Short proofs of ideal membership

Clemens Hofstadler*
clemens.hofstadler@mathematik.uni-kassel.de
Institute of Mathematics, University of Kassel
Kassel, Germany

Thibaut Verron†
thibaut.verron@jku.at
Institute for Algebra, Johannes Kepler University
Linz, Austria

ABSTRACT

A cofactor representation of an ideal element, that is, a representation in terms of the generators, can be considered as a certificate for ideal membership. Such a representation is typically not unique, and some can be a lot more complicated than others. In this work, we consider the problem of computing sparsest cofactor representations, i.e., representations with a minimal number of terms, of a given element in a polynomial ideal. While we focus on the more general case of noncommutative polynomials, all results also apply to the commutative setting.

We show that the problem of computing cofactor representations with a bounded number of terms is decidable and NP-complete. Moreover, we provide a practical algorithm for computing sparse (not necessarily optimal) representations by translating the problem into a linear optimization problem and by exploiting properties of signature-based Gröbner basis algorithms. We show that for a certain class of ideals, representations computed by this method are actually optimal, and we present experimental data illustrating that it can lead to noticeably sparser cofactor representations.

CCS CONCEPTS

• **Computing methodologies → Algebraic algorithms**.

KEYWORDS

Noncommutative polynomials, signature Gröbner basis, automated proofs, proof simplification

ACM Reference Format:
Clemens Hofstadler and Thibaut Verron. 2023. Short proofs of ideal membership. In . ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

In polynomial algebra, the ideal membership problem is one of the most fundamental problems with many important applications from polynomial system solving over polynomial identity testing to automated reasoning. In case of noncommutative polynomials in the free algebra, this last applications is particularly relevant and has been focus of recent research [3, 12, 25, 26]. Several frameworks have been developed that allow to prove the correctness of statements about linear operators (such as matrices, homomorphisms, bounded operators, etc.) by verifying ideal membership of noncommutative polynomials. In this setting, any cofactor representation, that is, any representation of an ideal element as a linear combination of the generators, can be considered as a proof of the corresponding operator statement, and noncommutative Gröbner bases can be used to compute such representations [13].

In general, cofactor representations are not unique and different representations can differ drastically in their complexity. We could observe empirically that representations computed by Gröbner bases are often significantly longer than necessary. In this work, we discuss the problem of finding *sparsest* cofactor representations of an ideal element, that is, representations with a minimal number of terms. We focus on the situation of noncommutative polynomials, as we are particularly interested in computing short proofs of operator statements. However, all techniques also apply analogously to commutative polynomials.

Although ideal membership in the free algebra is only semidecidable, we show that the problem of computing cofactor representations with the number of terms bounded by $N \in \mathbb{N}$ is decidable, yet NP-complete. This yields a first (impractical) algorithm for computing sparsest cofactor representations (Algorithm 1).

We then describe how to obtain a practical algorithm for computing sparse (not necessarily sparsest) representations by making two simplifications:

(1) We restrict the search space to a finite dimensional subspace by only considering cofactor representations with terms smaller than a designated bound.
(2) We use the sum of the absolute values of the coefficients, i.e., the ℓ_1-norm, of a representation as a complexity measure.

With these simplifications, we translate the problem of finding sparse cofactor representations into solving a linear programming problem. Our main result is Algorithm 3, which computes, starting from any given cofactor representation, a minimal one w.r.t. the conditions (1) and (2).

We also show that the second simplification does in fact impose no restriction for a class of ideals that appears frequently when translating operator statements. In particular, we prove that, under certain assumptions, Algorithm 3 computes a sparsest representation among all representations satisfying condition (1). Finally, we demonstrate the effectiveness of Algorithm 3 on several examples coming from operator statements.

Our algorithm relies on information provided by noncommutative signature-based Gröbner basis algorithms. Initially developed in the commutative setting to improve Gröbner basis computations, signature-based algorithms have been subject of extensive research,

*C. Hofstadler was supported by the Austrian Science Fund (FWF) grant P 32301.
†T. Verron was supported by the Austrian Science Fund (FWF) grant P 34872.

Conference'17, July 2017, Washington, DC, USA

see [9] for a survey of which. Recently, the authors proposed a generalization of these algorithms to the free algebra [14].

Signature-based algorithms compute, in addition to a Gröbner basis, some information on how the polynomials in that basis were computed. This additional information allows the algorithms not only to predict and avoid redundant computations, but also to compute a Gröbner basis of the syzygy module of the generators. In particular, signature-based algorithms compute this basis in a very structured way, and precisely this structure is what we exploit in our algorithm.

To the best of our knowledge, the problem at hand has not been studied in this form yet. While the complexity of the ideal membership problem itself has been studied extensively (especially in the commutative setting) [20–22], the problem of finding sparse cofactor representations seems to be unexplored. Proof simplification in automated theorem proving in general was addressed in e.g., [17, 28].

2 PRELIMINARIES

We recall basic definitions regarding noncommutative polynomials and signature-based Gröbner basis algorithms in the free algebra. For more details, we refer to [14]. For an introduction to noncommutative Gröbner bases without signatures, see [23, 29].

2.1 Free algebra

Throughout this paper, X is a finite set of indeterminates and $\langle X\rangle$ is the free monoid over X. For $m \in \langle X\rangle$, we denote by $|m|$ the length of m. Let K be a field and $K\langle X\rangle$ be the free algebra over X. We consider the elements in $K\langle X\rangle$ as *noncommutative polynomials*. For $F \subseteq K\langle X\rangle$, let (F) be the (two-sided) ideal generated by F.

A *monomial ordering* on $\langle X\rangle$ is a well-ordering $\leq$ compatible with the multiplication in $\langle X\rangle$, that is, $w \leq w'$ implies $awb \leq aw'b$ for all $a, b, w, w' \in \langle X\rangle$. We fix a monomial ordering $\leq$.

Example 2.1. The degree lexicographic ordering $\leq_{\text{deglex}}$ is a monomial ordering on $\langle X\rangle$ where two words $w, w' \in \langle X\rangle$ are first compared by their lengths and ties are broken by comparing the variables in w and w' from left to right using the lexicographic ordering $x_1 <_{\text{lex}} \cdots <_{\text{lex}} x_n$.

For $f \in K\langle X\rangle$, the support of f, denoted by $\operatorname{supp}(f)$, is the set of all monomials appearing in f. For $f \neq 0$, the leading monomial $\operatorname{lm}(f)$ of f is the maximal element w.r.t. $\leq$ in $\operatorname{supp}(f)$ and the degree $\deg(f)$ of f is $\deg(f) = \max_{m\in\operatorname{supp}(f)} |m|$. Additionally, we set $\deg(0) = -1$.

2.2 Free bimodule

We fix a family of polynomials $(f_1, \dots, f_r) \in K\langle X\rangle^r$, generating an ideal $I = (f_1, \dots, f_r)$. We extend the previous definitions to the *free $K\langle X\rangle$-bimodule* Σ of rank $r \in \mathbb{N}$, given by $\Sigma = (K\langle X\rangle \otimes K\langle X\rangle)^r$. The canonical basis of Σ is $\varepsilon_1, \dots, \varepsilon_r$, where $\varepsilon_i = (0, \dots, 0, 1 \otimes 1, 0, \dots, 0)$ with $1 \otimes 1$ appearing in the ith position for $i = 1, \dots, r$.

The set $M(\Sigma)$ of *module monomials* in Σ is given by $M(\Sigma) = \{a\varepsilon_i b \mid a, b \in \langle X\rangle, i = 1, \dots, r\}$. Every element $\alpha \in \Sigma$ has a unique representation of the form $\alpha = \sum_{i=1}^{d} c_i a_i \varepsilon_{j_i} b_i$ with nonzero $c_i \in K$ and pairwise different $a_i\varepsilon_{j_i}b_i \in M(\Sigma)$. We denote its support by $\operatorname{supp}(\alpha) = \{a_i\varepsilon_{j_i}b_i \mid i = 1, \dots, d\}$ and associate to it the polynomial $\overline{\alpha} := \sum_{i=1}^{d} c_i a_i f_{j_i} b_i \in I$. With this, the (weighted) degree of α is $\deg(\alpha) = \max_i \deg(a_i f_{j_i} b_i)$.

A *module ordering* on $M(\Sigma)$ is a well-ordering $\preceq$ compatible with scalar multiplication, that is, $\mu \preceq \mu'$ implies $a\mu b \preceq a\mu' b$ for all $\mu, \mu' \in M(\Sigma)$ and $a, b \in \langle X\rangle$. We fix a module ordering $\preceq$.

Example 2.2. The degree-over-position-over-term ordering $\preceq_{\text{DoPoT}}$ is a module ordering where two module monomials $a\varepsilon_i b, a'\varepsilon_j b' \in M(\Sigma)$ are first compared by their degrees and ties are broken by comparing the tuples (i, a, b) and (j, a', b') lexicographically using a monomial ordering for the monomial comparisons.

Definition 2.3. The *signature* $\mathfrak{s}(\alpha)$ of a nonzero $\alpha \in \Sigma$ is the maximal element w.r.t. $\preceq$ in $\operatorname{supp}(\alpha)$.

2.3 Signature-based algorithms

For the rest of this work, we assume that the monomial ordering $\leq$ and the module ordering $\preceq$ satisfy:

- $\leq$ and $\preceq$ are *compatible* in the sense that $a < b$ iff $a\varepsilon_i \prec b\varepsilon_i$ iff $\varepsilon_i a \prec \varepsilon_i b$ for all $a, b \in \langle X\rangle$ and $i = 1, \dots, r$;
- $\preceq$ is *fair*, meaning that the set $\{\mu' \in M(\Sigma) \mid \mu' \prec \mu\}$ is finite for all $\mu \in M(\Sigma)$;

Example 2.4. The module ordering $\preceq_{\text{DoPoT}}$ is fair and compatible with $\leq_{\text{deglex}}$.

A *labeled polynomial* $f^{[\alpha]}$ is a pair $(f, \alpha) \in I \times \Sigma$ with $f = \overline{\alpha}$. We call the set $I^{[\Sigma]} := \{f^{[\alpha]} \mid f \in I, \overline{\alpha} = f\}$ the *labeled module* generated by $f_1, \dots, f_r$. It forms a $K\langle X\rangle$-subbimodule of $I \times \Sigma$ with component-wise addition and scalar multiplication.

A syzygy of $I^{[\Sigma]}$ is an element $\gamma \in \Sigma$ such that $\overline{\gamma} = 0$. The set of syzygies of $I^{[\Sigma]}$, denoted by $\operatorname{Syz}(I^{[\Sigma]})$, forms a $K\langle X\rangle$-subbimodule of Σ. We recall the notion of Gröbner basis of $\operatorname{Syz}(I^{[\Sigma]})$.

Definition 2.5. A set $H \subseteq \operatorname{Syz}(I^{[\Sigma]})$ is a *Gröbner basis of* $\operatorname{Syz}(I^{[\Sigma]})$ *(up to signature $\sigma \in M(\Sigma)$)* if for all nonzero $\gamma \in \operatorname{Syz}(I^{[\Sigma]})$ (with $\mathfrak{s}(\gamma) \prec \sigma$), there exist $d \in \mathbb{N}$ and $\gamma_i \in H$, $c_i \in K$, $a_i, b_i \in \langle X\rangle$ such that $\gamma = \sum_{i=1}^{d} c_i a_i \gamma_i b_i$ and $\mathfrak{s}(a_i\gamma_i b_i) \preceq \mathfrak{s}(\gamma)$ for all $i = 1, \dots, d$.

In general, a syzygy module need not have a finite Gröbner basis. Nevertheless, Gröbner bases of $\operatorname{Syz}(I^{[\Sigma]})$ can be enumerated by increasing signatures using signature-based algorithms.

Theorem 2.6. *There exists an algorithm to correctly enumerate a Gröbner basis of* $\operatorname{Syz}(I^{[\Sigma]})$ *in increasing signature order. In particular, for all $\sigma \in M(\Sigma)$, stopping the algorithm at the first syzygy with signature $\succeq \sigma$ yields a finite Gröbner basis of* $\operatorname{Syz}(I^{[\Sigma]})$ *up to signature σ.*

Proof. The algorithm is [14, Algo. 1]. Its correctness is proved in [14, Thm. 42] and the fact that the signatures increase throughout is proved in [14, Lem. 43]. □

The algorithm uses signature-based Gröbner techniques, and also enumerates a (possibly infinite) *labeled Gröbner basis* of $I^{[\Sigma]}$. The definition and construction of labeled Gröbner bases are beyond the scope of this paper, but we note that among their properties, a labeled Gröbner basis is a set $G^{[\Sigma]} \subseteq I^{[\Sigma]}$ such that the set $\{f \mid f^{[\alpha]} \in G^{[\Sigma]}\}$ is a Gröbner basis of I. In particular, given $f \in I$, performing polynomial reductions by $G^{[\Sigma]}$ to reduce f to 0 and adding the labeling of the reducers yields a cofactor representation of f w.r.t. $f_1, \dots, f_r$.

Remark 2.7. The algorithm mentioned in the proof of Theorem 2.6 is inefficient as it has to perform a lot of expensive module arithmetic. A more efficient way of realizing Theorem 2.6 is to combine [14, Algo. 2,4] for computing *signature Gröbner bases*. These algorithms work with pairs $(f, \mathfrak{s}(\alpha))$ instead of labeled polynomials $f^{[\alpha]}$ and reconstruct the full module representations *a posteriori*, avoiding a lot of module arithmetic in this way.

A finite Gröbner basis of $\mathrm{Syz}(I^{[\Sigma]})$ up to signature σ, as output by the signature-based algorithm, provides an effective description of all syzygies with signature $\prec \sigma$. This is the crucial property of signature-based algorithms that we exploit in Section 4 to compute sparse cofactor representations.

3 DECIDABILITY AND COMPLEXITY

For the rest of this work, we restrict ourselves to the case $K = \mathbb{Q}$ and we fix a family of polynomials $(f_1, \ldots, f_r) \in \mathbb{Q}\langle X\rangle^r$ generating an ideal I. Any cofactor representation of an ideal member $f \in I$ can be identified with an element $\alpha \in \Sigma$ such that $\overline{\alpha} = f$. The *weight* of such a representation is given by the ℓ_0-"norm" $\|\alpha\|_0 := |\mathrm{supp}(\alpha)|$. Then, with the set $R(f) := \{\alpha \in \Sigma \mid \overline{\alpha} = f\}$ of *(cofactor) representations* of f, a *sparsest* (cofactor) representation of f corresponds to a minimal element w.r.t. $\|\cdot\|_0$ in $R(f)$. We denote the set of all such minimal elements by $R_0(f)$. If $f \notin I$, we set $R(f) = R_0(f) = \emptyset$.

Remark 3.1. Ideal membership in the free algebra is only semidecidable. Consequently, we can also not decide whether $R(f) = \emptyset$ or not.

Remark 3.2. The function $\|\cdot\|_0$ is not a norm as it is not homogeneous, but one can associate to it a metric called *Hamming distance*.

In the following, we study the decidability and complexity of computing cofactor representations of bounded weight. More precisely, we consider the following problem.

Problem 3.3 (Sparse cofactor representation)**.**
Input: $f, f_1, \ldots, f_r \in \mathbb{Q}\langle X\rangle$, $N \in \mathbb{N}$
Output: a cofactor representation $\alpha \in R(f)$ with $\|\alpha\|_0 \leq N$ if one exists, otherwise False.

We show that Problem 3.3 is decidable, and we give an algorithm reducing it to Problem 3.4 below of finding sparse solutions of linear systems [11, Problem MP5]. This not only yields an algorithm for computing sparsest cofactor representations if $f \in I$, but it, in principle, also provides a semidecision procedure for ideal membership. We focus on the first application here.

Problem 3.4 (Sparse solution of linear system [Min-RVLS])**.**
Input: $A \in \mathbb{Q}^{m\times n}, \mathbf{b} \in \mathbb{Q}^m$, $N \in \{0, \ldots, n\}$
Output: a vector $\mathbf{y} \in \mathbb{Q}^n$ with $A\mathbf{y} = \mathbf{b}$ and $\|\mathbf{y}\|_0 \leq N$ if one exists, otherwise False.

Problem 3.4 arises in many areas [1, 2, 8]. It is clearly decidable, for instance by looping over all N-subsets of $\{1, \ldots, n\}$ for possible sets of nonzero coefficients of solutions. Furthermore, it is known to be NP-hard, with the corresponding decision problem being NP-complete [11, Problem MP5].

In order to reduce the sparse cofactor representation problem to linear algebra, we need to constrain the solutions to a finite dimensional vector space, which requires to bound the degree of a solution. The degree of elements in $R(f)$ can be arbitrarily large, but we can bound the degree of *minimal* representations. A cofactor representation $\alpha = \sum_{i=1}^{d} c_i a_i \varepsilon_{j_i} b_i \in R(f)$ is *minimal* if no sub-sum is a syzygy, that is, $\sum_{i\in J} c_i a_i f_{j_i} b_i \neq 0$ for all non-empty subsets $J \subseteq \{1, \ldots, d\}$. To obtain the degree bound, we recall the notion of *(polynomial) rewriting* introduced in [25, Def. 2].

Definition 3.5. Let $f, g \in \mathbb{Q}\langle X\rangle$ and $a, b \in \langle X\rangle$ such that $\mathrm{supp}(f) \cap \mathrm{supp}(agb) \neq \emptyset$. For every $c \in K$, we say that f *can be rewritten to* $f + cagb \in \mathbb{Q}\langle X\rangle$ *by* g.

Furthermore, we say that f *can be rewritten to* h *by* $G \subseteq \mathbb{Q}\langle X\rangle$ if there are $h_0, \ldots, h_d \in \mathbb{Q}\langle X\rangle$, $h_d = f$, $h_0 = h$ and $g_1, \ldots, g_d \in G$ such that h_k can be rewritten to h_{k-1} by g_k for all $k = 1, \ldots, d$.

Rewriting can be considered as a weaker form of polynomial reduction, not requiring that a polynomial gets "simplified" by a rewriting step. Nevertheless, f can be rewritten to zero by $\{f_1, \ldots, f_r\}$ if and only if $f \in I$, see [25, Lem. 4]. More importantly, we can show that any minimal representation of f can be obtained by rewriting f to 0 by $\{f_1, \ldots, f_r\}$ and logging the rewriting steps.

Lemma 3.6. *Let* $f \in (f_1, \ldots, f_r)$ *and* $\alpha = \sum_{i=1}^{d} c_i a_i \varepsilon_{j_i} b_i \in R(f)$ *be a minimal representation of* f*. Furthermore, for* $k = 0, \ldots, d$*, let* $h_k = \sum_{i=1}^{k} c_i a_i f_{j_i} b_i$*. In particular,* $h_d = f$ *and* $h_0 = 0$*. Then, possibly after reordering the summands of* α*,* h_k *can be rewritten to* h_{k-1} *by* $\{f_1, \ldots, f_r\}$ *for all* $k = 1, \ldots, d$*.*

Proof. We perform induction on the weight d of a minimal representation of f. For $d = 0$ there is nothing to prove. Assume now that $d > 0$ and that the result is proven for polynomials with a minimal representation of weight $d - 1$. Because α is a minimal representation, f cannot be 0. Since $f = \overline{\alpha} = \sum_{i=1}^{d} c_i a_i f_{j_i} b_i$, the support of f is contained in the union of the supports of the $a_i f_{j_i} b_i$, and there exists $1 \leq k \leq d$ such that $\mathrm{supp}(f) \cap \mathrm{supp}(a_k f_{j_k} b_k) \neq \emptyset$. Possibly after reordering the summands of α, we can assume $k = d$. So f can be rewritten to $h_{d-1} = f - c_d a_d f_{j_d} b_d$ using $f_{j_d} \in \{f_1, \ldots, f_r\}$. Furthermore, $h_{d-1} = \sum_{i=1}^{d-1} c_i a_i f_{j_i} b_i$ has a minimal representation of weight $d - 1$, because adding one term results in a minimal representation of weight d for f. So, by induction hypothesis, this representation of h_{d-1} is (up to reordering of the summands) a sequence of rewritings by $\{f_1, \ldots, f_r\}$. □

Another crucial property of rewriting is that we can bound the degree of the output in terms of the degree of the input and the *degree difference* of the rewriter. The *degree difference* $\mathrm{degdiff}(g)$ of a nonzero $g \in \mathbb{Q}\langle X\rangle$ is $\mathrm{degdiff}(g) = \deg(g) - \mathrm{degmin}(g)$, where $\mathrm{degmin}(g) = \min_{m\in\mathrm{supp}(g)} |m|$.

Lemma 3.7. *Let* $f, g \in \mathbb{Q}\langle X\rangle$ *and* $c \in \mathbb{Q}$*,* $a, b \in \langle X\rangle$*. If* f *can be rewritten to* $h = f + cagb$ *by* g*, then* $\max(\deg(h), \deg(agb)) \leq \deg(f) + \mathrm{degdiff}(g)$*.*

Proof. By definition, there exists a monomial m in $\mathrm{supp}(g)$ such that $amb \in \mathrm{supp}(f)$, so $|amb| \leq \deg(f)$, or equivalently $|a| + |b| \leq \deg(f) - |m|$. Since $m \in \mathrm{supp}(g)$, $|m| \geq \mathrm{degmin}(g)$, and all in all,

$$\begin{aligned}\deg(agb) &= |a| + |b| + \deg(g)\\ &\leq \deg(f) - \mathrm{degmin}(g) + \deg(g) = \deg(f) + \mathrm{degdiff}(g).\end{aligned}$$

As $\deg(h) \leq \max(\deg(f), \deg(agb))$ and $\mathrm{degdiff}(g) \geq 0$, we conclude that also $\deg(h) \leq \deg(f) + \mathrm{degdiff}(g)$. □

Combining Lemma 3.6 and 3.7, we obtain a bound on the degree of minimal cofactor representations of f of bounded weight. Since any sparsest cofactor representation is, in particular, minimal, this also yields a bound on the degree of sparsest representations.

Corollary 3.8. *Let $f \in (f_1, \dots, f_r)$, $N \in \mathbb{N}$ and $\alpha \in R(f)$ be a minimal representation of f. If $\|\alpha\|_0 \leq N$, then* $\deg(\alpha) \leq \deg(f) + N \max_i \operatorname{degdiff}(f_i)$.

Proof. Write α as $\alpha = \sum_{i=1}^{d} c_i a_i \varepsilon_{j_i} b_i$. By definition, $\deg(\alpha) = \max_i \deg(a_i f_{j_i} b_i)$, and, according to Lemma 3.6, each $a_i f_{j_i} b_i$ is a rewriter in a rewriting sequence from f to 0. Thus, Lemma 3.7 shows inductively that $\deg(\alpha) \leq \deg(f) + \|\alpha\|_0 \max_i \operatorname{degdiff}(f_i)$ and the result follows since $\|\alpha\|_0 \leq N$. □

As a consequence, we can state an algorithm for computing a cofactor representation of weight bounded by $N \in \mathbb{N}$, reducing to the problem of finding a sparse solution of a linear system.

Algorithm 1: Sparse cofactor representation

Input: $f, f_1, \dots, f_r \in \mathbb{Q}\langle X\rangle$, $N \in \mathbb{N}$
Output: $\alpha \in R(f)$ with $\|\alpha\|_0 \leq N$ if one exists, otherwise False

1 $D \leftarrow \deg(f) + N \max_i \operatorname{degdiff}(f_i)$;
2 $L \leftarrow \{a f_i b \mid a, b \in \langle X\rangle, \deg(a f_i b) \leq D, i = 1, \dots, r\}$;
3 **return** a $\mathbb{Q}$-linear combination of elements of L equal to f with $\leq N$ nonzero summands if one exists, otherwise False;

Corollary 3.9. *Algorithm 1 terminates and is correct.*

Proof. The algorithm reduces the problem to that of finding sparse solutions of a linear system. This problem is decidable, so the algorithm terminates.

There exists a representation of f of weight $\leq N$ if and only if there exists a *minimal* representation α of f of weight $\leq N$. By Corollary 3.8, this representation is given by a linear combination of weight $\|\alpha\|_0 \leq N$ consisting of elements of L. So the algorithm is correct. □

It is also possible to describe a reduction of Problem 3.4 to Problem 3.3, which allows us to characterize the complexity of the problem of finding sparse representations.

Theorem 3.10. *The problem of, given $f, f_1, \dots, f_r \in \mathbb{Q}\langle X\rangle$ and $N \in \mathbb{N}$ (in unary form), deciding whether there exists a cofactor representation of f of weight at most N, is* NP*-complete.*

Proof. The decision problem associated to Problem 3.4 is NP-complete [11, Problem MP5]. Given an input $A, \mathbf{b}, N$ to that problem, introduce one variable x_i for each row of A, interpret each column of A as the polynomial $f_j = \sum_i A_{i,j} x_i$, and the right-hand side as the polynomial $f = \sum_i \mathbf{b}_i x_i$. There is a one-to-one correspondence between solutions with N nonzero entries to the linear systems, and cofactor representations of f with weight N . So the problem of finding a representation of weight at most N is also NP-hard.

Furthermore, if there exists a representation of weight $\leq N$, then there exists one with degree $\leq \deg(f) + N \max_i \operatorname{degdiff}(f_i)$, which makes it polynomial size in N and the size of the input polynomials. Furthermore, its validity can be verified in polynomial time. So the problem is NP, and therefore NP-complete. □

Remark 3.11. The requirement that N be given in unary format is necessary because unlike Problem 3.4, the input of Problem 3.3 is not at least of size N. If N is given in binary format, the decision problem is still NP-hard but no longer NP, because the degree bound is not polynomial in $\log(N)$. Also note that Algorithm 1 does not provide a polynomial time reduction of Problem 3.3 to Problem 3.4, even as a function of N.

We note that the last step of Algorithm 1 is infeasible for non-trivial examples. To illustrate this point, we consider the following simple statement about the Moore-Penrose inverse.

Theorem 3.12 ([15, Ch. 5.7 Fact 11])**.** *Let A be an invertible matrix with inverse B. Then B is the Moore-Penrose inverse of A.*

Example 3.13. Theorem 3.12 can be encoded in terms of the ideal membership $b - a^\dagger \in (F)$ with $F = \{ab - 1, ba - 1, aa^\dagger a - a, a^\dagger a a^\dagger - a^\dagger, (a^\dagger)^* a^* - a a^\dagger, a^* (a^\dagger)^* - a^\dagger a\}$. A cofactor representation of $b - a^\dagger$ certifying the ideal membership is

$$b - a^\dagger = a^\dagger(ab - 1) - b(ab - 1) - b(aa^\dagger a - a)b + (ba - 1)a^\dagger ab. \quad (1)$$

This cofactor representation consists of 4 terms. To see if there exists a representation with ≤ 3 terms, we can call Algorithm 1 with $N = 3$. The set L constructed in this case consists of more than 17 000 elements. This is too large to test all 3-subsets exhaustively.

Using the techniques of Section 4, we will see that a much smaller set of elements is sufficient and by applying the results of Section 4.5, we will be able to verify that (1) is in fact a sparsest representation of $b - a^\dagger$. Hence, we can conclude with the following result.

Fact 3.14. *A shortest proof of Theorem 3.12 consists of 4 steps, given by $B = BAB = BAA^\dagger AB = BAA^\dagger = A^\dagger$.*

4 COMPUTING SPARSE REPRESENTATIONS

We have seen in the previous section that computing sparsest cofactor representations is equivalent to the NP-hard problem of finding sparsest solutions of a linear system. Several methods have been proposed to obtain approximate solutions of the latter [2, 4, 19] by using other measures as proxies for the sparsity of a solution and by minimising over them. One of these methods, called *Basis Pursuit* [2], uses the ℓ_1-norm as an approximation for the sparsity of a solution.

In the following, we follow the Basis Pursuit approach and use the ℓ_1-norm $\|\alpha\|_1 := \sum_{i=1}^{d} |c_i|$ of $\alpha = \sum_{i=1}^{d} c_i a_i \varepsilon_{j_i} b_i$ as a surrogate complexity measure of a cofactor representation. The advantage of this approach is that an ℓ_1-minimal solution of a linear system can be found efficiently using linear programming. Additionally, we use the effective description of the syzygy module provided by signature-based algorithms to reduce the size of the linear system that we have to consider.

Based on Corollary 3.8, it suffices to consider only cofactor representations up to a degree bound when computing minimal representations. Here, we, more generally, restrict to representations with signature less than a designated bound $\sigma \in M(\Sigma)$. Since the module ordering is assumed to be fair, this ensures that we work in a finite dimensional vector space. If the module ordering is also

compatible with the degree, i.e., if $\deg(\alpha) \leq \deg(\beta)$ implies $\alpha \preceq \beta$, this includes all cofactor representations of degree $< \deg(\sigma)$.

So, formally, we seek a minimal element w.r.t. $\|\cdot\|_1$ in the set $R(f,\sigma) := \{\alpha \in R(f) \mid \mathfrak{s}(\alpha) \prec \sigma\}$ of cofactor representations of f *up to signature* $\sigma \in M(\Sigma)$. We denote the set of all such ℓ_1-minimal elements by $R_1(f,\sigma)$. Analogously, we let $R_0(f,\sigma)$ be the set of all minimal elements w.r.t. $\|\cdot\|_0$ in $R(f,\sigma)$.

The results in this section rely on the fact that we have some $\alpha \in R(f,\sigma)$. However, in general, for σ too small, the set $R(f,\sigma)$ can be empty, even if $R(f) \neq \emptyset$. To resolve this issue, we assume that we have a cofactor representation $\alpha \in R(f)$ and that σ is chosen so that $\sigma \succ \mathfrak{s}(\alpha)$. Note that this assumption, in particular, implies that $f \in (f_1,\dots,f_r)$. Such α can be obtained, for example, by reducing f to zero using a (partial) labeled Gröbner basis and keeping track of the reductions. With this in mind, we assume that $R(f,\sigma) \neq \emptyset$.

In the following, we describe Algorithm 3, which allows to compute an element in the set $R_1(f,\sigma)$. To this end, we denote by $I^{[\Sigma]}$ the labeled module generated by $f_1,\dots,f_r$ and by H_σ a Gröbner basis of $\mathrm{Syz}(I^{[\Sigma]})$ up to signature σ.

The general idea of Algorithm 3 is still to reduce the problem of computing sparse cofactor representations to computing certain solutions of a linear system. However, instead of choosing all polynomials af_ib to form the linear system like Algorithm 1 does, we use the information provided by H_σ to trim this set. More precisely, we find a finite set of module monomials $B = \{\mu_1,\dots,\mu_d\} \subseteq M(\Sigma)$ such that $R_i(f,\sigma)$, $i = 0,1$, has non-empty intersection with the $\mathbb{Q}$-vector space generated by B and then only consider the polynomials $\{\overline{\mu}_1,\dots,\overline{\mu}_d\}$ to form the linear system. Furthermore, we now no longer seek a sparsest solution of the resulting system but an ℓ_1-minimal solution, which can be found with linear programming.

It remains to discuss how to find a suitable basis B and how to translate the problem of finding ℓ_1-minimal solutions of a linear system into a linear programming problem.

4.1 Finding a suitable basis B

Algorithm 1 essentially uses the basis $B = \{a\varepsilon_i b \mid a,b \in \langle X\rangle, i = 1,\dots,r, \mathfrak{s}(a\varepsilon_i b) \prec \sigma\}$, which leads to finite dimensional, yet infeasibly large, linear systems. Using a Gröbner basis of $\mathrm{Syz}(I^{[\Sigma]})$ up to signature σ, we can drastically reduce the dimension of the search space. To this end, we extend the notion of rewriting to module elements.

Definition 4.1. Let $\alpha,\beta \in \Sigma$ and $a,b \in \langle X\rangle$ such that $\mathrm{supp}(\alpha) \cap \mathrm{supp}(a\beta b) \neq \emptyset$. For every $c \in \mathbb{Q}$, we say that α *can be rewritten to* $\alpha + ca\beta b$ *by* β.

Furthermore, we say that α *can be rewritten to* β *by* $H \subseteq \Sigma$ if there are $\beta_0,\dots,\beta_d \in \Sigma$, $\beta_d = \alpha$, $\beta_0 = \beta$ and $\gamma_1,\dots,\gamma_d \in H$ such that β_k can be rewritten to β_{k-1} by γ_k for all $k = 1,\dots,d$.

With this, we can state a module version of Lemma 3.6. We note that we state all results in this section for both the ℓ_0-"norm" and the ℓ_1-norm to emphasize that they hold for both complexity measures likewise and that the restriction to $\|\cdot\|_1$ only comes later for the linear programming.

Lemma 4.2. *Let $i \in \{0,1\}$. Furthermore, let $\alpha \in R(f,\sigma)$, $\alpha_i \in R_i(f,\sigma)$, and let H_σ be a Gröbner basis of $\mathrm{Syz}(I^{[\Sigma]})$ up to signature σ. Then α can be rewritten to α_i by H_σ. In particular, this rewriting can be done so that the signature of every rewriter $a_j\gamma_j b_j$ is less than σ.*

To prove Lemma 4.2, we make use of the fact that the ℓ_0-"norm" and the ℓ_1-norm are linear for elements of disjoint support.

Lemma 4.3. *Let $\alpha,\beta \in \Sigma$ such that $\mathrm{supp}(\alpha) \cap \mathrm{supp}(\beta) = \emptyset$. Then $\|\alpha+\beta\|_i = \|\alpha\|_i + \|\beta\|_i$ for $i = 0,1$.*

Proof of Lemma 4.2. The difference $\alpha - \alpha_i$ is a syzygy with signature $\prec \sigma$. Since H_σ is a Gröbner basis of $\mathrm{Syz}(I^{[\Sigma]})$ up to signature σ, there exist $d \in \mathbb{N}$ and $\gamma_j \in H_\sigma, c_j \in K, a_j,b_j \in \langle X\rangle$ such that $\alpha_i = \alpha - \sum_{j=1}^d c_j a_j\gamma_j b_j$ and $\mathfrak{s}(a_j\gamma_j b_j) \preceq \max\{\mathfrak{s}(\alpha),\mathfrak{s}(\alpha_i)\} \prec \sigma$ for all j. Now, we essentially follow the proof from Lemma 3.6 and perform induction on d.

The case $d = 0$ is clear. Assume now that $d > 0$ and that the result is proven for all pairs (α,α_i) such that $\alpha - \alpha_i$ has a representation with $d-1$ terms. Let $\beta = \sum_{j=1}^d c_j a_j\gamma_j b_j$. If $\beta = 0$, we are done since $\alpha = \alpha_i$. So assume $\beta \neq 0$, which implies $\|\beta\|_i > 0$. Then we must have $\mathrm{supp}(\alpha) \cap \mathrm{supp}(\beta) \neq \emptyset$, as otherwise Lemma 4.3 would yield the contradiction $\|\alpha_i\|_i = \|\alpha-\beta\|_i = \|\alpha\|_i + \|\beta\|_i > \|\alpha\|_i \geq \|\alpha_i\|_i$, where the last inequality follow from the minimality of $\|\alpha_i\|_i$. Thus, we have $\mathrm{supp}(\alpha) \cap \mathrm{supp}(a_j\gamma_j b_j) \neq \emptyset$ for some $1 \leq j \leq d$. W.l.o.g. assume $j = d$. Hence, α can be rewritten to $\beta_{d-1} = \alpha - c_d a_d\gamma_d b_d$ by $\gamma_d \in H_\sigma$. Note that $\mathfrak{s}(a_d\gamma_d b_d) \prec \sigma$. Since $\beta_{d-1} - \alpha_i = \sum_{j=1}^{d-1} c_j a_j\gamma_j b_j$ has a representation with $d-1$ terms, the induction hypothesis implies that β_{d-1} can be rewritten to α_i by H_σ using only rewriters $a_j\gamma_j b_j$ with signature $\prec \sigma$. □

Lemma 4.2 says that any $\alpha \in R(f,\sigma)$ can be rewritten to each element in $R_i(f,\sigma)$, $i = 0,1$, by H_σ using only rewriters with signature bounded by σ. Consequently, to find a suitable basis B, it suffices, starting from some α, to only choose those syzygies that can appear in such rewriting sequences. Finding these elements is a purely combinatorial problem that can be solved without performing any actual rewriting steps. This leads to Algorithm 2, in which we collect precisely all those relevant syzygies. Algorithm 2 can be considered as an adaptation of the symbolic preprocessing in the F4 algorithm [10]. In the following, for $V \subseteq \Sigma$, let $\mathrm{supp}(V) = \bigcup_{\gamma\in V} \mathrm{supp}(\gamma)$.

Algorithm 2: Finding relevant syzygies

Input: $\alpha \in R(f,\sigma)$, H_σ a GB of $\mathrm{Syz}(I^{[\Sigma]})$ up to sig. σ
Output: $V \subseteq \mathrm{Syz}(I^{[\Sigma]})$ s.t. $R_i(f,\sigma) \subseteq \alpha + \mathrm{span}_{\mathbb{Q}}(V)$, $i = 0,1$

```
V ← ∅;
todo ← supp(α); done ← ∅;
while todo ≠ ∅ :
    select μ ∈ todo, remove it, and add it to done;
    new ← {aγb | a,b ∈ ⟨X⟩, γ ∈ H_σ, μ ∈ supp(aγb),
           𝔰(aγb) ≺ σ};
    todo ← todo ∪ (supp(new) \ done);
    V ← V ∪ new;
return V;
```

Proposition 4.4. *Algorithm 2 terminates and is correct.*

Proof. The conditions on the elements in new ensure that only module monomials smaller than σ are inserted into todo. Furthermore, each monomial is processed at most once. Consequently, termination follows from the fact that there are only finitely many monomials smaller than σ (recall that $\preceq$ is fair). Correctness follows from Lemma 4.2. □

Using Algorithm 2, we can set $B = \operatorname{supp}(\alpha) \cup \operatorname{supp}(V)$ as a basis of the search space, where V is the output of the algorithm given α and H_σ as input. In many cases, this set is small enough to reasonably work with.

4.2 Detecting redundant syzygies

As an optional step, we can remove redundant elements from V before forming the basis B in order to obtain a smaller basis, and thus, a smaller linear program to solve. More precisely, since we only want to compute one element in $R_i(f,\sigma)$, $i = 0, 1$, we can remove syzygies as long as we can ensure that there remains at least one rewriting sequence from α to at least one element in $R_i(f,\sigma)$. We mention two basic techniques that turned out useful in practice.

The first technique allows to remove syzygies from V that consist mostly of terms that appear in no other element. Such syzygies cannot lead to simpler representations. To make this statement precise, for $W \subseteq V$ and $\beta = \sum_j c_j\mu_j \in W$ with $c_j \in K$, $\mu_j \in M(\Sigma)$, we denote

$$\beta_U \coloneqq \textstyle\sum_j c_j\mu_j \text{ with } j \text{ such that } \mu_j \notin \operatorname{supp}((V \cup \{\alpha\}) \setminus \{\beta\}),$$
$$\beta_V \coloneqq \textstyle\sum_j c_j\mu_j \text{ with } j \text{ such that } \mu_j \in \operatorname{supp}((V \cup \{\alpha\}) \setminus W).$$

Intuitively, the element β_U contains all those terms of β that are unique to β and that appear in no other element of $V \cup \{\alpha\}$, and β_V contains those terms that appear in β as well as in elements outside of W.

Proposition 4.5. *Let $i \in \{0,1\}$. Furthermore, let $\alpha \in R(f,\sigma)$ and $V \subseteq \operatorname{Syz}(I^{[\Sigma]})$ such that*

$$(\alpha + \operatorname{span}_{\mathbb{Q}}(V)) \cap R_i(f,\sigma) \neq \emptyset.$$

If $W \subseteq V$ satisfies $\|\beta_V\|_i \leq \|\beta_U\|_i$ for all $\beta \in W$, then

$$(\alpha + \operatorname{span}_{\mathbb{Q}}(V \setminus W)) \cap R_i(f,\sigma) \neq \emptyset.$$

Proposition 4.5 provides a sufficient condition for a subset $W \subseteq V$ to be redundant. In order to prove this, we need the following two lemmas. The first one states that the required property of W extends to the whole linear span. To this end, we extend the definition of β_U and β_V to elements $\beta = \sum_j b_j\beta_j \in \operatorname{span}_{\mathbb{Q}}(W)$, where $b_j \in \mathbb{Q}$ and $\beta_j \in W$, by $\beta_U \coloneqq \sum_j b_j\beta_{j,U}$ and $\beta_V \coloneqq \sum_j b_j\beta_{j,V}$.

Lemma 4.6. *Let V, W be as in Prop. 4.5. If $\beta \in \operatorname{span}_{\mathbb{Q}}(W)$, then $\|\beta_V\|_i \leq \|\beta_U\|_i$.*

Proof. Write $\beta = \sum_j b_j\beta_j$ with nonzero $b_j \in \mathbb{Q}$ and $\beta_j \in W$. By assumption $\|\beta_{j,V}\|_i \leq \|\beta_{j,U}\|_i$ for all j. Furthermore, all $\beta_{j,U}$ have pairwise different supports as they consist of the monomials that are unique to each β_j. So Lemma 4.3 implies that $\|\cdot\|_i$ is linear on linear combinations of the $\beta_{j,U}$. Using this and the triangular inequality, we get with $c_j = 1$ if $i = 0$ and $c_j = |b_j|$ if $i = 1$:

$$\begin{aligned}\|\beta_V\|_i &\leq \textstyle\sum_j \|b_j\beta_{j,V}\|_i = \sum_j c_j\|\beta_{j,V}\|_i \\ &\leq \textstyle\sum_j c_j\|\beta_{j,U}\|_i = \sum_j \|b_j\beta_{j,U}\|_i = \|\beta_U\|_i.\end{aligned}$$ □

The second lemma provides a lower bound on the norm of sums $\gamma + \beta \in \alpha + \operatorname{span}_{\mathbb{Q}}(W)$.

Lemma 4.7. *Let α, V, W be as in Prop. 4.5. If $\beta \in \operatorname{span}_{\mathbb{Q}}(W)$ and $\gamma \in \alpha + \operatorname{span}_{\mathbb{Q}}(V \setminus W)$, then $\|\gamma + \beta\|_i \geq \|\gamma\|_i - \|\beta_V\|_i + \|\beta_U\|_i$.*

Proof. Let $\beta' = \beta - (\beta_U + \beta_V)$. By definition, β_U and β' have pairwise different supports. Furthermore, $\gamma + \beta_V$ does not share a monomial with β_U and β' as $\operatorname{supp}(\gamma+\beta_V) \subseteq \operatorname{supp}((V \cup \{\alpha\}) \setminus W)$ and all monomials of β that lie in this set are collected in β_V. Therefore, Lemma 4.3 and the inverse triangle inequality imply

$$\begin{aligned}\|\gamma + \beta\|_i &= \|\gamma + \beta_V\| + \|\beta_U\|_i + \|\beta'\|_i \\ &\geq \|\gamma + \beta_V\| + \|\beta_U\|_i \geq \|\gamma\|_i - \|\beta_V\|_i + \|\beta_U\|_i.\end{aligned}$$ □

Proof of Proposition 4.5. We claim that removing, if present, elements from W from a representation $\delta \in \alpha + \operatorname{span}_{\mathbb{Q}}(V)$ cannot increase the norm. This implies the assertion of the proposition. To prove our claim, write δ as $\delta = \gamma + \beta$ with $\gamma \in \alpha + \operatorname{span}_{\mathbb{Q}}(V \setminus W)$ and $\beta \in \operatorname{span}_{\mathbb{Q}}(W)$. Now, Lemma 4.6 and 4.7, show that $\|\delta\|_i = \|\gamma + \beta\|_i \geq \|\gamma\|_i - \|\beta_V\|_i + \|\beta_U\|_i \geq \|\gamma\|_i$. □

The redundancy test provided by Proposition 4.5 is computationally fairly cheap to check for a given set $W \subseteq V$. However, finding suitable candidates for W is not so trivial. In our implementation, we test all singletons $\{\beta\} \subseteq V$ and all subsets $\{\beta, \gamma\} \subseteq V$ where $\operatorname{supp}(\beta) \cap \operatorname{supp}(\gamma) \neq \emptyset$ and where β and γ consist to at least a third of unique monomials that appear in no other element in V. This empirically provided the best trade-off between efficiency in applying the criterion and the effect it had on pruning V.

The second method does not directly allow to detect redundant elements in V. Instead it can be considered as an auxiliary technique that can cause additional applications of Proposition 4.5. The idea is to replace elements in V by linear combinations so that the number of occurrences of certain monomials is reduced. In particular, by exploiting the fact that

$$\operatorname{span}_{\mathbb{Q}}(V \cup \{\alpha - \beta, \gamma + \beta\}) = \operatorname{span}_{\mathbb{Q}}(V \cup \{\alpha - \beta, \alpha + \gamma\}), \quad (2)$$

we can reduce the number of occurrences of β at the cost of increasing the occurrences of α.

In our implementation, we apply this technique to all binomial syzygies $\mu - \sigma \in V$. After removing all occurrences of σ, Proposition 4.5 allows to delete the binomial syzygy from V. Additionally, we apply (2) randomly to elements $\alpha - \beta$ where $\|\beta\|_i > c\|\alpha\|_i$ for fixed $c > 1$. Often, this process triggers further invocations of Proposition 4.5 to remove elements from V. Table 1 shows the efficiency of the two methods presented in this section.

4.3 Translation into linear program

Once we have obtained a reasonable basis of module monomials $B = \{\mu_1, \ldots, \mu_d\}$ such that the $\mathbb{Q}$-vector space generated by B has non-empty intersection with $R_i(f,\sigma)$, $i = 0, 1$, we can set up a linear system $A\mathbf{y} = \mathbf{b}$, where A is the matrix of size $s \times d$, with $s = |\bigcup_j \operatorname{supp}(\overline{\mu}_j)|$, whose jth column contains the coefficients of $\overline{\mu}_j$. Similarly, $\mathbf{b}$ is a vector of size s containing the coefficients of f. The matrix A bears resemblance to the matrices appearing in Gröbner basis computations such as the F4 algorithm, aside from two main differences. In Gröbner basis computations, polynomials are encoded as the rows of a matrix and the columns have to be ordered w.r.t. the

(polynomial) monomial ordering. In our approach, polynomials are encoded as the columns and the order of the columns is irrelevant.

Every ℓ_i-minimal solution of $A\mathbf{y} = \mathbf{b}$ corresponds to an element in $R_i(f, \sigma)$. As noted before, computing ℓ_0-minimal, i.e., sparsest, solutions is NP-hard. Therefore, we restrict ourselves to the case $i = 1$ and consider the problem

$$(P_1): \quad \min_{\mathbf{y}} \|\mathbf{y}\|_1 \quad \text{subject to} \quad A\mathbf{y} = \mathbf{b},$$

where $\|\mathbf{x}\|_1 = \sum_j |\mathbf{x}_j|$. It is well-known that (P_1) can be recast as a linear program, see e.g. [2, Sec. 3.1]. A linear program (in standard form) [27] is an optimization problem for $\mathbf{v} \in \mathbb{Q}^t$ of the form

$$(LP): \quad \min_{\mathbf{v}} \mathbf{c}^T\mathbf{v} \quad \text{subject to} \quad U\mathbf{v} = \mathbf{w}, \quad \mathbf{v} \geq 0,$$

where $\mathbf{v} \geq 0$ is to be understood component-wise. The problem (P_1) can be equivalently formulated as a linear program by setting

$$t = 2d, \quad \mathbf{c}^T = (1, \ldots, 1), \quad U = (A \mid -A), \quad \mathbf{v} = \begin{pmatrix}\mathbf{p}\\ \mathbf{q}\end{pmatrix}, \quad \mathbf{w} = \mathbf{b},$$

with vectors $\mathbf{p}, \mathbf{q} \in \mathbb{Q}^d$. This linear program can then be solved efficiently using the simplex algorithm [6] or interior-point methods [24] and a solution $\mathbf{y}$ of (P_1) is given by $\mathbf{y} = \mathbf{p} - \mathbf{q}$.

4.4 Putting everything together

Finally, we combine the results of the previous sections to form Algorithm 3 for computing an element in $R_1(f, \sigma)$. In the algorithm, $I^{[\Sigma]}$ denotes the labeled module generated by $f_1, \ldots, f_r$.

Algorithm 3: ℓ_1-minimal cofactor representation

Input: $(f_1, \ldots, f_r) \in \mathbb{Q}\langle X\rangle^r$, $f \in (f_1, \ldots, f_r)$, $\sigma \in M(\Sigma)$, $\alpha \in R(f, \sigma)$

Output: an element in $R_1(f, \sigma)$

1 $H_\sigma \leftarrow$ GB of $\mathrm{Syz}(I^{[\Sigma]})$ up to sig. σ;
2 $V \leftarrow$ apply Algorithm 2 to α and H_σ;
3 $V \leftarrow$ prune V using the techniques from Section 4.2;
4 $\{\mu_1, \ldots, \mu_d\} \leftarrow \mathrm{supp}(V \cup \{\alpha\})$;
5 $A \leftarrow$ matrix with columns containing the coeffs of $\overline{\mu}_1, \ldots, \overline{\mu}_d$;
6 $\mathbf{b} \leftarrow$ vector containing the coefficients of f;
7 $\mathbf{v} \leftarrow$ solution of the linear program (LP) with

$$\mathbf{c}^T = (1, \ldots, 1), \quad U = (A \mid -A), \quad \mathbf{v} = \begin{pmatrix}\mathbf{p}\\ \mathbf{q}\end{pmatrix}, \quad \mathbf{w} = \mathbf{b};$$

8 **return** $\sum_{i=1}^{d} (\mathbf{p}_i - \mathbf{q}_i)\mu_i$;

Theorem 4.8. *Algorithm 3 terminates and is correct.*

Proof. Termination follows from the fact that H_σ can be computed in finite time by Theorem 2.6, and from Proposition 4.4. Correctness follows from the discussions in Section 4.1, 4.2 and 4.3. □

Remark 4.9. Algorithm 3 weighs each monomial μ_i equally by a weight of 1. It is also possible to weigh the monomials differently by changing the vector $\mathbf{c}$ so that $\mathbf{c}_i$ encodes the weight of μ_i. This allows, for example, to weigh monomials by their degree, yielding representations that prefer monomials with small degree. In this case, the output of the algorithm is no longer guaranteed to be in $R_1(f, \sigma)$.

4.5 Special case: totally unimodular matrices

In general, the output of Algorithm 3 need not be a sparsest representation of f up to signature σ, i.e., it need not be an element in $R_0(f, \sigma)$. In this section, we discuss a special case when this indeed true. To this end, we consider the linear system $A\mathbf{y} = \mathbf{b}$ constructed in Algorithm 3. We are interested in situations where the augmented matrix $(A \mid \mathbf{b})$ is *totally unimodular* as defined below.

Definition 4.10. A matrix $T \in \{-1, 0, 1\}^{m\times n}$ is called *totally unimodular* if every square submatrix of T has determinant 0 or ±1.

Theorem 4.11. *Let A and $\mathbf{b}$ as constructed in Algorithm 3. If the augmented matrix $(A \mid \mathbf{b})$ is totally unimodular, then the output of Algorithm 3 is an element in $R_0(f, \sigma)$.*

In order to prove the theorem, we take a closer look at the coefficients of the sparsest and ℓ_1-minimal solutions of $A\mathbf{y} = \mathbf{b}$. It is well-known that totally unimodular coefficient matrices and integer right-hand sides yield integer optima for linear programs [27, Thm. 19.1]. The following lemma, extends this statement under slightly stricter assumptions.

Lemma 4.12. *Let the augmented matrix $(A \mid \mathbf{b})$ be totally unimodular. If $A\mathbf{y} = \mathbf{b}$ is solvable, then any sparsest or ℓ_1-minimal solution $\mathbf{y}$ satisfies $\mathbf{y} \in \{-1, 0, 1\}^d$.*

Proof. Since removing linearly dependent rows does not change the solution set of a solvable system, we can assume that A has full row rank $s = \mathrm{rank}(A)$.

Sparsest solution. The columns of A corresponding to the nonzero entries of $\mathbf{y}$ have to be linearly independent (otherwise there would exist a sparser solution). We can extend them by further columns of A to obtain an invertible $s \times s$ matrix A'. Then $A'\mathbf{y}' = \mathbf{b}$, where $\mathbf{y}'$ contains those coordinates of $\mathbf{y}$ that correspond to the columns of A that are in A'. By assumption $\det(A') = \pm 1$. Furthermore, the matrix A'_i obtained by replacing the ith column of A' by $\mathbf{b}$ is – up to permutation of columns – a submatrix of $(A \mid \mathbf{b})$. Consequently, $\det(A'_i) \in \{-1, 0, 1\}$ and applying Cramer's rule shows $\mathbf{y}'_i = \frac{\det(A'_i)}{\det(A')} \in \{-1, 0, 1\}$. Then the result follows since any coordinate of $\mathbf{y}$ which does not appear in $\mathbf{y}'$ has to be zero.

ℓ_1-minimal solution. We consider the equivalent linear program (LP) and note that $\mathbf{y} \in \{-1, 0, 1\}^d$ if and only if $\mathbf{v} \in \{0, 1\}^{2d}$. If $\mathbf{v}$ is a solution of (LP), then it has to be a basic feasible solution. This means $\|\mathbf{v}\|_0 \leq s$ and that the columns of U that correspond to the nonzero coordinates of $\mathbf{v}$ can be extended to an invertible $s \times s$ submatrix U' of U. Since $(U \mid \mathbf{b}) = (A \mid -A \mid \mathbf{b})$ is totally unimodular, the same arguments as in the other case show that $\mathbf{v} \in \{-1, 0, 1\}^{2d}$, and the statement follows from the non-negativity constraint of (LP). □

Using this lemma, we can now prove Theorem 4.11.

Proof of Theorem 4.11. By construction, the system $A\mathbf{y} = \mathbf{b}$ has a solution. For $i = 0, 1$, let α_i be the module element corresponding to an ℓ_i-minimal solution of the system. Note that, again by construction, $\alpha_i \in R_i(f, \sigma)$. By Lemma 4.12, α_i contains only nonzero coefficients ±1, which implies that $\|\alpha_i\|_0 = \|\alpha_i\|_1$ for $i = 0, 1$, and the result follows. □

In most of our applications, all polynomials involved are of the form $a - b$ with $a, b \in \langle X \rangle \cup \{0\}$ encoding identities of operators of the form $A = B$. Such polynomials are called *pure difference binomials*. The following corollary of Theorem 4.11 ensures that Algorithm 3 computes a sparsest representation up to signature σ provided that the input polynomials are pure difference binomials.

Corollary 4.13. *Let $f, f_1, \ldots, f_r \in \mathbb{Q}\langle X \rangle$ be pure difference binomials, $\sigma \in M(\Sigma)$ and $\alpha \in R(f, \sigma)$. Given these elements as input, the output of Algorithm 3 is an element in $R_0(f, \sigma)$.*

Proof. Let A, $\mathbf{b}$ as constructed in Algorithm 3. By assumption on $f, f_1, \ldots, f_r$, each column of $(A \mid \mathbf{b})$ contains at most one entry $+1$ and at most one entry -1 with all other entries being 0. Each square submatrix U of $(A \mid \mathbf{b})$ either contains a zero column (then U is singular), a column with one nonzero entry (then expansion of $\det(U)$ along this column yields inductively $\det(U) \in \{-1, 0, 1\}$), or each column of U contains exactly one entry $+1$ and one entry -1 (then $\mathbf{1}^T U = 0$ showing that U is singular). Thus, $(A \mid \mathbf{b})$ is totally unimodular and the result follows from Theorem 4.11. □

Example 4.14. We revisit Example 3.13. All polynomials that appear in this example are pure difference binomials. Hence, Corollary 4.13 implies that Algorithm 3 yields a sparsest cofactor representation up to the used signature bound σ. In particular, if a degree-compatible module ordering is used and σ is chosen so that $\deg(\sigma) > 7$, then by Corollary 3.8 the computed representation is a sparsest one (independent of any bound).

Applying Algorithm 3 to Example 3.13, with the cofactor representation given in (1) and a suitable signature bound σ, yields again (1), showing that this is a sparsest cofactor representation. The basis used to form the linear system only consists of 232 elements, compared to the 17 000 that Algorithm 1 would need.

5 EXPERIMENTAL RESULTS

We have written an implementation of Algorithm 3 for SageMath[1] using our package SignatureGB[1] for the signature-based computations and the IBM ILOG CPLEX optimization studio [16] for the linear programming.

In Table 1, we compare the weight of cofactor representations computed by Algorithm 3 to those found by other approaches. In particular, we compare our algorithm to tracing standard Gröbner basis computations and reductions, and to tracing reductions done with a signature Gröbner basis.

As benchmark examples we encode the following statements about Moore-Penrose inverses in terms of noncommutative ideal membership.

- `SVD` encodes [15, Ch. 5.7 Fact 4]. This statement provides a formula for the Moore-Penrose inverse of a matrix in terms of the matrix's singular value decomposition.
- `ROL` encodes the implication (2) ⇒ (1) in [18, Thm. 3]. This statement provides a sufficient condition for the identity $(AB)^\dagger = B^\dagger A^\dagger$ to hold, where $X^\dagger$ is the Moore-Penrose inverse of an element in a ring with involution.
- `ROL-`n encodes the implication $(n) \Rightarrow (1)$ in [7, Thm. 2.1]. This family provides several sufficient conditions for the identity $(AB)^\dagger = B^\dagger A^\dagger$ to hold, where $X^\dagger$ is the Moore-Penrose inverse of bounded operators on Hilbert spaces.
- `Hartwig-`n encodes the implication $(n) \Rightarrow (1)$ in [5, Thm. 2.3]. This family provides several sufficient conditions for the identity $(ABC)^\dagger = C^\dagger B^\dagger A^\dagger$ to hold, where $X^\dagger$ is the Moore-Penrose inverse of an element in a ring with involution.

For all examples, $\leq_{\text{deglex}}$ is used in combination with the degree-compatible ordering $\leq_{\text{DoPoT}}$ for the signature-based computations.

The first columns of Table 1 contain information about the ideals that arise when translating the operator statements. In particular, we list the number of generators of each ideal and their maximal degree. Moreover, in the column for Algorithm 3, we provide information on the used signature bound. A value n in this column indicates that we consider only cofactor representations of degree $< n$. The degree bounds were chosen so that the computation would finish for the smaller examples `SVD`, `ROL`, `ROL-`n within a few minutes and for the larger examples `Hartwig-`n within a few hours on a regular laptop. We note that these degree bounds are strictly smaller than those that Corollary 3.8 yields, but the latter were computationally infeasible. Nevertheless, Table 1 shows that Algorithm 3 still allows to find sparser representations. Also, all benchmark examples only consist of pure difference binomials. Hence, Corollary 4.13 implies that the representations computed by Algorithm 3 are the sparsest up to the respective degree bounds.

Table 1 shows that the representations found by standard Gröbner bases are often significantly longer than necessary, and that the representations computed by signature Gröbner bases are in many cases already a lot sparser and within 15 – 30% of the optimum. Only in two examples (`ROL-2` and `ROL-5`) is the standard Gröbner basis representation sparser than the signature-based representation. However, for none of the examples is any of the (signature) Gröbner basis representations optimal.

We also tested an adapted version of Algorithm 3 as described in Remark 4.9 that minimizes the total number of symbols appearing in a cofactor representation. For most benchmark examples, the thereby computed representations have the same (minimal) weight as those found with the standard version of the algorithm, but the total number of symbols decreases by up to 15%. Only for `ROL-3` does the weight increase by one, while the number of symbols decreases from 196 to 172.

In the last columns of Table 1, we compare the size of the matrix A constructed in Algorithm 3 with and without applying the pruning techniques discussed in Section 4.2. We also list the ratio between the number of nonzero entries in the pruned matrix and the number of nonzero entries in the original matrix. As the table shows, in some examples the size of the resulting linear system can be reduced drastically, cutting the number of nonzero entries almost in half.

Acknowledgement. We thank G. Regensburger for valuable remarks.

[1] Available at https://clemenshofstadler.com/software/

REFERENCES

[1] E. J. Candes and T. Tao. Decoding by Linear Programming. *IEEE Trans. Inform. Theory*, 51(12):4203–4215, 2005.
[2] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic Decomposition by Basis Pursuit. *SIAM Rev.*, 43(1):129–159, 2001.
[3] C. Chenavier, C. Hofstadler, C. G. Raab, and G. Regensburger. Compatible rewriting of noncommutative polynomials for proving operator identities. In *Proc. ISSAC '20*, pages 83–90, 2020.

Example	#gens	deg	GB	SigGB	Algo. 3 (bound)	w/o pruning	w/ pruning	ratio ≠ 0
SVD	32	3	51	39	25 (10)	127 k × 397 k	118 k × 328 k	0.83
ROL	28	5	80	39	30 (12)	22 k × 101 k	22 k × 56 k	0.55
ROL-2	28	5	20	21	15 (12)	24 k × 107 k	23 k × 60 k	0.56
ROL-3	28	5	49	44	31 (12)	19 k × 87 k	18 k × 46 k	0.53
ROL-4	28	5	59	46	33 (12)	68 k × 236 k	64 k × 137 k	0.58
ROL-5	28	5	28	30	22 (12)	33 k × 134 k	31 k × 80 k	0.60
ROL-6	28	5	39	39	30 (12)	22 k × 99 k	21 k × 55 k	0.56
ROL-7	40	9	85	23	17 (12)	18 k × 86 k	17 k × 46 k	0.54
ROL-8	44	7	241	19	17 (12)	258 k × 962 k	249 k × 560 k	0.58
Hartwig-4	23	15	316	54	46 (18)	353 k × 1 739 k	349 k × 1 460 k	0.84
Hartwig-5	26	15	99	43	35 (17)	407 k × 1 642 k	398 k × 1 374 k	0.84
Hartwig-6	24	15	86	33	29 (17)	218 k × 958 k	217 k × 808 k	0.84

Table 1: Comparison of weights of cofactor representations computed by standard Gröbner bases (`GB`), by signature Gröbner bases (`SigGB`), and by Algorithm 3 (`Algo. 3`). Also, size comparison of the coefficient matrix A (rounded to thousands) in Algorithm 3 with and without applying the pruning techniques from Sec. 4.2.

[4] R. R. Coifman and M. V. Wickerhauser. Entropy-Based Algorithms for Best Basis Selection. *IEEE Trans. Inform. Theory*, 38(2):713–718, 1992.

[5] D. S. Cvetković-Ilić, C. Hofstadler, J. Hossein Poor, J. Milošević, C. G. Raab, and G. Regensburger. Algebraic proof methods for identities of matrices and operators: improvements of Hartwig's triple reverse order law. *Appl. Math. Comput.*, 409:126357, 2021.

[6] G. B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activ. Anal. Proc. Alloc.*, 13:339–347, 1951.

[7] D. S. Djordjević and N. Č. Dinčić. Reverse order law for the Moore-Penrose inverse. *J. Math. Anal. Appl.*, 361(1):252–261, 2010.

[8] D. L. Donoho. Compressed Sensing. *IEEE Trans. Inform. Theory*, 52(4):1289–1306, 2006.

[9] C. Eder and J.-C. Faugère. A survey on signature-based algorithms for computing Gröbner bases. *J. Symbolic Comput.*, 80:719–784, 2017.

[10] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F4). *J. Pure Appl. Algebra*, 139(1-3):61–88, 1999.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[12] J. W. Helton and J. J. Wavrik. Rules for computer simplification of the formulas in operator model theory and linear systems. In *Nonselfadjoint operators and related topics*, pages 325–354. Springer, 1994.

[13] C. Hofstadler, C. G. Raab, and G. Regensburger. Certifying operator identities via noncommutative Gröbner bases. *ACM Commun. Comput. Algebra*, 53(2):49–52, 2019.

[14] C. Hofstadler and T. Verron. Signature Gröbner bases, bases of syzygies and cofactor reconstruction in the free algebra. *J. Symbolic Comput.*, 113:211–241, 2022.

[15] L. Hogben. *Handbook of linear algebra*. CRC press, 2013.

[16] IBM ILOG CPLEX Optimization Studio. *Version*, 22.1(1987–2023), 2023.

[17] M. Kinyon. Proof simplification and automated theorem proving. *Philos. Trans. Roy. Soc. A*, 377(2140):20180034, 2019.

[18] J. J. Koliha, D. Djordjević, and D. Cvetković. Moore-Penrose inverse in rings with involution. *Linear Algebra Appl.*, 426(2-3):371–381, 2007.

[19] S. G. Mallat and Z. Zhang. Matching Pursuits With Time-Frequency Dictionaries. *IEEE Trans. on Signal Process.*, 41(12):3397–3415, 1993.

[20] E. Mayr. Membership in Polynomial Ideals over Q Is Exponential Space Complete. In *STACS 89: 6th Annual Symposium on Theoretical Aspects of Computer Science Paderborn*, pages 400–406. Springer, 1989.

[21] E. W. Mayr and A. R. Meyer. The Complexity of the Word Problems for Commutative Semigroups and Polynomial Ideals. *Adv. in Math.*, 46(3):305–329, 1982.

[22] E. W. Mayr and S. Toman. *Complexity of Membership Problems of Different Types of Polynomial Ideals*, pages 481–493. Springer International Publishing, 2017.

[23] T. Mora. *Solving Polynomial Equation Systems IV: Volume 4, Buchberger Theory and Beyond*, volume 158. Cambridge University Press, 2016.

[24] F. A. Potra and S. J. Wright. Interior-point methods. *J. Comput. Appl. Math.*, 124(1-2):281–302, 2000.

[25] C. G. Raab, G. Regensburger, and J. Hossein Poor. Formal proofs of operator identities by a single formal computation. *J. Pure Appl. Algebra*, 225(5):106564, 2021.

[26] L. Schmitz and V. Levandovskyy. Formally Verifying Proofs for Algebraic Identities of Matrices. In *International Conference on Intelligent Computer Mathematics*, pages 222–236. Springer, 2020.

[27] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[28] R. Veroff. Finding Shortest Proofs: An Application of Linked Inference Rules. *J. Automat. Reason.*, 27(2):123–139, 2001.

[29] X. Xiu. *Non-Commutative Gröbner Bases and Applications*. PhD thesis, University of Passau, Germany, 2012. Available at http://www.opus-bayern.de/uni-passau/volltexte/2012/2682/.