynomial to D we satisfy the definitions of G and D and make sure that the normal form of g_i will eventually become a member of G .

The theorem results in the procedure in Table 1, called *SGrobnerLoop*, that gives an impression of core of the Gröbner basis computation in the GAP package GBNP [1].

We briefly discuss the functions that are not standard GAP functions. The function *MkMonicNP* makes a non-zero non-commutative polynomial monic by a suitable scalar multiplication, *LTerms* (which still reflects the fact that some people refer to members of T as terms instead of monomials) lists the leading monomials of a list of polynomials, and *NormalForm* is as described below Definition 3.

The function *Occur(m,t)* finds an index i such that t is a subword of m beginning at the i -th symbol; if there is no such occurrence, the function returns 0. In our implementation a more efficient technique, viz. *tries*, cf. [3, Section 6.3], is used to check multiple cases in one treatment instead of these individual invocations of *Occur*.

The function *Obs(N,G,D)* occurs twice in *SGrobnerLoop*. It is actually a procedure in the sense that it alters its argument D . First it computes obstructions of the form $(l, N, r; \lambda, j, \rho)$. It uses the Lemmas 1.1, 1.3, 1.4, 1.6, and 1.9 to minimize the number of obstructions as much as possible. Instead of returning obstructions the function computes the corresponding S-polynomials and adds their normal forms with respect to $G \cup D$ to D .

```

SGrobnerLoop := function(G, D) local i, lG, Ls, s, l, h;
lG := Length(G);
while D ≠ ∅ do s := D[1]; Ls := s[1][1];
  Add(G, s);
  Obs(lG + 1, G, D);
  i := 1; l := lG;
  while i < l do h := G[i];
    if Occur(Ls, h[1][1]) > 0 then
      RemoveElmList(G, i); s := NormalForm(h, G ∪ D);
      if s = 0 then lG := lG - 1;
      else Add(G, MkMonicNP(s));
        Central(lG, G, D); Obs(lG, G, D);
      fi; l := l - 1;
    else i := i + 1;
    fi;
  od; l := Length(D); i := 2;
  while i ≤ l do h := D[i];
    if Occur(Ls, h[1][1]) > 0 then
      RemoveElmList(D, i); s := NormalForm(h, G ∪ D);
      if s ≠ 0 then
        InsertElmList(D, i, MkMonicNP(s)); i := i + 1;
      else l := l - 1;
      fi;
    else i := i + 1;
    fi;
  od;
  RemoveElmList(D, 1); s := LTerms(D); SortParallel(s, D, Lt);
od;
end;

```

TABLE 1. The procedure *SGrobnerLoop*

The last function that is not a standard GAP function is $Central(N, G, D)$. Usually central obstructions are subsumed by normal forms of the polynomials involved in G . In the part of the loop where this function is placed however we stay away from replacing a polynomial by a normal form of it with respect to the rest of G ; see the discussion before *SGrobnerloop*. This is to avoid a recursive procedure in which for these new normal forms again obstructions have to be searched. Instead, when we find a central obstruction, we add the normal form of its S-polynomial with respect to $G \cup D$ to D . This means that we actually place the normal form of the polynomial as a new polynomial in D . This guarantees that eventually we

will deal with this new polynomial and at that point the polynomial in G will be substituted by its normal form.

The GAP function $SortParallel(s,D,Lt)$, where $s = LTerms(D)$, sees to it that D is ordered according to increasing leading monomials.

2. MODULE CONSTRUCTION IN GBNP

This section describes the background of work carried out by Jan Willem Knopfer during his stay at the University of St Andrews with Steve Linton.

2.1. Introduction. Steve Linton has written two articles about vector enumeration. The first is called “Constructing Matrix Representations of Finitely Presented Groups” [4] and the second is called “On vector enumeration” [5]. Inspired by these papers, we outline a construction of modules using Gröbner basis methods, which is implemented in GBNP. Let k be a field and A be the algebra $k\langle X \rangle$, where $X = \{x_1, \dots, x_n\}$. Let G_{ts} be a set of generators of an ideal I (ideals are two-sided unless explicitly mentioned otherwise) of A . Put $\bar{A} = A/I$. Let A^s be the free right module of rank s over A and let G_p be a finite subset of A^s . The elements of G_p are called *module relations*. Here the suffix p refers to prefix rules for the module and ts to two-sided ideal rules. Denote by $\overline{G_p A}$ the image in \bar{A}^s of the submodule of A^s spanned by G_p . We describe a way to compute the quotient module $\bar{A}^s / \overline{G_p A}$, if possible at all, by use of the Gröbner basis method.

In order to embed the data into a single ring of noncommutative polynomials, we view A^s as the free A -module freely generated by $M = \{m_1, \dots, m_s\}$ and introduce a new indeterminate e (for the identity) and view m_1, \dots, m_s as indeterminates over $k\langle X \rangle$. So we will be working in the free algebra $k\langle X \cup M \cup \{e\} \rangle$. Let $W_e = \{x \cdot e - x \mid x \in X \cup M \cup \{e\}\} \cup \{e \cdot t - t \mid t \in X\}$. Let $W_M = \{x \cdot m_i \mid x \in X \cup M \cup \{e\}, m_i \in M\}$. Note that W_e contains the relation $e^2 - e$, so e is an *idempotent*.

Theorem 2.1. $k\langle X \cup M \cup \{e\} \rangle / (W_e \cup W_M) \cong (A + ek) \oplus m_1 A \oplus \dots \oplus m_s A$ as A -algebras where $A + ek$ is the A -module spanned by a free submodule A and an additional generator e satisfying $e \cdot 1 = e$ and $e \cdot t = t$ for $t \in X$.

Proof. Let x be a monomial in $k\langle X \cup M \cup \{e\} \rangle$. If $x = 1$ or $x = e$, then $x \in A + ek$. Suppose $x \neq 1, e$. If x contains a factor m_i but does not begin with it, it can be reduced to 0 modulo W_M . If x contains e as a factor, but not the first, it can be reduced by a rule in W_e so that it starts with e ; in particular it does not start with an m_i and so we may assume that it does not contain a factor m_i . As $x \neq e$, there is another indeterminate $y \in X \cup \{e\} \setminus M$ such that x begins with ey , say $x = eyz$. But then it can be reduced to yz . Continuing this way, we may suppose that x does not contain e . Thus, after reduction, each monomial x falls into one of two categories:

- x does not contain any $m \in M$. Then $x \in A + ek$.
- x contains a single $m \in M$, contains no factor e , and begins with m , in other words $x \in mA$.

In both cases, $x \in (A + ek) \oplus m_1 A \oplus \dots \oplus m_s A$. It is easily seen that no further collapse of monomials occurs modulo $(W_e \cup W_M)$, and so $k\langle X \cup M \cup \{e\} \rangle / (W_e \cup W_M) \cong (A + ek) \oplus m_1 A \oplus \dots \oplus m_s A$ as k -modules.

Right multiplication by elements of A on the quotient algebra of $k\langle X \cup M \cup \{e\} \rangle$ by $(W_e \cup W_M)$ coincides with the right multiplication specified for $(A + ke) \oplus m_1 A \oplus \cdots \oplus m_s A$. Hence the isomorphism is an isomorphism of A -modules. \square

Corollary 2.2. *If F is a Gröbner basis in $k\langle X \cup M \cup \{e\} \rangle$ of $W \cup G \cup W_e \cup W_M$, where $G \subseteq A$ and $W \subseteq A^s$, then $F \cap k\langle X \rangle$ is a Gröbner basis for (G) in $k\langle X \rangle$. Moreover, for $x \in m_1 A \oplus \cdots \oplus m_s A$ and $a \in A$, the normal form of xa with respect to F belongs to $m_1 A \oplus \cdots \oplus m_s A$ and gives a unique representative for the image of xa in $(m_1 \bar{A} \oplus \cdots \oplus m_s \bar{A}) / (\bar{A}^s \cap \bar{F})$.*

In our implementation we do not add W_M and W_e explicitly and work with W as a set of module relations, kept separate from the set G of ideal relations.

Since the whole Gröbner basis for the relations in G is needed, it might as well be calculated at the start; doing so will usually make the algorithm faster.

3. THE DIMENSION OF QUOTIENT ALGEBRAS

This section was originally contributed by Chris Krook and describes his work carried out during his stay at Lund University with Victor Ufnarovski.

Once we have found a Gröbner basis G for an ideal I in $k\langle T \rangle$, we can consider the quotient algebra $Q = k\langle T \rangle / (G)$. This quotient algebra need not be finite dimensional. In fact, there are many cases where the quotient algebra is infinite dimensional and has polynomial or even exponential growth. We present some functions that investigate the dimension of the quotient algebra.

For the study of the dimension, instead of looking at Q , we only need consider the monomial algebra $Q' := k\langle T \rangle / (G')$ where $G' := \{L(b) \mid b \in G\}$. Here, a *monomial algebra* is a quotient algebra of $k\langle T \rangle$ by an ideal generated by monomials. Although in general Q' will be a different algebra from Q , their ideals have the same set of *standard monomials*, that is, monomials not divisible by any leading monomial of a non-zero element the ideal. The images of these standard monomials are a basis of both quotient algebras (regardless of the reduction order chosen in the selection of the leading monomials among G and G'), so both quotient algebras have essentially the same basis of standard monomials. The dimension of the quotient algebra and other concepts of interest such as the Hilbert series are equal for the algebras Q and Q' since they depend only on the standard monomials. For this reason we will restrict ourselves to the study of monomial algebras in the remainder of this chapter.

3.1. The algorithm FinCheckQA. The first goal is to determine whether the quotient algebra is of finite or of infinite dimension. A monomial algebra is determined by an alphabet X and a set of monomials M , which is *reduced* in the sense that none of the monomials in M divides another monomial in M . Both X and M are assumed to be finite. Furthermore, each $m \in M$ is a word of finite length in X^* , the set of all *words* over the alphabet X of finite or infinite length. Recall that we call a word u *standard* (with respect to M) if no monomial in M divides u ; notation $M \nmid u$. Observe that if v is a standard monomial, the so is each of its divisors.

The standard words with respect to M are a convenient basis of the monomial algebra Q ; of course we mean here their images under the canonical projection $A \rightarrow Q$, but we allow ourselves to be imprecise in this respect. This basis is infinite if and only if it is possible to construct a standard word $w \in X^*$ of infinite length.

It is obvious that this condition is sufficient, since the word w will contain infinitely many different standard subwords. The other direction follows directly from the assumption that our alphabet is finite. Up to a given length, only finitely many words exist. This provides us with a direct and intuitive approach to determine finiteness given X and M . We simply try to construct an infinite standard word. If we succeed, we can conclude that the dimension of the monomial algebra is infinite. Otherwise we conclude that the dimension is finite.

In order to make this more precise, we introduce a graph on standard monomials and use this graph to obtain some information on the structure of infinite words.

Definition 11. Given an alphabet X and a set of monomials M , we define the *Ufnarowski Graph*¹ (see [7]) Γ_M . Its vertex set V consists of all standard words $w \in X^{l_M}$, where $l_M := -1 + \max\{|m| \mid m \in M\}$, where $|m|$ is the length of m . For each $v, w \in V$ there is a directed edge $v \rightarrow w$ if and only if there exist $a, b \in X$ such that $va = bw$ and $M \nmid va$.

As one readily checks there is a one-to-one correspondence between paths of length l in Γ_M and standard words of length $l + l_M$. This implies that each infinite standard word corresponds to an infinite path in Γ_M . Since the graph has a finite number of vertices, due to the finiteness of X and M , this implies that such an infinite path must contain a cycle. But then the word corresponding to merely repeating this cycle is an infinite standard word as well. Our conclusion is as follows.

Remark 3.1. If there exists an infinite word that is standard with respect to M , then either it is cyclic or it gives rise to a cyclic infinite word that is also standard with respect to M .

We will use this remark in the proof of the following lemma, in which the reduction ordering $<$ is used.

Lemma 3.2. *If there exists an infinite word $w' \in X^*$ that is standard with respect to M , then there also exists a cyclic infinite word $w \in X^*$ that is standard with respect to M such that*

$$(1) \quad \forall_{r,s \geq 1} w[1..s] \leq w[r..r+s-1].$$

Here $w[p..q]$ stands for the subword of w obtained by removing the first until the $(p-1)$ st and the $(q+1)$ st up to the last. Before giving the proof we introduce the notation $u \trianglelefteq v$ to denote that u is a prefix of v and the notation $u \triangleleft v$ to denote that u is a proper prefix of v . Furthermore u^t denotes the concatenation of t copies of a word u .

Proof. Let $w' \in X^*$ be infinite and standard with respect to M . Then, according to Remark 3.1, w' gives rise to a cyclic infinite word $w'' = v'^\infty$, where $v' \in X^p$ for some finite $p > 0$. We assume that v is the lexicographically smallest cyclic shift of v' . Then there is a $u \triangleleft v'$ such that $v'^\infty = uv^\infty$. Now define $w := v^\infty$ and the lemma follows immediately. \square

¹This graph should not be confused with *the graph of standard words*.

- (1) Build the tree \mathcal{T} of reversed monomials from M . It has root 1, leaves the reversed members of M and nodes all prefixes, with v descending from u whenever $|v| = |u| + 1$ and u is a prefix of v . See Example 3.14 for a drawing of \mathcal{T} in case of an example.
- (2) Start with the smallest word $w := x_1$. Set a pointer p at the first position, thus $p := 1$.
- (3) Check whether $M|w$ by use of \mathcal{T} (\star):
 - if w is standard, then we:
 - (a) check if we can conclude the dimension is infinite (\boxtimes) in which case we **terminate** instantly;
 - (b) extend word: $w := w + w[p]$ and $p := p + 1$.
 - if w is not standard, then we:
 - (a) check if $w[1] = x_k$ in which case we can conclude the dimension is finite and **terminate** instantly;
 - (b) increase w in a minimal way: $w := u$ where u is the smallest word satisfying $w < u$ and $|u| \leq |w|$. Furthermore the pointer is reset to $p := 1$.
- (4) Repeat Step (3) until **termination** is achieved.

TABLE 2. Construct an infinite word

Remark 3.3. For a given word u that is lexicographically smaller than all its cyclic shifts, the word $w = u^q u'$ with $u' \triangleleft u$ and $q \geq 1$, satisfies condition (1). This is an immediate consequence of the property of u .

In order to find infinite words, it suffices to use only words satisfying (1). This implies that in its construction, given a word $w \in X^l$ satisfying (1) we have the following two ways to proceed;

- if $M|w$ then $w := w'$ where w' is the smallest word satisfying $w' > w$ and $|w'| \leq |w|$. It is obvious that w' will also satisfy condition (1) since the symbol increased will be the last position of w' . Example with $X = \{x, y\}$ and $xyxy \in M$: if $w = xyxy$ then $w' = xyxy$.
- if $M \nmid w$ then we want to extend the potential beginning of an infinite word, such that (1) remains satisfied. For this purpose we can use a pointer that moves along the word while lengthening it and is reset to the first position if the word is increased. The pointer points to the next symbol to be added. By doing so, w will always have the form mentioned in Remark 3.3, thus satisfying condition (1). In the following example we denote the pointer position using bold letters. Example with $X = \{x, y\}$ and $M = \{xxx\}$:
 $\dots \rightarrow \mathbf{x}xy \rightarrow \mathbf{x}xyx \rightarrow \mathbf{x}xyxx \rightarrow \mathbf{x}xy\mathbf{x}xy \rightarrow \dots$

This leads to the following algorithm for constructing infinite words on the alphabet $X = \{x_1, \dots, x_k\}$.

ad \star : While constructing our word w we already know that the longest proper prefix w' of w is standard, thus we only have to check whether each suffix v of w is standard. By use of a tree structure \mathcal{T} to store the reversed monomials from M this comes down to checking whether there is no branch b in \mathcal{T} such that $b \leq \text{rev}(v)$. This will always take at most $|v|$ comparisons.

ad ✘: We still need to work out how to conclude infinite dimensionality from a word w such that $M \nmid w$. For this purpose, use the Ufnarowski graph Γ_M from Definition 11 and in particular the one-to-one correspondence between infinite paths in Γ_M (i.e. cycles) and infinite words. This tells us how to adapt our algorithm to detect that the dimension is infinite. While constructing our word w , we simultaneously build up that part of Γ_M that is on our route. Now we can easily check our word for cycles, by just checking whether the vertex we want to add to our graph is already in the vertex set.

Algorithm 2 automatically keeps track of the route in Γ_M since all the words on this route can simply be read from the word w itself.

Example 3.4. $X = \{x, y\}, M = \{xx, yyy\}$

$$\begin{array}{llll}
 w = \mathbf{x} & V = \{\} & & \\
 w = x\mathbf{x} & V = \{\} & M|w & \\
 w = \mathbf{x}y & V = \{xy\} & & \\
 w = x\mathbf{y}x & V = \{xy, yx\} & & \\
 w = xy\mathbf{x}y & V = \{xy, yx\} & DONE &
 \end{array}$$

In the final step, we find a cycle and conclude that the dimension is infinite. The bold letters indicate the position of the pointer.

Lemma 3.5. *Algorithm 2 terminates and concludes whether the monomial algebra determined by an alphabet X and a set of monomials M , ordered lexicographically, is finite or infinite.*

Proof. We need to show termination of the algorithm. In each step the word w is increased lexicographically. Since Γ_M consists of at most k^{l_M} vertices, a word length greater than $k^{l_M} + l_M$, which corresponds to a path of length greater than k^{l_M} in Γ_M implies a cycle and thus infinite dimensionality. Thus, if an infinite word exists, one will be found after a finite number of steps.

Now assume that all words are finite. There are only finitely many words with length $\leq k^{l_M} + l_M$ thus in a finite number of steps $w[1] = x_k$ will hold. Notice that from this point on, increasing $w = (x_k)^q$ gives us $w = (x_k)^{q+1}$ and after finitely many steps $M|w$ will hold, which implies finite dimensionality since all attempts to create infinite words have failed. \square

We will consider two small examples, showing how the algorithm works in both the finite and the infinite case. All words considered by the algorithm are written down.

Example 3.6. (infinite) $X = \{x, y\}, M = \{xx, xyx, yyy\}$

$$\begin{array}{llll}
 w = \mathbf{x} & V = \{\} & & \\
 w = x\mathbf{x} & V = \{\} & M|w & \\
 w = \mathbf{x}y & V = \{xy\} & & \\
 w = x\mathbf{y}x & V = \{xy\} & M|w & \\
 w = \mathbf{x}yy & V = \{xy, yy\} & & \\
 w = x\mathbf{y}yx & V = \{xy, yy, yx\} & & \\
 w = xy\mathbf{y}xy & V = \{xy, yy, yx\} & DONE &
 \end{array}$$

We conclude infinite dimensionality as we encounter a cycle. The infinite standard word is $w = (xyy)^\infty$. Notice that increasing the word xx in the third step ensures that, from that point on, no words containing xx will be considered. Therefore in

the final step the word $xyyxy$ is checked instead of first checking $xyyxx$. One can easily think of examples where this principle reduces the number of words to be considered much more.

Example 3.7. (finite) $X = \{x, y\}$, $M = \{xx, yxy, yyy\}$

$w = \mathbf{x}$	$V = \{\}$	
$w = x\mathbf{x}$	$V = \{\}$	$M w$
$w = \mathbf{xy}$	$V = \{xy\}$	
$w = xy\mathbf{x}$	$V = \{xy, yx\}$	
$w = xy\mathbf{xy}$	$V = \{xy, yx\}$	$M w$
$w = \mathbf{xyy}$	$V = \{xy, yy\}$	
$w = xy\mathbf{yx}$	$V = \{xy, yy, yx\}$	
$w = xy\mathbf{yxy}$	$V = \{xy, yy, yx\}$	$M w$
$w = \mathbf{xyyy}$	$V = \{xy, yy\}$	$M w$
$w = \mathbf{y}$	$V = \{\}$	
$w = y\mathbf{y}$	$V = \{yy\}$	
$w = yy\mathbf{y}$	$V = \{yy\}$	$M w, DONE$

We conclude finite dimensionality. In the finite case we always have to consider all possible words satisfying condition (1), having no proper prefix divisible by M .

We have seen that in the finite case, our algorithm has to check many words. One might wonder if there are also some bad scenarios in the infinite case. Unfortunately there are. We will describe them using the Ufnarovski Graph Γ_M .

Lemma 3.8. *Let $|X| = n$. Then for each $l \geq 1$ there exists a set M such that $l_M = l$ and Γ_M contains only a cycle of length n^l , thus visiting all the vertices in Γ_M .*

Proof. The cycle we are looking for is well known as the *De Bruijn-cycle*. Construct a graph G which has as vertex set all possible words of length $l - 1$. There is a directed edge from vertex u to vertex v if and only if there exist $a, b \in X$ such that $ua = bv$. Label this edge with b . Now notice that for each vertex v we have $\deg_{in}(v) = \deg_{out}(v) = n$. Therefore there must exist an Euler cycle in G and one readily sees that writing down the labels in this cycle gives us exactly the *De Bruijn-cycle*, a cycle of length $n \cdot n^{l-1}$ indeed.

Now given the cycle in the graph Γ_M , it is easy to define the set M such that M induces Γ_M , just by preventing all edges not on the cycle to exist. \square

As an unwelcome result of Lemma 3.8, there always exists a bad case in which the length of the shortest cycle and thus the amount of comparisons needed to conclude infinite dimensionality grows exponentially in l_M , roughly said the length of the largest monomial in M .

3.2. The algorithm FinCheckQA. The algorithm FinCheckQA determines the growth of the QA. Sometimes we want to know more about the dimension of our algebra than just whether it has a finite or infinite basis. The growth of the algebra is such an example. We will give a definition of the concept growth and we will present an algorithm for computing it.

Definition 12. Given a monotone function $f : \mathbb{N} \rightarrow \mathbb{R}^+$ we define the growth of f as the equivalence class $[f]$ of f where

$$[f] = \{g : \mathbb{N} \rightarrow \mathbb{R}^+ \mid \exists_{c_1, c_2, m_1, m_2 \in \mathbb{N}} \forall_{z \in \mathbb{N}} c_1^{-1} g(zm_1^{-1}) \leq f(z) \leq c_2 g(m_2 z)\}$$

The usual inequality for functions carries over to the equivalence classes. Furthermore we can distinguish the following cases.

- If $[f] \leq [1]$ then we say that f has *finite growth*,
- If $[f] \leq [z \mapsto z^d]$ for some $d > 0$ and d is a minimal natural number satisfying this inequality, then we call f *polynomial* of degree d .
- If $[f] \geq [z \mapsto a^z]$ for $a > 1$, then we call f *exponential*.

3.3. Growth of an algebra. In Section 3.1 we studied whether the dimension of a monomial algebra determined by an alphabet X and a set of monomials M is finite or not, i.e., whether the number of standard words of finite length was finite. More generally, growth will be measured by counting the number of words up to each given length. More formally, we consider our algebra Q which is graded as a vector space by $Q = \bigoplus_i Q_i$, where Q_i is the subspace of Q spanned by all standard words of length i . Now define the *growth of the algebra* Q as the function $f : \mathbb{N} \rightarrow \mathbb{N}$ by $f(n) = \dim(\bigoplus_{i=1}^n Q_i)$.

3.4. Growth of a graph. For computations we will translate the problem of computing the growth of the algebra, into the problem of computing the growth of the graph Γ_M . So we will also define a growth function on graphs. Let G be a finite graph. The intended function counts the number of different paths in G up to each given length n . Thus we define $f : \mathbb{N} \rightarrow \mathbb{N}$ by $f(z) = \#\text{different paths in } G \text{ of length up to } z$.

Note that our algebra corresponds to the graph Γ_M in such a way that there exists a bijection between words of length n in the algebra and paths of length n in the graph. Hence these notions of growth are equivalent.

One readily sees that if a graph contains a cycle, there exist paths of length n for all $n \in \mathbb{N}$, thus the graph must be at least polynomial. We can intuitively see a relation between the growth of a graph and the number of cycles in it.

- If a finite graph contains no cycles, then the total number of different paths is finite and so the graph has finite growth.
- If a finite graph contains a path visiting d non-intersecting cycles, then it has at least polynomial growth of degree d . Intuitively, there are $k + d - 1$ road segments, k of which are cycles and $d - 1$ of which are connecting paths between the cycles. The number of paths that contain k cycles equals the number of ways to appoint k cycles in a set of $k + d - 1$ road segments which equals $\binom{k+d-1}{k}$ which is polynomial in k . Since the length of the cycles is upper bounded, this leads to polynomial growth in the length of the paths as well.
- If a graph contains 2 intersecting cycles, then the number of different paths of length up to n grows exponentially in n . Intuitively each time you reach a vertex that is contained in both cycles, you can go 2 ways. So there are already 2^k different paths that visit the vertex k times.

This topic is dealt with more thoroughly in [8], which also contains a proof of the above intuitive remark.

3.4.1. Constructing an infinite word. We will next present an algorithm that is based on Algorithm 2 and that can distinguish the three cases; finiteness, polynomial growth and exponential growth. In the case of polynomial growth, it does not always compute the exact degree, but it will give an upper and a lower bound.

- (1) Build a tree \mathcal{T} of reversed words from M .
- (2) Start with the smallest word $w := x_1$. Set a pointer p at the first position, thus $p := 1$. Furthermore let $cycles := \{\}$.
- (3) Check whether $M|w$ (\star).
 - if w is standard, then
 - (a) if w contains a cycle of subwords of length l_M
 - that intersects with a cycle in the list $cycles$, then we conclude exponential growth and **terminate**.
 - that does not intersect with any cycle in the list $cycles$, then we add a pair of indices $\{i_1, i_2\}$ to the list $cycles$ such that $w[i_1..i_2 + l_M]$ is the cycle.
 - (b) extend the word: $w := w \cdot w[p]$ and $p := p + 1$.
 - if w is not standard, then
 - (a) check if $w[1] = x_k$ in which case we have checked all words. We can conclude polynomial growth of degree $d_{low} \leq d \leq d_{upp}$ (\star) and will **terminate**. The upper bound d_{upp} is equal to the total number of cycles encountered in the entire check. The lower bound d_{low} is equal to the maximum number of disjoint cycles encountered within one word during the entire check. $d = 0$ corresponds to finiteness.
 - (b)
 - Increase w in a minimal way: $w := u$ where u is the smallest word satisfying $w < u$ and $|u| \leq |w|$.
 - The pointer is reset to $p := 1$.
 - Furthermore the list $cycles$ is updated as to only consider beginnings of cycles in the new word u . Thus for each pair $\{i_1, i_2\}$ in $cycles$, it is deleted if $|u| \leq i_1$ or it is replaced by $\{i_1, |u| - l_M + 1\}$ if $i_1 < |u| \leq i_2 + l_M$.
- (4) Repeat Step (3) until **termination** is achieved.

TABLE 3. Construct an infinite word and look for intersecting cycles

As we mentioned earlier, our algorithm only gives an upper bound and a lower bound on the degree of polynomial growth. We will give two examples to clarify the algorithm and to show the complications that can occur in the computations, which cause the use of bounds in stead of an accurate value.

Example 3.9. We take $X = \{x, y\}$, $M = \{yxx, yy\}$, and use the Ufnarovski graph Γ_M .

$$\widehat{x} \rightarrow xy \rightleftharpoons xy$$

In this case no problems occur and polynomial growth of degree 2 is detected. Algorithm 3 first notices that xxx contains a cycle and then tries to expand xx further. In this process the word $xyxy$ is encountered, which also contains a cycle. Since $|cycles| = 2$ for this word, which implies that there is a word containing 2 cycles, we have $d_{low} = 2$. On the other hand no other cycles are encountered, so

$d = d_{low} = d_{upp} = 2$. More precisely algorithm 3 follows the following steps.

$w = \mathbf{x}$	$V = \{\}$	$cycles = \{\}$	
$w = \mathbf{xx}$	$V = \{xx\}$	$cycles = \{\}$	
$w = \mathbf{xxx}$	$V = \{xx\}$	$cycles = \{\{1, 2\}\}$	
$w = \mathbf{xy}$	$V = \{xx, xy\}$	$cycles = \{\{1, 1\}\}$	
$w = \mathbf{xyx}$	$V = \{xx, xy, yx\}$	$cycles = \{\{1, 1\}\}$	
$w = \mathbf{xyxx}$	$V = \{xx, xy, yx\}$	$cycles = \{\{1, 1\}\}$	$M w$
$w = \mathbf{xyxy}$	$V = \{xx, xy, yx\}$	$cycles = \{\{1, 1\}, \{2, 4\}\}$	
$w = \mathbf{xyyy}$	$V = \{xx, xy\}$	$cycles = \{\{1, 1\}, \{2, 2\}\}$	$M w$
$w = \mathbf{y}$	$V = \{xy\}$	$cycles = \{\}$	
$w = \mathbf{y}$	$V = \{\}$	$cycles = \{\}$	
$w = \mathbf{yy}$	$V = \{yy\}$	$cycles = \{\}$	$M w, DONE$

The list *cycles* holds all pairs of indices $\{i_1, i_2\}$ such that $w[i_1..i_2]$ is the beginning of a cycle. Thus i_2 is sometimes lowered if w is increased. Notice that if a cycle is encountered that begins with a word w of length l_M and this cycle is fully examined, i.e. increasing of the word means removing the cycle starting with w from the list *cycles*, we can add w to M . We do not miss out on exponential growth, for this would have been detected in examining this cycle. In this way we may loose more information on the actual degree of growth though.

Example 3.10. Take $X = \{x, y\}$, $M = \{xy, yy\}$. Then the degree of the polynomial growth is not detected. The Ufnarowski graph Γ_M is as follows.

$$xy \leftrightarrow yx \rightarrow \widehat{xx}$$

Algorithm 3 follows the following steps.

$w = \mathbf{x}$	$V = \{\}$	$cycles = \{\}$	
$w = \mathbf{xx}$	$V = \{xx\}$	$cycles = \{\}$	
$w = \mathbf{xxx}$	$V = \{xx\}$	$cycles = \{\{1, 2\}\}$	
$w = \mathbf{xy}$	$V = \{xx\}$	$cycles = \{\{1, 1\}\}$	$M w$
$w = \mathbf{xy}$	$V = \{xy\}$	$cycles = \{\}$	
$w = \mathbf{xyx}$	$V = \{xy, yx\}$	$cycles = \{\}$	
$w = \mathbf{xyxy}$	$V = \{xy, yx\}$	$cycles = \{\{1, 3\}\}$	
$w = \mathbf{xyyy}$	$V = \{xy, yy\}$	$cycles = \{\{1, 1\}\}$	$M w$
$w = \mathbf{y}$	$V = \{\}$	$cycles = \{\}$	
$w = \mathbf{yy}$	$V = \{yy\}$	$cycles = \{\}$	$M w, DONE$

Observe that $d_{low} = 1$ since the maximum number of disjoint cycles encountered at any stage is 1. Furthermore $d_{upp} = 2$ since in total 2 cycles are encountered. However, $d = 2$. The reason that this is not detected by the algorithm is that it will not go back from a cycle to a lexicographical smaller cycle. Remember that at all times we increase words in such a way that they satisfy equality (1) in section 3.1.

3.5. Computing Hilbert Series. The GBNP package also offers a function to compute partial Hilbert Series of a monomial algebra. This Hilbert series can be used as a measurement of growth and can be used to bound Gröbner basis computations.

Given a graded algebra $Q = \bigoplus_{i=0}^{\infty} Q_i$ all of whose subspaces Q_i are finite dimensional, define the Hilbert series by

$$H_Q := \sum_{i=0}^{\infty} (\dim Q_i) t^i.$$

Note that the monomial algebra we consider can always be split up into homogeneous subspaces, namely by taking Q_i to be the subspace spanned by monomials of length i . It is immediately clear from the definition that all the coefficients of the Hilbert series will be non-negative. The function offered is an implementation of an algorithm described in [9] and it uses the concept of a graph of chains and some cohomology result.

3.6. The PreProcess algorithm. Sometimes the set of monomials M can be easily *reduced*, which can save us a lot of time later on. By reduction, we mean that we can simplify the set in such a way that it does not affect the growth of the related monomial algebra. It will however change the set of standard words, and thus generate a different algebra. We distinguish the following two possibilities of reduction. Let $X = \{x_1, \dots, x_n\}$ and $w \in X^*$.

- (1) if $wx_1, \dots, wx_k \in M$ then $M := (M \setminus \{wx_1, \dots, wx_k\}) \cup \{w\}$
- (2) if $x_1w, \dots, x_kw \in M$ then $M := (M \setminus \{x_1w, \dots, x_kw\}) \cup \{w\}$

Remark 3.11. We are only interested in the second type of reduction, which we call *left reduction*, since the first type of reduction is implemented in Algorithms 2 and 3 and can be achieved by mere bookkeeping, as follows from the following argument. Consider a word $w = uv$ where v is a standard word of length l_M . If no extension of v leads to an infinite word, then we should not consider words containing v anymore in the remainder of our search and we can achieve this by adding v to M . By doing so, in the rest of the check, each word $w' = u'v$ where $u' > u$ will be recognized as a dead end immediately, without needing further checks.

Example 3.12. It is not always directly obvious that left reduction is possible. To see this, take $X = \{x, y\}$ and $M = \{xx, yxy\}$. As xx belongs to M , we can temporarily add the monomials xxx , xyx , and yxx to M . Now both xyx and yxx are in M , so we replace these by xy . Finally, we replace xxx and yxx back by xx . Thus M is left reduced to $M = \{xx, xy\}$.

One readily checks that if we would also have been interested in right reduction, then by applying reduction more than once we could even reduce the set M to $M = \{x\}$, in which case it is clear that $yy\dots$ is an infinite standard word.

Thus in order to find all possibilities of left reduction, we sometimes need to add subwords to the right of existing member of (M) first. We will make this more precise.

Definition 13. Given an alphabet X of size k . We call a set of monomials M *left reduced* if, for each $m \in M$, we have $|\{t \in M \mid t[2..|t|] \triangleleft m[2..|m|]\}| < n$.

Equality would imply that left reduction is possible; suppose n_i ($i = 1, \dots, k$) are different monomials in M such that $n_i[2..|n_i|] \triangleleft m[2..|m|]$. Then since n_i ($i = 1, \dots, k$) belong to M and so do not divide one another, $n_1[1], n_2[1], \dots, n_k[1]$ are all different and form the k leaves of a full subtree.

We can choose to reduce our monomial set M while building the tree \mathcal{T} before using Algorithms 2 and 3. Reduction does not alter infinite paths, but only cuts down dead-ends. Therefore these algorithms, which all look for infinite paths, still work on our reduced set and return the same result.

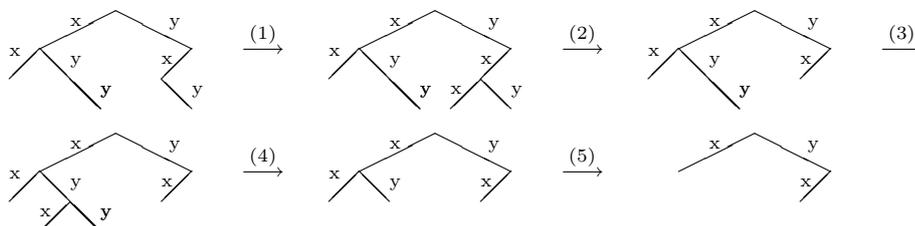
Since left reduction changes the set of standard words, the Hilbert series of this new algebra will differ from the original one, so we should not preprocess before computing the Hilbert series. Definition 13 hints us to a reduction algorithm.

Algorithm 3.13. Reduce a monomial set while building the corresponding tree

- (1) Initialize $\mathcal{T} = []$.
- (2) For all $u \in M$ consider all relevant expansions of u :
for all $v \in M$ with $|v| > |u|$ and $v[2..|u|] = u[2..|u|]$
 - Let w be the suffix of v of length $|v| - |u|$.
 - Add uw to \mathcal{T} .
 - Reduce \mathcal{T} such that it does not contain full subtrees.
- (3) Let M' be the set of words in M belonging to \mathcal{T} .
- (4) If $M \neq M'$ then go to Step (2) with $M := M'$.
- (5) **Return**(M).

The recursive character of the algorithm is due to the fact that new possibilities of left reduction may occur after a reduction step. This occurs in the following example.

Example 3.14. Take $X = \{x, y\}$ and $M = \{xx, yxy, yyx\}$. We will show the reduction process using the tree notation of reversed members of M .



- (1) the extension xyx of the word xy is added to the tree;
- (2) branches xyx and yxy can be reduced to branch xy ;
- (3) the extension yyx of the word xy is added to the tree;
- (4) branches xyx and yyx are reduced to branch yx ;
- (5) branches xx and yx are reduced to branch x .

Steps (1) and (2) occur in the first recursion step, Steps (3), (4), and (5) in the second recursion step.

Instead of full preprocessing we can also choose to upper bound the number of recursions. In this way we have some control over the amount of preprocessing.

REFERENCES

- [1] A.M. Cohen & D.A.H. Gijssbers, GBNP, A GAP Package for Gröbner bases of non-commutative polynomials.
<http://www.win.tue.nl/~amc/pub/grobner/doc.html>.

- [2] Edward L. Green, *Noncommutative Grobner bases, and projective resolutions*, pp. 29-60 in "Computational Methods for Representations of groups and algebras (Essen 1997)", Birkhauser, Basel 1999.
- [3] Donald E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, second edition, Addison Wesley Longman, 1998.
- [4] S. A. Linton, *Constructing matrix representations of finitely presented groups*, Computational group theory, Part 2, J. Symbolic Comput., **12** (1991)427-438.
- [5] S. A. Linton, *Computational linear algebra in algebraic and related problems (Essen, 1992)*, Linear Algebra Appl., **192** (1993)235-248.
- [6] T. Mora, *An introduction to commutative and non-commutative Gröbner Bases*, Journal of Theoretical Computer Science **134** (1994) 131-173.
- [7] P. Nordbeck, *Canonical Bases for Algebraic Computations*, Doctoral Thesis in Mathematical Sciences, LTH Lund (2001)
- [8] V.A. Ufnarovski, *A Growth Criterion for Graphs and Algebras Defined by Words*, Mathematical Notes 31, 238-241 (1982)
- [9] V.A. Ufnarovski, *Combinatorial and Asymptotic Methods in Algebra*, Algebra-VI, Encyclopedia of Mathematical Sciences, Volume 57, Springer (1995),5-196

ARJEH M. COHEN, DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE, EINDHOVEN UNIVERSITY OF TECHNOLOGY, POBox 513, 5600 MB EINDHOVEN, THE NETHERLANDS
E-mail address: A.M.Cohen@tue.nl