On lexicographic Gröbner bases whose primary components are triangular sets

Xavier Dahan

Ochanomizu University, 2-1-1 Otsuka, Bunkyo-ku. Tokyo, Japan 112-8610

Abstract

Let $I \subset k[x_1, \ldots, x_n]$ be a polynomial ideal of dimension zero. Assume that each of its primary components has a minimal lexicographic Gröbner basis (lexGb) in a purely triangular form, that is is a triangular set. This article presents:

1) An algorithm that finds the leading monomials of minimal lexGbs of I (or, it equivalently can find the standard monomials).

2) a Chinese Remainder-like reconstruction algorithm that computes a minimal but not reduced in general, lexGb of I from that of its primary components (no linear algebra, nor Gröbner basis computation are used).

3) a factorization pattern of the polynomials in minimal lexGb's of I, to as "Generalized Lazard's theorem".

4) a proof that the conservation property of the Gröbner basis (sometimes coined "stability") hold under specialization maps in this context (sometimes referred to as "Generalized Gianni's theorem")

These results generalize and improve in several directions previous works, most of which are under the assumption that primary ideals are ideals of points. The complexity analysis of 2) is new even in that case. The proposed framework not only allows to treat all four aspects above uniformly with a complexity analysis, but is simple enough to implement easily. One is proposed in Maple.

Keywords: Groebner basis, Lexicographic order, Primary ideals, Idempotent, Triangular set, Henselian ring

1. Introduction

Notations and preliminaries. All along this article I denotes an ideal of dimension zero of a polynomial ring $R := k[x_1, \ldots, x_n]$ over a field k of characteristic 0 or larger than the degree $D := D(I) := \dim_k R/I$. All Gröbner bases, normal form computations are with respect to the lexicographic monomial order denoted $\prec_{lex}$ where $x_1 \prec_{lex} x_2 \prec \cdots \prec_{lex} x_n$. The notation "mod $\langle g_1, \ldots, g_s \rangle$" is the same as "normal form" in the sense of Gröbner basis, that is $\mathrm{NF}(\,.\,, [g_1, \ldots, g_s])$. The semi-order on the monomials induced by the divisibility relation is denoted $\leq_{mon}$. By an abuse of notation it will also refer to the semi-order on the n-uplets seen as exponents of monomials.

Email address: `xdahan@gmail.com` (Xavier Dahan)

Given a subset $\mathcal{F}$ of a Cartesian product E^n, the notation $\mathcal{F}_{\leq \ell} \subset E^\ell$ refers to the projection on the first ℓ coordinates of elements in $\mathcal{F}$. Then a subset $A \subset \mathcal{F}$ is said to *extend* an element $a \in \mathcal{F}_{\leq \ell}$ if $A_{\leq \ell} = \{a\}$ (if $A = \{a'\}$ is a single element, we may simply write a' *extends* a). For example, with $E = \mathbb{Z}_{\geq 0}$, $\mathcal{F} = \mathbb{Z}^2_{\geq 0}$, $A = \{(1,2),(1,3),(1,0)\}$, we remark that A extends $a = (1) \in \mathcal{F}_{\leq 1} = \mathbb{Z}_{\geq 0}$. Let $a' = (1,2) \in A$ and $A \supset A' = \{(1,2)\} = \{a'\}$; then A' or simply a' extends a.

It will be convenient to adopt a similar notation for polynomials, although the meaning is not exactly the same as for sets: as above let $R = k[x_1, \ldots, x_n]$ and $\mathcal{F} \subset R$ be a subset of polynomials. Denote $\mathcal{F}_{\leq \ell} := \mathcal{F} \cap k[x_1, \ldots, x_\ell] \subset \mathcal{F}$. In particular $R_{\leq \ell} := k[x_1, \ldots, x_\ell]$.

A family of triangular sets (see definition Eq (1) below) $\mathcal{A} = \{\mathbf{t}^{(i)}\}$, with $\mathbf{t}^{(i)} = (t_1^{(i)}(x_1), \ldots, t_n^{(i)}(x_1, \ldots, x_n))$ pairwise distinct is said to *extend* a triangular set $\boldsymbol{\sigma} = (s_1(x_1), \ldots, s_\ell(x_1, \ldots, x_\ell)) \in R_{\leq \ell} = k[x_1, \ldots, x_\ell]$, if for all i, $\mathbf{t}^{(i)}_{\leq \ell} = (t_1^{(i)}(x_1), \ldots, t_\ell^{(i)}(x_1, \ldots, x_\ell)) = \boldsymbol{\sigma}$.

Given an algebraic extension K of k, the set of associated primes of $I \subset K[x_1, \ldots, x_n]$ is written $Ass_K(I)$. Since I is zero dimensional, $Ass_{\bar{k}}(I)$ consists of *ideals of points*, which are: $\langle x_1 - \rho_1, \ldots, x_n - \rho_n \rangle := \mathfrak{m}_\rho$ for a point $\rho := (\rho_1, \ldots, \rho_n) \in \bar{k}^n$. This set of points is written $V(I)$; it is the set of common zeros in $\bar{k}^n$ of the polynomials in I. The number of points in $V(I)$ is written D. In general $D \leq d(I)$ with equality if and only if I is radical (degenerate situations which may occur in non-separable extensions in characteristic > 0 do not hold here since we have assumed that $\mathrm{char}(k) > d(I)$).

The primary decomposition (unique) over the base field k is written $I = \prod_{i=1}^t \mathfrak{q}_i$, where the associated prime to $\mathfrak{q}_i$ is denoted $\mathfrak{m}_i$. This work assumes that the ideal I verifies:

(H) each primary ideal $\mathfrak{q}_i$ possesses a lexGb that is a triangular set $\mathbf{t}^{(i)}$, as defined in (1) below.

The terminology of *triangular set* comes from the Wu-Ritt (Lazard, Kalkbrener, Wang, Moreno-Maza etc.) method to decompose polynomial systems. In this article this refers to a lexGb that is also a regular sequence, with as many polynomials of variables; hence is of dimension zero, and has the following shape:

$$\mathbf{t}\left\{\begin{array}{rcl} t_n(x_1, x_2, \ldots, x_{n-1}, x_n) & = & x_n^{d_n} + \cdots \\ t_{n-1}(x_1, \ldots, x_{n-1}) & = & x_{n-1}^{d_{n-1}} + \cdots \\ \ddots & & \vdots \\ t_1(x_1) & = & x_1^{d_1} + \cdots \end{array}\right. \tag{1}$$

Note that general primary ideals have a lexGb that is not necessarily triangular (see Prop. 1).

Contributions. Under the assumption **(H)**, this article presents the following four related contributions:

Contribution 1 (Corollary 3). *The input data of the lexGb's of the primary ideals are arranged in a* tree $\mathcal{T}$ *with one polynomial at each node.*

To each node of this tree, except the leaves, Algorithm 1 (Section 2) assigns an exponent tree *constructed by using only comparison tests of nonnegative integers smaller than the degree of the ideal $< D(I)$.*

The exponent trees at nodes at depth ℓ in the primdec tree $\mathcal{T}$ encodes the monomials at the $\ell+1$-border monomials of I (see Definition 3), which contains the minimal exponents of the lexGb in $k[x_1, \ldots, x_{\ell+1}] \setminus k[x_1, \ldots, x_\ell]$.

Contribution 2 (4). *Given the input data arranged into trees and exponent trees, and a minimal exponent $(\sigma_1, \ldots, \sigma_n)$ as Contribution 1) allows to find, Algorithm 4 computes an element of $\mathscr{G}(I)$ of leading monomial $x_1^{\sigma_1} \cdots x_n^{\sigma_n}$.*

From the recursive nature of the algorithms, it is easy to unveil the following structure theorem, sometimes referred to as the "generalized" Lazard's theorem in the literature.

Contribution 3 (Theorem 2). *If $g \in \mathscr{G}$ not in $\mathscr{G}_{\leq n-1}$,*

$$\operatorname{LM}(g) = x_1^{i_1} \cdots x_n^{i_n} \Rightarrow g \equiv \chi_1 \cdots \chi_n \bmod \langle \mathscr{G}_{\leq n-1} \rangle, \tag{2}$$

for some polynomials $\chi_1, \ldots, \chi_n$ verifying $\chi_j \in k[x_1, \ldots, x_j]$ and $\operatorname{LM}(\chi_j) = x_j^{i_j}$.

The fourth contribution below is a generalization of Gianni-Kalkbrener's specialization theorem. To state this result, we first restate [1, Theorem 3.1] below. Let G be a Gröbner basis (not necessarily a lexGb) of I, for an elimination order on the variables $x_1, \ldots, x_\ell \ll x_{\ell+1}, \ldots, x_n$, write $\operatorname{LC}_\ell(f) \in k[x_1, \ldots, x_\ell]$ the leading coefficient of f seen as a polynomial in $k[x_1, \ldots, x_\ell][x_{\ell+1}, \ldots, x_n]$. A specialization map ϕ_a at a point $a \in \bar{k}^\ell$, splits G into two disjoint sets: $G_a^0 := \{g \in G \mid \phi_a(\operatorname{LC}_\ell(g)) = 0\}$, and $G_a := G \setminus G_a^0$. Then, $\operatorname{LM}(\phi_a(I)) = \phi_a(\operatorname{LM}_\ell(I))$ holds if and only if, for all $g \in G$:

$$\operatorname{NF}(\phi_a(g),\ \phi_a(G_a \setminus \{g\})\ ) = 0 \Leftrightarrow \phi_a(\operatorname{LC}_\ell(g)) = 0. \tag{3}$$

Contribution 4 (Corollary 2). *The specialization property* (3) *holds for lexGb of ideals I verifying Assumption* **(H)**.

1.1. Previous work

Remark about Assumption **(H)**. Thus Assumption **(H)**does not fully cover the setting of [2]: the $\langle x_1 - \alpha_, x_2 - \alpha_2 \rangle$-primary ideal $\langle (x_1 - \alpha)^2, (x_1 - \alpha_1)(x_2 - \alpha_2), (x_2 - \alpha_2)^2 \rangle$ does not verify Assumption **(H)**. However $\langle x_1 - \alpha_, x_2 - \alpha_2 \rangle$-primary ideals such as $\langle (x_1 - \alpha_1)^2, (x_2 - \alpha_2)^2 + 2(x_2 - \alpha_2)(x_1 - \alpha_1) + 3(x_1 - \alpha_1) \rangle$ which falls into Assumption **(H)** are not covered in [2]. For primary triangular ideals more general than monomial ones, or translations of thereof, Contribution 1 is new.

These are given by the intersection of cancellation identities of higher differential operators of polynomials evaluated at a point. Like the work [2] it includes cases not covered by Assumption **(H)**: the $\langle x_1, x_2 \rangle$-primary ideal: $(g_1(x_1) = x_1^2, g_{2,1}(x_1, x_2) = x_1 x_2, g_{2,2}(x_1, x_2) = x_2^2$, is associated the two linear operators: $\frac{\partial .}{\partial x_1} |_{0,0}$ and $\frac{\partial .}{\partial x_2} |_{0,0}$.

Example 1. Consider the triangular set: $t_1(x) := x^2,\ t_2(x, y) := y^2 + y(2x) - x$. Let $I := \langle t_1, t_2 \rangle \subset \mathbb{Q}[x, y]$ for $x < y$, be the primary ideal of radical the maximal ideal $\langle x, y \rangle$.

Then $\frac{\partial t_1}{\partial x}(0, 0) = 0$ while $\frac{\partial t_2}{\partial x}(0, 0) = 2y - 1|_{0,0} = -1$. This contradicts the fact that polynomials in I can be determined by Hermite conditions.

About contribution 1). Given an input polynomial system and a monomial order, computing the minimal exponents of a minimal Gröbner basis is believed in general to be as hard as computing the Gröbner basis itself. In general, there is no other way than computing a Gröbner basis. Now given the Gröbner basis of the primary components of an ideal I, the problem can be easier. For the lexicographic order, Contribution 1) affirms that computing these minimal exponents require no arithmetic operations, at least when I is zero-dimensional and verifies Assumption **(H)**.

An early instance of this fact (see Introduction of [3] for more historical details) is due to Cerlienco-Mureddu [2]. The main result is a combinatorial algorithm, based on Ferrer diagrams, which outputs the standard monomials from the data of points. Minimal exponents can be thus deduced from the standard monomials. Besides the case of ideals of points [2, § 3] mentions the case of shifted monomial ideals; it sketches the possibility set up a similar combinatorial algorithm, but left unproved and accompanied by the comment "it is possible *(but by no mean trivial)* to prove that. . . is a linear basis of $K[X]/\mathcal{I}(P)$ which is minimal" (p.82, end of § 3 in [4]).

The two works of Marinari-Mora [3, 5] are somewhat affiliated to the Cerlienco-Mureddu approach, so are mentioned here, even though their core claimed results are more related to Contributions 3) & 4). For shifted monomial ideals (called "CeMu-ideal" therein) the unproved in [2] result is taken for granted. Moreover it is claimed in [5, Section 5] an algorithm that generalizes Cerlienco-Mureddu's one to *any* zero-dimensional ideals. But this Section 5 is just a sketch, without any proof. Most new claimed results in [5, Sections 6-7] build up crucially on these purpotted generalizations.

The lex game [6] treats of ideals of points only, but provides a fresh viewpoint to the problem. While necessitating no arithmetic operations, its complexity analysis in terms of number of equality tests have been analyzed (see Theorem 14 yielding $O(Dnr)$ where r is the maximal degree of the "point trie", similar to our primdec tree defined in Section 2,) and further improved by in [7, Thm. 4.2]. This is better than a straightforward implementation of Cerlienco-Mureddu's algorithm, running in $O(D^2n^2)$ (see [6, p.62]). Our complexity claim in Contribution 1) is a direct and simple generalization of the results in [6, 7] from "point trie" to the primdec tree.

The article of Lederer [8] does not focus on the problem of finding standard monomials, but implicitly an algorithm can be deduced easily. The approach is based on an operation on standard monomials called *four-in-a-row* in reference to the the famous board game where players must align four pieces of the same color.

Our proposal encompasses this *four-in-a-row* without reference to an operation on standard monomials, but translates the problem to a tree data structure amenable to implementation and complexity analysis. This point of view is inspired by the *lex trie* introduced in [6] (see also [7, Ex. 5.12]) but with some subtleties; our aim is indeed to compute directly the minimal exponents, and not the standard monomials. It happens that what the exponent trees compute are the ℓ-border monomials (see Def. 3) for $\ell = 1, \ldots, n$. This sum is always smaller than the number of standard monomials. In the process of computing the exponent trees, it also naturally gathers data necessary to carry out the computation of the lexGb itself in Algorithm 4. Thus, this variation of the lex trie is significant.

As for implementations, the lex game algorithm of [6] is implemented in Singular, and we are not aware of any implementation related to the works [8, 2, 3, 5].

The recent paper [9] reports on a generalization of the *four-in-a-row* operation to points with a "multiplicity structure" (equivalent to a shifted monomial ideals). The article has several drawbacks. First there is the misconception that any primary ideal is as defined in the paper, hence is monomial. In Lemma 1 therein, the claim is indeed made for *any* ideal I, but the article treats only shifted monomial ideals. Hence the proof is not correct. Similarly, Section 7 uses this Lemma 1 to decompose along the smallest variable a polynomial in the lexGb. Even under this strong assumption, it must furthermore be restricted to shifted monomial ideals only, and not to any ideal as written therein.

Another feature that appears to be a drawback is that it attempts to generalize to the geometric *four-in-a-row* operation of Lederer on standard monomials. The presentation is sometimes very hard to follow, and the algorihtms proposed ressemble more sketches than ready to implement routines. The managmenent of data appears indeed crucial both for the clarity of the proofs and the presentaion of the algorithms. The geometric viewpoint of the *four-in-a-row* operation is probably not the most suitable, although it is visually appealing. Besides, the authors themselves acknowledge of this (page 277, third paragraph):

> ...and a full complexity analysis would be infeasible. Our method may not be particularly efficient....

About contribution 2). This can largely be viewed as interpolation, or the recombination part of the isomorphism in the Chinese Remainder theorem. The earliest, let us say "post-Gröbner", instance of reconstructing a Gröbner basis from points dates back to the famous Buchberger-Möller algorithm (often written BM-algorithm). For related earlier works, consult *e.g.* the introduction of Marinari-Mora's [3]. This family of BM-algorithms is far more general in that it deals with *any* monomial order, not only the lexicographic one. It computes a "normal form" matrix for the target monomial order, and is very comparable to the famous FGLM matrix. However, it does not allow to find the minimal exponents without arithmetic operations as in this work, nor does it unveil the structure (Contributions 1), 2) and 3) here). The generalization from ideals of points to primary ideals was undertaken in [10]. Algorirhtm GBM therein has some limitations, but since under our setting the inputs are Gröbner bases, they are endowed with the required "normal form vector maps". The algebraic complexity is therefore $O(d(I)^2(\ell_{\mathscr{G}} + d(I)) + d(I)^2n^2)$, where ℓ_G is the number of elements in a minimal Gröbner basis. Remark that the authors of [10] also present a modular version of Agloriothm GBM, wich they also analysze in term of bit-complexity. Main Algorithm 4 can also be made modular, but we will restrict to the algebraic complexity for now; the analysis is already enough complicated.

In a slightly different direction, Marinari-Mora [3] have exploited the earlier combinatorial approach on monomials of Cerlienco-Mureddu [2], combined with BM-like linear algebra, to deduce an interpolation algorithm; the main purpose of this work is to compute the factorization pattern of Contribution 3) though. In term of method and complexity, it is thus affiliated with the BM-algorithm. (again the generelization to shifted monomial ideals in [5] relies on the largely unproved Cerlienco-Mureddu [2, § 3] correspondence in this case).

The present work is not affiliated with the BM-algorithms but to [11, 8, 3, 6, 5, 9, 4, 2], for which we provide hereunder a comparison. The lex game [6] implicitly, and especially Lederer [8], Gao-Stroomer-Rodriguez [11] provides explicit algorithms for constructing the Gröbner basis in the case of ideal of points. None provides a complexity analysis, and only [11] has been implemented. Only the construction of Lederer [8] computes the reduced lexGb without additional normal form computations. It does complicate the description of the algorithm though, which indeed has not been implemented, and is not amenable to complexity analysis; yet alone the attempt of generalization [9], as have the pointed out drawbacks suggested.

We have concluded that the above approach of Cerlienco-Mureddu pursued by Marinari-Mora, and the *four-in-a-row* approach of Lederer, seem less prone to generalizations/improvements than the lex game [6]. The primdec tree introduced in [12] and used here coincide with the "point trie" introduced in [6, § 3] in case of ideal of points. The procedure that fills these trees, Algorithm 1, is reminiscent of thow the "lex tries" of the lex game are filled [7, Ex. 5.12].

About contribution 3). Lazard's structural theorem [13] is certainly the starting point of the study of structure of lexGb's. It is constrained to polynomials in two variables, but without assumption on the ideal I. A generalization is proposed by Marinari-Mora [3, Theorem 11.4 (M)] to zero-dimensional radical ideals. Assuming the unproved result of Cerlienco-Mureddu concerning ideals given by Hermite conditions, they show in [5, bottom of p.2] that the factorization property hold also in this case. We recall that in both cases a necessary interpolation part is not fully presented, but shown to exist. In the present work, the factorization patterns is a simple by-product of the recursive nature of the recombination Algorithm 4.

About contribution 4). The first "post-Gröbner" instance of a specialization result like Contribution 4) is due to the famous Gianni's paper [14], also known as Gianni-Kalkbrener theorem but Gianni's paper is more complete. It says that the theorem that follows Eq. (3) holds whenever I is zero-dimensional and all variables but the largest one are specialized, that is when $\ell = n-1$. Becker [15] proves that the specialization property for any ℓ under the additional assumption that I must be radical. Kalkbrener [1] has provided necessary and sufficient conditions for the Gröbner property to be preserved by specialization.

In the non-radical case, in [5] is reported that "at least in the radical ideal case, this combinatorial description subsumes Gianni-Kalkbrener Theorem as a corollary and gives a combinatorial justification of their algorithm." This refers explicitly to conservation of the Gröbner property under specialization. But from the factorization pattern written therein, namely,

$$f_i = \prod_m \prod_\delta \gamma_{m\delta i} \bmod \langle f_1, \dots, f_{i-1} \rangle$$

it appears not obvious, and no proof is given. The one given here is indeed a corollary of the factorization pattern (Corollary 2) but it relies crucially on the more precise factorization

$$f_i = \prod_m \prod_\delta \gamma_{m\delta i} \bmod I_{\leq n-1}.$$

1.2. Organization of the paper

Section 2 sets the input data: the lexGb's of the primary ideals $\mathfrak{q}_i$ (which are triangular sets according to **(H)**) into a labeled tree, coined "prim-dec tree" in [12] generalizing the "poin tree" of [6]. Later algorithms will deal directly with this tree. To each node N of this tree, not a leaf, is associated in Section 3 another tree $\mathcal{E}_I(N)$, called "exponent tree", which encodes the leading exponents of some polynomials in I. Two other related trees, the "discard" and "interpolate" trees denoted $\mathcal{R}_I(N)$ and $\mathcal{N}_I(N)$ are also introduced. They have the same shape and carry information used in the "recombination" (generalizing interpolation) algorithm 4 "`computePoly`" in Section 4. These exponent trees are trivial (made of one branch) in the case of ideal of points, but complications occur otherwise that we have convenient to carefully record in trees. Section 4 presents the algorithm "`computePoly`" that constructs some polynomials of leading exponent prescribed by the exponent trees $E(N)$, and are shown to belong to I. Section 5 proves that they form indeed polynomials in the lexGb. The last section 6 is made of comments: what happens when Assumption **(H)** is lifted ? Can we compute the reduced Gröbner basis without *a posteriori* normal forms ? and so on.

2. The primary decomposition tree

We recall the construction of the prim-dec tree explicitly introduced in [6] (but already implicitly used in [16] to define the equiprojectable decomposition) and generalized in [12] to primary ideals which are triangular sets. This requires only equality testing on coefficients of the input triangular sets $\mathtt{t}^{(\alpha)}$ and comparisons in $\mathbb{Z}_{\geq 0}$. Its construction involves no arithmetic operations hence is cheap compared to the computation of the polynomials.

Then in § 3 to each parent of leaves (called *preleaf* thereafter) N of this labeled tree, an algorithm that assigns an "exponent tree" $\mathcal{E}(N)$, in which paths from the root to the leaves encodes n-uplet of non-negative integers. These will happen to be leading exponents of some polynomials in I (see section 4), who form a Gröbner basis of I (see section 5).

Tree's terminology. We recall some common definitions and wordings in trees. Everything is standard, intuitive and presents no difficulty.

(root) The *root* is a special determined node.

(path) A *path* of length $\ell - 1$ is a sequence of successively connected vertices $(v_1, \dots, v_\ell)$ such that $v_{i+1} \neq v_{i-1}$ (a.k.a walk without backtracking).

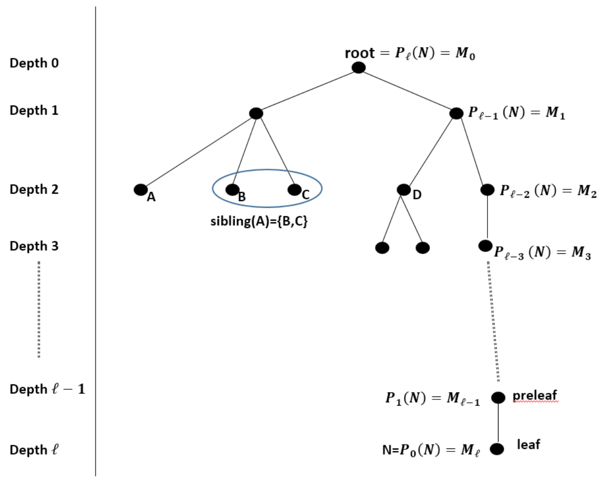
(depth) A node N is said to be at depth ℓ iff there is a path of length ℓ connecting root to N. The path is then unique (because there is no cycle in a tree by definition). Write it $(M_0, M_1, \dots, M_{\ell-1}, M_\ell)$, with $M_\ell = N$ and $M_0 = \mathsf{root}$. Note that the depth of M_i is i, in particular the depth of root is zero.

(parent) If $\ell > 0$, the *parent* of N is the unique node $M_{\ell-1}$. The root has no parent. We write $P_1(N) = M_{\ell-1}$.

(ancestor) More generally if $k \le \ell$, the k-th ancestor of N is the node $M_{\ell-k}$. It is written $P_k(N) = M_{\ell-k}$, hence the 1-st ancestor is the parent. By convention the 0-th ancestor of N is N itself denoted $P_0(N) = N$. A node M is an ancestor of N iff there is a $k \le \ell$ such that $M_k = M$. Note that the root is an ancestor of all nodes, including itself.

(descendant) A node N is a *descendant* of a node M if M is an ancestor of N.

(child, leaf) If N has other neighbors than its parent, then these nodes are called the *children* of N, denoted $\mathrm{Child}(N)$. Else N has no child and is a *leaf*.

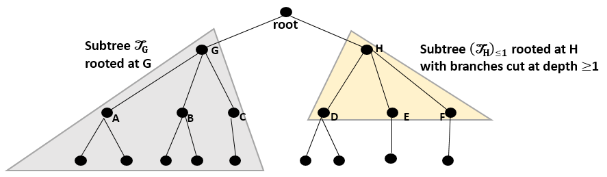


(preleaf) In this work parents of leaves will play an important role, they will be called *preleaves*. Moreover, all leaves will be at the same depth, which is the *depth of the tree*.

(sibling) Given a node N not the root, sibling(N) denotes the set Child$(P_1(N)) \setminus \{N\}$.

(branches cut) Given a tree $\mathcal{T}$ of depth n, and $0 \leq \ell \leq n$, $\mathcal{T}_{\leq \ell}$ is the tree where all nodes at depth $> \ell$ and edges connecting them are deleted. In particular, $\mathcal{T}_{\leq 0} = [\mathsf{root}]$, $\mathcal{T}_{\leq n} = \mathcal{T}$.

(rooted subtree) Given a node N of a $\mathcal{T}$, the *subtree rooted at* N denotes the tree $\mathcal{T}_N$ whose root is N, and descendants are the descendants of N in $\mathcal{T}$. If N is at depth ℓ in $\mathcal{T}$ and M is a descendant of N at depth $k \geq \ell$ in $\mathcal{T}$, then M is at depth $\ell - k$ in $\mathcal{T}_N$. In particular the (depth of $\mathcal{T}_N$) is equal to (the depth of $\mathcal{T}$) $-\ell$.


root
Subtree $\mathcal{T}_G$ rooted at G
G
A
B
C
H
Subtree $(\mathcal{T}_H)_{\leq 1}$ rooted at H with branches cut at depth ≥ 1
D
E
F

2.1. Gröbner basis of a primary ideal

An ideal of dimension zero I admits a unique minimal primary decomposition, $I = \mathfrak{q}_1 \ldots \mathfrak{q}_t$. The radical decomposition (also unique) is then $\sqrt{I} = \mathfrak{m}_1 \cdots \mathfrak{m}_t$ where $\mathfrak{m}_i = \sqrt{\mathfrak{q}_i}$. The lemma below gathers basic well-known facts about lexGb.

Lemma 1. *Let $Ass(I) := \{\mathfrak{m}_i,\ i = 1, \ldots, t\}$. The lexGb of $\mathfrak{m}_i$ is a triangular set $(p_1^{(i)}, \ldots, p_n^{(i)})$, where $p_{j+1}^{(i)}$ is an irreducible polynomial over the field $K_j := k[x_1, \ldots, x_j]/\langle p_1^{(i)}, \ldots, p_j^{(i)} \rangle$.*

$Ass(I_{\leq \ell}) = \{\mathfrak{m}_{\leq \ell}^{(i)},\ i \in$ subset of $\{1, \ldots, t\}\}$, and $\mathfrak{m}_{\leq \ell}^{(i)} = (p_1^{(i)}, \ldots, p_\ell^{(i)})$. The $\mathfrak{m}_{\leq \ell}^{(i)}$-primary component of $I_{\leq \ell}$ is the triangular set $\mathbf{t}_{\leq \ell}^{(i)} = (t_1^{(i)}, \ldots, t_\ell^{(i)})$.

Proof. That maximal ideals $\mathfrak{m}^{(i)}$ have a basis made of a triangular set is classical see *e.g.* [17, Section 5]. Other properties are direct consequences of the elimination property hold by the lexicographical order.

The result below is derived from [17, Prop. 5.2]. Comparing to the statement therein (recalled in Remark 1 (b) hereudner), it emphasizes the structure with respect to the irreducible polynomials p_i of Lemma 1, and helps understand why the specialization property may fail for non-triangular primary ideals (see the example in § 6).

Proposition 1 (2.2 of [12]). *The reduced lexGb of a primary ideal $\mathfrak{q} \subset k[x_1, \ldots, x_n]$ of associated prime $\mathfrak{p} = \langle p_1, \ldots, p_n \rangle$ of dimension zero can be written as follows:*

$$\begin{aligned}
g_{1,1}(x_1) &= p_1^{e_1} \\
g_{2,s_2}(x_1, x_2) &= p_2^{e_2} + \sum_{i_1=0}^{e_1-1} \sum_{i_2=0}^{e_2-1} c[i_1, i_2] p_1^{i_1} p_2^{i_2} \\
\vdots \qquad & \qquad \ddots \\
g_{n,s_n}(x_1, \ldots, x_n) &= p_n^{e_n} + \sum_{i_1=0}^{e_1-1} \cdots \sum_{i_{n-1}=0}^{e_{n-1}-1} \sum_{i_n=0}^{e_n-1} c[i_1, \ldots, i_n] \prod_{j=1}^{n} p_j^{i_j}
\end{aligned}$$

where: (i) $c[i_1,\dots,i_n] \in \bar{k}$
(ii) $c[0,\dots,0,i_\ell] = 0$ for all $i_\ell < e_\ell$ and for $\ell = 2,\dots,n$.
As for $h = g_{i,j}$ with $1 \le j < s_j$, w.l.o.g. assuming that $h \notin \bar{k}[x_1,\dots,x_{n-1}]$:

$$h = p_1^{\ell_1}\cdots p_n^{\ell_n} + \sum_{i_1=0}^{\ell_1-1}\cdots\sum_{i_{n-1}=0}^{\ell_{n-1}-1}\sum_{i_n=0}^{\ell_n-1} c[i_1,\dots,i_n]\prod_{j=1}^{n} p_j^{i_j}$$

where (ii) also holds for h, and moreover
(iii) $\ell_u < e_u$ for $1 \le u \le n$ and at least one $\ell_v \neq 0$ for $v < n$.

Remark 1. (a) When $p_i = x_i - \alpha_i$ (tis is always the case when k is algebraically closed), t is related to multivariate Taylor expansion, viz:
$c[i_1,\dots,i_n] = \frac{1}{i_1!i_2!\cdots i_{n-1}!i_n!}\frac{\partial^{i_1+\cdots+i_n} g_{n,s_n}}{\partial x_1^{i_1}\cdots\partial x_n^{i_n}}(\alpha_1,\dots,\alpha_n)$.
(b) The characterizations (i)-(iii) are equivalent to: $h(\alpha_1,\dots,\alpha_{n-1},x_n) = 0$ and $g_{i,s_i}(\alpha_1,\dots,\alpha_{i-1},x_i) = (x_i-\alpha_i)^{e_i}$ for all points $(\alpha_1,\dots,\alpha_n) \in \bar{k}^n$ solutions of the polynomial system $(p_1,\dots,p_n)$.

Assumption **(H)** asserts that the lexGb of all $\mathfrak{q}_i$ is *triangular* that is there is no polynomial h as above, only polynomials g_{i,s_i} for $i = 1,\dots,n$, denoted $g_{i,s_i} = t_i$. The following Lemma is critical for the definition of the prim-dec tree in the next paragraph 2.2.

Lemma 2. *Given two triangular sets $\mathbf{t}^{(i)}$ and $\mathbf{t}^{(j)}$ generating primary components of I, their* level *is the unique integer $0 \le \ell < n$ such that $t_k^{(i)} = t_k^{(j)}$ for $k = 1,\dots,\ell-1$ and $t_\ell^{(i)} \neq t_\ell^{(j)}$. Then $\langle t_\ell^{(i)}\rangle + \langle t_\ell^{(j)}\rangle = \langle 1\rangle$ modulo $\langle \mathbf{t}_{\le \ell-1}^{(i)}\rangle$.*

Proof. This is a standard consequence of Krull's theorem: the ideal $\mathfrak{p} := \mathfrak{q}_{\le\ell}^{(i)} + \mathfrak{q}_{\le\ell}^{(j)}$ is not contained is any maximal ideal of $k[x_1,\dots,x_\ell]/I$ hence is equal to I itself.

2.2. The primary decomposition prim-dec tree

Write $\{\mathbf{t}^{(i)},\ i = 1,\dots,t\}$ the triangular sets generating the primary ideals of I, thereby $Ass(I) := \{\sqrt{\langle \mathbf{t}^{(i)}\rangle}\}$ is the set of associated primes to I over k. We organize the data into a labeled tree $\mathcal{T}(I)$, which makes really sense only when several lexGb's of input primary triangular sets have a level larger than zero. As the results of this section shows a general lexGb has many more polynomials than the number of variables only when this is the case (under the Assumption **(H)**).

(1) The root of $\mathcal{T}(I)$ is denoted "root" and is unlabeled.
(2) depth 1 children are the *distinct* polynomials $\{t_1^{(i)},\ i \in \{1,\dots,t\}\}$. We index this set by children nodes of root and write it $\{t_1^{(N)},\ N \in \mathrm{Child}(\mathsf{root})\}$.
(3) To a depth 1 node N, the label $\delta_1(N)$ is the degree in x_1 of $t_1^{(N)}$.
(4) (**Recursion**) Assume the tree is defined up to depth $\ell < n$ and let N be a node at depth ℓ, with its labels $t_\ell^{(N)}$ and $\delta_\ell(N) = \deg_{x_\ell}(t_\ell^{(N)})$.
Let $N_0 = \mathsf{root}$, $N_\ell = N$ and $(N_0, N_1,\dots,N_\ell)$ the unique path from the root to N.
The children of N are the *distinct* $(\ell+1)$-th polynomial $t_{\ell+1}^{(i)}$ of a triangular set $\mathbf{t}^{(i)}$ in the family of all triangular sets generating the primary ideals of I, for which $(t_1^{(i)},\dots,t_\ell^{(i)}) = (t_1^{(N_1)},\dots,t_\ell^{(N_\ell)})$.
This set of polynomial is now indexed by Child(N) and for $P \in$ Child(N), we denote unambiguously the polynomial at P by $t_{\ell+1}^{(P)}$.

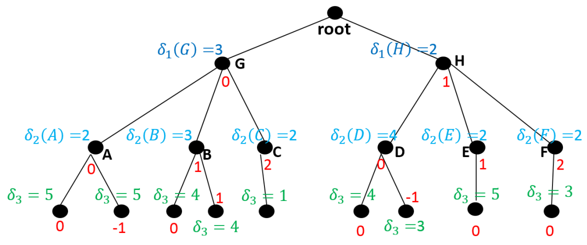
(5) Consider a child $P \in \text{Child}(N)$. The label $\delta_{\ell+1}(P)$ is $\deg_{x_{\ell+1}}(t_{\ell+1}^{(P)})$.

The correctness of the construction is justified by Lemma 2. From now on we will refer to triangular sets as elements organized in this tree, with the following notations.

Remark 2. (Notation update)

1. (Abuse of notation) If A is the ancestor at depth ℓ of a node B at depth $r > \ell$, then we tolerate the notation of the labels $\delta_\ell(A)$ and $t_\ell^{(A)}$ by $\delta_\ell(B)$ and $t_\ell^{(B)}$ instead, since the uniqueness of A (being an ancestor) leaves no ambiguity.
2. By extension the unique path $B_0, \dots, B_r$ from the root $B_0 = \texttt{root}$ to $B = B_\ell$ is uniquely determined by B. Hence B uniquely determines the polynomials at the nodes B_i in the path. Thus B uniquely determines the triangular set $(t_1^{(B_1)}, \dots, t_{\ell-1}^{(B_{\ell-1})})$, which can be denoted $\mathbf{t}_{\leq \ell-1}^{(B)} = (t_1^{(B)}, \dots, t_{\ell-1}^{(B)})$.
3. There is a one-one correspondence between the leaves of the tree $\mathcal{T}(I)$ and the primary ideals $\mathfrak{q}_i = \langle \mathbf{t}^{(i)} \rangle$, $i = 1, \dots, t$. According to the remarks above, from now on we will denote the primary decomposition of I as $I = \prod_{L \in \text{Leaves}(\mathcal{T}(I))} \langle \mathbf{t}^{(L)} \rangle$.

Example 2. Consider $I \subset \mathbb{C}[x_1, x_2, x_3]$ having nine primary components whose associated primes are ideals of points in $\mathbb{C}^3$, displayed on the figure below



The coordinates of points are in red, and a point is given by the paths from depth 1 nodes to depth 3 nodes: for example from left to right we read: $(0, 0, 0)$ $(0, 0, -1)$ $(0, 1, 0)$ $(0, 1, 1)$ etc. $(1, 2, 0)$.

The shape of the tree depends only on the configuration of the points. On top of the coordinates are added the degree labels $\delta_i(\,.\,)$ at each node. The first triangular set is given below, associated to the point $p = (0, 0, 0)$ is (for example):

$$\begin{cases} t_1^{(p)}(x_1) = x_1^3 \quad (\delta_1(G) = 3) \\ t_2^{(p)}(x_1, x_2) = x_2^2 + (2x_1^2 - x_1)x_2 \quad (\delta_2(A) = 2) \\ t_3^{(p)}(x_1, x_2, x_3) = x_3^5 + x_3^4(x_2^2 - x_1 + x_1 x_2) - x_3 \quad (\delta_3 = 5) \end{cases}$$

Remark 3. With the notation 2 of § 2, and by definition of $I_{\leq \ell} = I \cap k[x_1, \dots, x_\ell]$, one has $\mathcal{T}(I_{\leq \ell}) = \mathcal{T}(I)_{\leq \ell}$.

Algorithms and proofs are recursive on this tree and to handle these conveniently some properties are useful:

Proposition 1. *Given a node R not a leaf of $\mathcal{T}(I)$ at depth ℓ, write $A_R := k[x_1,\ldots,x_\ell]/\langle \mathbf{t}^{(R)}_{\leq \ell}\rangle$ and*
$I_R := I \otimes A_R$. The ring A_R is Henselian and $I_R \subset A_R$ is zero-dimensional. The prim-dec tree $\mathcal{T}(I_R)$ is identified with the subtree of $\mathcal{T}$ whose root is R (see 2 in § 2).

If $R = \mathsf{root}$, that is $\ell = 0$, then by convention $A_R = k$, $I_R = I$ and $\mathcal{T}(I_R) = \mathcal{T}(I)$.

Proof. That A_R is Henselian is well-known see *e.g.* [12, Theorem 3.1]. According to Lemma 2, the irredundant primary decomposition of $I_{\leq \ell}$ is

$$\prod\nolimits_{\text{node } R \text{ at depth } \ell \text{ in } \mathcal{T}(I)} \langle \mathbf{t}^{(R)}_{\leq \ell}\rangle.$$

Then for all sibling node L or R (if there are) , $\langle \mathbf{t}^{(L)}_{\leq \ell}\rangle + \langle \mathbf{t}^{(R)}_{\leq \ell}\rangle = 1$ (see Lemma 2), therefore the canonical map $\psi_R : k[x_1,\ldots,x_n] \to A_R[x_{\ell+1},\ldots,x_n]$ sends $\langle \mathbf{t}^{(L)}_{\leq \ell}\rangle$ to A_R, given that $\psi_R(\langle \mathbf{t}^{(L)}_{\leq \ell}\rangle)$ contains a unit. It follows that the image of the primary decomposition of I by ψ_R in $A_R[x_{\ell+1},\ldots,x_n]$ consists of the product of these primary ideals $\psi_R(\langle \mathbf{t}^{(L)}\rangle)$, where L is a leaf descendant of R. Therefore ψ_R sends the subtree of $\mathcal{T}(I)$ rooted at R to $\mathcal{T}(I_R)$ in a one-one fashion, preserving all labels.

Finally a corollary ends this section.

Corollary 1. *With the same notation as in Prop.1, one has*

$$I_R \simeq \left(\prod\nolimits_{C\in\mathrm{Child}(R)} t^{(C)}_{\ell+1}\right) \bmod \langle \mathbf{t}^{(R)}_{\leq \ell}\rangle.$$

Proof. Results from: $\mathrm{Leaves}(\mathcal{T}(I_R)) = \cup_{C\in\mathrm{Child}(R)}\mathrm{Leaves}(\mathcal{T}(I_C))$, according to the previous proposition.

3. The exponent trees and other trees

3.1. Some specifications

Exponent trees. To each node not a leaf N of $\mathcal{T}(I)$, say at depth $\ell < n$ in $\mathcal{T}(I)$, Algorithm 1 computes an "exponent tree[1]" denoted $\mathcal{E}_I(N)$ or simply $\mathcal{E}(N)$ if the dependence to I or $\mathcal{T}(I)$ is unambiguous from the context. It encodes the exponents of leading exponents, and has the following specificities, clear from the definition but precising them here makes them more transparent.

(i) it has depth $\ell = \mathrm{depth}(N)$ (with the convention that the depth of $\mathsf{root}(\mathcal{T}(I))$ is zero).
(ii) each node is labeled by a nonnegative integer.
(iii) unlike the tree $\mathcal{T}(I)$ the root of $\mathcal{E}(N)$ is labeled. It is equal to $\sum_{L\in\mathrm{Child}(N)} \delta_{\ell+1}(L)$ (L is a node in $\mathcal{T}(I)$ at depth $\ell + 1$).
(iv) all nodes of $\mathcal{E}(N)$ (except leaves) at a same depth $1 \leq d \leq \ell$ have the same number of children whose labels are consecutive nonnegative integers like $[a, a+1, \ldots, a+\delta_{\ell-d}(N)-1]$ (see Specification 5 thereafter for the definition of a).

[1]and actually two other trees, one called the discard tree, and the other the interpolate tree, denoted respectively $\mathcal{N}(I)$ and $\mathcal{R}(I)$

(v) in particular, if the triangular sets all have degree 1, whence $\delta_{\ell-d}(N) = 1$ (when I is radical and the primary ideals are ideal of points) then the tree $\mathcal{E}(N)$ is made of one branch joining the root to the leaf. It is then not necessary to define these trees in the case of ideal of points. This is a novelty compared to [11, 8, 6].

Remark 4. The n-uplet of integers $(\nu_{\ell+1}, \ldots, \nu_{\ell+d-1})$ is identified as the labels at each node in the path from the root to ν. This sequence of integers hereby obtained is *unique* in the tree $\mathcal{E}(N)$. Therefore, nodes of $\mathcal{E}(I)$ can be identified unambiguously with this sequence of nonnegative integers. In particular, writing $(\nu_{\ell+1}, \ldots, \nu_{\ell+d-1}) \in \mathcal{E}(N)$ carries the following information:

1) N is at depth $\ell \in \mathcal{T}(I)$

2) the node ν pointed by the path is at depth d in $\mathcal{E}(N)$.

3) the node pointed by $(\nu_{\ell+1}, \ldots, \nu_{\ell+d-1-j})$ is at depth $d-j$ in $\mathcal{E}(N)$ and is the j-th ancestor of ν.

Each path from the root to a leaf in such an exponent tree encodes an ℓ-uplet that will be shown in the next section to be a leading exponent of a polynomial in I. Those polynomials corresponding to minimal exponent for the semi-order $\leq_{mon}$ will be polynomials in a minimal lexGb of I, as described in Section 5.

Algorithms 1, 2, 3 show how these exponent trees (as well as their cousins the discard and interpolate trees) have their labels filled by nonnegative integers.

Two other trees or TRIE. While the exponent trees encode the leading exponents, it is necessary to record more data if one wants to reconstruct the polynomials in the lexGb (Section 4). It is realized by two other trees, similar to exponent trees, attached to each node except leaves of $\mathcal{T}(I)$.

They are denoted $\mathcal{N}_I(N)$ and $\mathcal{R}_I(N)$ called "discard" and "interpolate" trees respectively. The definition of these trees is naturally related to how are defined the nodes of $\mathcal{E}_I(N)$. In particular the same algorithms that compute $\mathcal{E}_I(N)$ also compute $\mathcal{N}_I(N)$ and $\mathcal{R}_I(N)$. We gather in this paragraph some properties that are convenient to understand what these trees are:

1. For each $N \in \mathcal{T}(I)$ not a leaf, the three trees $\mathcal{E}_I(N)$, $\mathcal{N}_I(N)$ and $\mathcal{R}_I(N)$ have the same tree shape (depth, number of children etc.).
2. In particular the unique path from the root to a node in $\mathcal{E}_I(N)$ also uniquely determines a path in $\mathcal{N}_I(N)$ and $\mathcal{R}_I(N)$.
3. Let N be a node at depth ℓ in $\mathcal{T}(I)$.
   To a path $\boldsymbol{\sigma} = (\sigma_{\ell+1}, \ldots, \sigma_{\ell+1-d})$ from the root to a node at depth d in the exponent tree $\mathcal{E}_I(N)$ the corresponding labels in $\mathcal{N}_I(N)$ and in $\mathcal{R}_I(N)$ are denoted by $\mathcal{N}_I(N, \boldsymbol{\sigma})$ and by $\mathcal{R}_I(N, \boldsymbol{\sigma})$, respectively.
4. The label $\mathcal{N}_I(N, \boldsymbol{\sigma})$ of a node of the discard tree $\mathcal{N}_I(N)$ is a set of nodes of $\mathcal{T}(I)$ at depth $\ell - d$; more precisely a subset of sibling nodes of $C = P_{d-1}(N) \in \mathcal{T}(I)$, the $d-1$-th ancestor of C.
5. the labels of the children (at depth $d+1$) of the path $\boldsymbol{\sigma}$ in $\mathcal{E}_I(N)$ are $[a, \ldots, a + \delta_{\ell-d}(N) - 1]$ (See (iv)), where:
$$a = \sum_{A \in \mathcal{N}_I(N, \boldsymbol{\sigma})} \delta_{\ell-d}(A).$$
   (Note that according to the identification of Remark 4 (or inherent to a trie) the set of nonnegative integers $\{(\nu_{\ell+1}, \ldots, \nu_{\ell+1-d}, i) \ : \ i = 1, \ldots, a + \delta_{\ell-d}(A) - 1\}$ can be identified with these children).

6. $\mathcal{R}_I(N, \boldsymbol{\sigma})$ is an *array*. Each $S \in \mathrm{Child}(P_d(N))$, $S \notin \mathcal{N}(N, \boldsymbol{\sigma})$, is a key for the array $\mathcal{R}_I(N, \boldsymbol{\sigma})$ for which the stored label is denoted $R_I(N, \boldsymbol{\sigma})[S]$. For example if

$$\mathcal{R}_I(N, \boldsymbol{\sigma})[S] := [C, \boldsymbol{\tau}],$$

then C is a descendant of S, say at depth $\ell_C \leq \ell = \mathrm{depth}(N)$, as well as a path $\boldsymbol{\tau} = (\tau_{\ell_C+1}, \dots, \tau_{\ell-d+1}) \in \mathcal{E}(S)$. It also verifies:

$$(\tau_{\ell_C+1}, \dots, \tau_{\ell-d+1}) \leq_{mon} (\sigma_{\ell_C+1}, \dots, \sigma_{\ell-d+1}).$$

7. Note that in particular

$$\mathcal{N}_I(N, \boldsymbol{\sigma}) \cup \{\text{keys of the array } \mathcal{R}_I(N, \boldsymbol{\sigma})\} = \mathrm{Child}(P_d(N))$$

where the union is disjoint, and the r.h.s. is the set of children nodes of the d-th ancestor of N in $\mathcal{T}(I)$.

3.2. Algorithm

The main routine is Algo. 1 below. It fills the nodes of the exponent, discard and interpolate trees, hence do not return anything.

It is recursive on the children of the root of tree $\mathcal{T}(I)$. If C is such a child, then one recursive call at Line 6 fills all exponent, discard, interpolate trees of the tree $\mathcal{T}(I_C)$. Since this tree has depth $n-1$ and by Proposition 2 it is identified with the subtree $\mathcal{T}_R$ of $\mathcal{T}(I)$ rooted at C, it fills the exponent, discard and interpolate trees $\mathcal{E}(N), \mathcal{N}(N)$, $\mathcal{R}(N)$ entirely except the leaves, for any node N descendant of C. The missing values at leaves are completed by the "lastExpAssign" routine at Line 7.

This "lastExpAssign" routine is the hard part since these values depend on the whole tree. This must be done "globally" on the tree $\mathcal{T}(I)$, and captures the "lex game" of [6], or the "four-in-a-row" operations on standard monomials in [8]. We have put significant effort to put Algo. 2 into a framework that is *easy to implement*. For comparison, this is less the case for the "four-in-a-row" operation of [8] where no implementation has been reported (and even more for the attempt of generalization [9]). A simple implementation of the routine "isDivided" is proposed in Algorithm 3,

Correctness of Algo. 1 "assignExp": If the ideal I is in $k[x_1]$ then the prim-dec tree $\mathcal{T}(I)$ is of depth one and the algorithm ends at Line 3. In this case the only preleaf of $\mathcal{T}(I)$ is the root, whence only one exponent, discard and interpolate tree (the ones attached to $\mathsf{root}(\mathcal{T}(I))$), with one path limited to the root value of $\mathcal{E}(\mathsf{root})$ or $\mathcal{N}(\mathsf{root})$ or $\mathcal{R}(\mathsf{root})$. This value is computed at Lines 1-2-3. It fulfills the specification (iii). The specifications (5)-(7) related to the discard and interpolate trees are trivially satisfied (the other ones (1)-(4) are descriptive and need not be verified).

If the ideal I is in a polynomial ring of more than one variable, then the subtrees $\mathcal{T}(I_C)$ rooted at the child C of root are of depth $n-1 \geq 1$ and recursive calls are possible. Thus by induction hypothesis, after Line 6 of Algo 1, each node (not leaves) of the tree $\mathcal{T}(I)$ have their exponent, discard and interpolate trees filled completely except the labels at their leaves. Filling these missing values at the leaves is the role of the routine lastExpAssign at Line 7.

Remark 5. One could blink eyes at seeing that each node in the discard and interpolate trees $\mathcal{N}_J(\,.\,)$ $\mathcal{R}_J(\,.\,)$ have the same children. This is not optimal. However, this makes these trees have the same shape as the exponent trees; their label can be easily accessed by paths of the exponent tree (see specifications 1 3). Besides simplicity and convenience, this does not impact the efficiency.

Algorithm 1: assignExp($\mathcal{T}(I)$) (recursively)

```
Input: prim-dec tree $\mathcal{T}(I)$ of depth $n$
Output: None: assigns labels of exponent, discard and interpolate trees
        $\mathcal{E}(N)$, $\mathcal{N}(N)$, $\mathcal{R}(N)$ at each node $N$ (not leaves) of $\mathcal{T}(I)$
1 root of $\mathcal{E}(\mathsf{root}) \leftarrow \sum_{L\in \mathrm{Child}(\mathsf{root}(\mathcal{T}(I)))} \delta_1(L)$
2 root of $\mathcal{N}(\mathsf{root}) \leftarrow$ Child($\mathsf{root}$)
3 root of $\mathcal{R}(\mathsf{root}) \leftarrow$ []                        // empty array
4 if $n > 1$ then                                              // depth of $\mathcal{T}$ is $> 1$
5     for all children $C$ of root do
              // $\mathcal{T}(I_C)$ is the subtree of depth $n-1$ of $\mathcal{T}$ rooted at $C$
6         assignExp($\mathcal{T}(I_C)$)
7     lastExpAssign($\mathcal{T}$)
```

Correctness of Algo. 2 "lastExpAssign": Step 1, termination: The exponent tree $\mathcal{E}(\mathsf{root})$, discard tree $\mathcal{N}(\mathsf{root})$ and interpolate tree $\mathcal{R}(\mathsf{root})$ attached to the root of $\mathcal{T}$ are already filled by Lines 1- 2- 3 of Algo. 1. This algorithm then processes one subtree rooted at a child C of root after another: the "pop" operation at Line 3 removes one arbitrarily child C from the list of nodes "`toProcess`" after one, until "`toProcess`" is empty (insuring termination).

Step 2, any node N not a leaf of $\mathcal{T}(I)$ is treated: Inside the While Loop (Line 2 till the end), the algorithm fills the values of leaves of each exponent, discard and interpolate trees of all nodes that are descendant of C (currently popped node from "`toProcess`″). These descendant nodes are precisely the nodes of the tree $\mathcal{T}(I_C)$ (Proposition 1), as explained hereunder.

The two For loops of Lines 4-5 run through all the preleaves K (breadth-first depth-first) of the trees $\mathcal{T}((I_C)_{\leq \ell})$ (seen as embedded in $\mathcal{T}(I_{\leq \ell+1})$ by Proposition 1), for $\ell = 1, \ldots, n-1$. In this way, all nodes (not leaves) of the subtree $\mathcal{T}(I_C)$ (embedded in $\mathcal{T}(I)$) are covered. The algorithm is processing all the children C of root one after another (While Loop 2), therefore all nodes (not leaves) of $\mathcal{T}(I)$ are covered by Algo. "lastExpAssign".

Step 3: Now that we have proved that all nodes (except unnecessary leaves) are treated by the algorithm, it remains to check that all paths in the exponent, discard, interpolate trees $\mathcal{E}(K)$, $\mathcal{N}(K)$, $\mathcal{R}(K)$, for a preleaf K of the tree $\mathcal{T}((I_C)_{\leq \ell})$, have a value assigned to their leaves in accordance with the Specifications (5)-(7) and (iv) This is addressed by the For Loop of Line 6. The value ν_1 is incremented according to a "division test" over all siblings of C, described in Definition 2 hereafter, and realized by Algo. 3 isDivided($C, \boldsymbol{\nu}, \ell, L, \mathtt{bool}_{L,c}, \mathcal{T}(J)$) (here $L \in$ sibling(C), $\mathtt{bool}_{L,C}$ is defind hereunder).

The children of the node $\boldsymbol{\nu}$ in $\mathcal{E}(K)$, $\mathcal{N}(K)$ and $\mathcal{R}(K)$ are defined at Lines 15, 17,18 respectively. It is immediate to verify the specifications (iv), (1), and (5) above.

Remark 6. Let $\mathcal{E}(K)$ be an exponent tree as being filled in the for loop of Line 5. It follows from Line 15 that two different paths from the root to two different leaves never have the all same labels. By induction, one can assume that this is is the case for paths from the root to preleaves. Let $\boldsymbol{\nu} = (\nu_{\ell+1}, \ldots, \nu_2)$ be such a path. Then the children have all different lables as one can check in Line 15.

Algorithm 2: lastExpAssign($\mathcal{T}$)

Input: prim-dec tree $\mathcal{T}(J)$ of depth $m \geq 2$

Assumption: if K is any node at depth $1 \leq \ell < m$, $\mathcal{E}_J(K)$, $\mathcal{N}_J(K)$, $\mathcal{R}_J(K)$ are filled entirely except at the leaves.
(implying $\mathcal{E}_J(K)_{\leq \ell-1}$, $\mathcal{N}_J(K)_{\leq \ell-1}, \mathcal{R}_J(K)_{\leq \ell-1}$ are filled entirely)

Output: None: assign the values at the leaves of the exponent, discard and interpolate trees $\mathcal{E}_J(K)$, $\mathcal{N}_J(K)$, $\mathcal{R}_J(K)$, respectively; and for all nodes K of $\mathcal{T}$ at depth ℓ, for all $\ell < m$

```
1  toProcess ← Child(root)
2  while toProcess is not empty do
3      $C$ ← pop(toProcess)                                        // depth($C$) = 1
4      for $\ell = 1, \dots, m-1$ do                                       // top-down in $\mathcal{T}(J)$
5          for all preleaves $K$ of $\mathcal{T}(J)_{\leq \ell+1}$, descendant of $C$ do  // depth($K$) = $\ell$. Left to right
6              for all paths $\boldsymbol{\nu} = (\nu_{\ell+1}, \dots, \nu_2)$ in $\mathcal{E}_J(K)_{\leq \ell-1}$ do
7                  $\nu_1 \leftarrow 0$; nodes ← []; recursives ← []
8                  for $L \in$ sibling($C$) do                              // depth($L$) = 1
9                      bool , $S$ , $\sigma$ ← isDivided( $C$, $\boldsymbol{\nu}, \ell, L, L \in$ toProcess ?, $\mathcal{T}(J)$ )
10                     if bool then
11                         recursives ← recursives cat [($S, \boldsymbol{\sigma}$)]
                            // depth($K$) ≤ depth($S$) ≤ $m-1$
12                     else
13                         $\nu_1 \leftarrow \nu_1 + \delta_1(L)$
14                         nodes ← nodes cat [$L$]
15                 Assign the children of the path $\boldsymbol{\nu} = (\nu_{\ell+1}, \dots, \nu_2)$ in $\mathcal{E}_J(K)_{\leq \ell-1}$:
                    $[\nu_1, \nu_1 + 1, \dots, \nu_1 + \delta_1(C) - 1]$
16                 for $i = 0, \dots, \delta_1(C) - 1$ do
17                     $\mathcal{N}_J(K, (\nu_{\ell+1}, \dots, \nu_2, \nu_1 + i))$ ← nodes
18                     $\mathcal{R}_J(K, (\nu_{\ell+1}, \dots, \nu_2, \nu_1 + i))$ ← recursives
```

Definition 1. At Line 3 of Algo. 2, denote by $\texttt{toProcess}_C$ the content of the list $\texttt{toProcess}$ after C was popped from it.

In relation, $\texttt{bool}_{A,C}$ denotes the boolean $A \in \texttt{toProcess}_C\,?$.

Definition 2. Given a tree $\mathcal{S}$ of depth $t \geq 2$, a preleaf K, let $C = P_{t-2}(K)$ be the ancestor of K being a child of $\mathsf{root}(\mathcal{S})$. Assume that C has a sibling L. Write $\texttt{toProcess}_C \subset \text{sibling}(C) \subset \text{Child}(\mathsf{root}(\mathcal{S}))$.

For a path $\boldsymbol{\nu} = (\nu_t, \ldots, \nu_2)$ found at the exponent tree $\mathcal{E}(K)$ define the predicate isDivided$(C, \boldsymbol{\nu}, t-1, L, \texttt{bool}_{L,C}, \mathcal{S})$ equal to:

$$\begin{cases} \texttt{bool}, \text{a node } S, \boldsymbol{\tau} & \text{if } \texttt{bool} = \texttt{True} \\ \texttt{bool}, \_\_, \_\_ & \text{if } \texttt{bool} = \texttt{False} \end{cases}$$

where S is a node descendant of L, say at depth $1 \leq d \leq t-1$ in $\mathcal{S}$ (hence not a leaf) such that there is a path $\boldsymbol{\tau} = (\tau_{d+1}, \ldots, \tau_2)$ found at $\mathcal{E}(S)_{\leq d+1}$ verifying *one* of the following two conditions:

(c.1) $\texttt{bool}_{L,C} == \texttt{True}$ and $(0, \ldots, 0, \tau_{d+1}, \ldots, \tau_2) \leq_{mon} (\nu_t, \ldots, \nu_2)$, where $d+1 = t$ is possible.

(c.2) $\texttt{bool}_{L,C} == \texttt{False}$ and $(0, \ldots, 0, \tau_{d+1}, \ldots, \tau_2) <_{mon} (\nu_t, \ldots, \nu_2)$ ($d+1 = t$ is possible), *and* there is *no* preleaf M descendant of L carrying an exponent $\boldsymbol{\nu} \in \mathcal{E}(M)$ such that $\boldsymbol{\nu} = \boldsymbol{\sigma}$.

Lemma 3. *In Algorithm 3 "isDivided", denote* $\texttt{toProcess}$, C , ℓ, K *and* $\boldsymbol{\nu} = (\nu_{\ell+1}, \ldots, \nu_2)$ *the parameters at Line 1, Line 3, Line 4, Line 5 and Line 6. After the for loop of Line 6, we have*

$$\nu_1 = \sum_{L \in \mathcal{N}_I(K, \boldsymbol{\nu})} \delta_1(L). \tag{4}$$

Proof. According to the definition 2 of the predicate isDivided, and to Line 14 and 17 of Algo. 2, $L \in \mathcal{N}(K, \boldsymbol{\nu})$ iff $\texttt{bool} == \texttt{False}$. Now from Lines 10-13 this happens exactly when $\delta_1(L)$ contributes to ν_1, whence Formula (4).

Correctness of Algo. 3 "isDivided": We must prove that the output of the algorithm is in accordance with Definition 2.

If it exits at Line 7 then $\texttt{bool} == \texttt{True}$ and $k = \ell$ and $\texttt{bool}_{L,C} == \texttt{False}$ meaning that the node S is a preleaf, and that $L \notin \texttt{toProcess}_C$. Thus Condition (c.1) is not satisfied. But we have $\boldsymbol{\nu} = \boldsymbol{\sigma}$; this precisely means that Condition (c.2) of Def 2 is not met neither.

If it ends at Line 9, then $k < \ell$ meaning that when k was equal to ℓ (see the for loop at Line 1) the equality $\nu = \sigma$ if $L \notin \texttt{toProcess}_C (\Leftrightarrow \texttt{bool}_{L,C} == \texttt{False})$ was not detected. Therefore, Condition (c.2) is met. If $L \in \texttt{toProcess}$, and $\texttt{bool} == \texttt{True}$ as Line 8 insures, then this is Condition (c.1) that is met.

Finally, if it exits at the return of Line 10, then no exponent smaller or equal to $\boldsymbol{\nu}$ was found, thus none of Conditions (c.1) and (c.2) is verified and it returns $\texttt{bool} = \texttt{False}$, as required.

Example 3. The figure below shows a prim-dec tree $\mathcal{T}(I)$ of depth 3, as in Example 2. The nodes at depth 3 are not displayed, neither are the edges connecting them to their parents, the preleaves. Below each preleaf A,B,C,D,E,F there is the exponent tree $\mathcal{E}(A), \ldots, \mathcal{E}(F)$ as output by Algo. 1. The numbers in red correspond to the number computed at Line 1 in the recursive call to $\mathcal{T}(I_A), \ldots, \mathcal{T}(I_F)$ respectively.

It is easy to isolate the minimal exponents for the semi-order $\leq_{mon}$, they are:

$(5, 0, 0)$ at depth 0 (monomial in x_1 only), then $(8, 0, 0)$, $(7, 2, 0)$ at depth 1 (monomials in x_1, x_2).

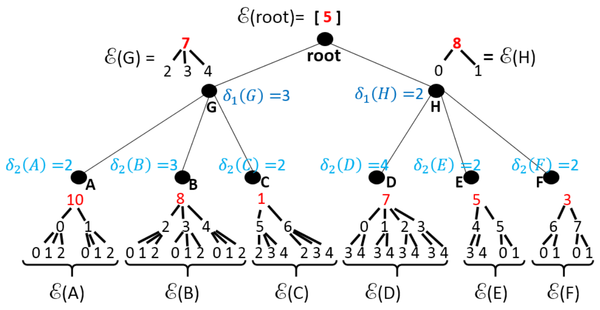
Algorithm 3: isDivided$(C, (\nu_{\ell+1}, \dots, \nu_2), \ell, L, \texttt{bool}_{L,C}, \mathcal{T})$

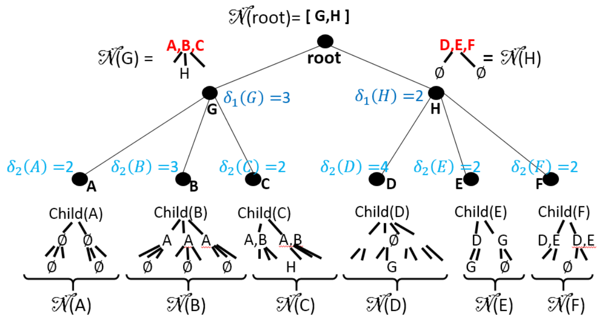
Input: $C \in \text{Child}(\mathsf{root}(\mathcal{T}))$ (in Algo. 2 it is popped from `toProcess` at Line 3)
ℓ-uplet $(\nu_{\ell+1}, \dots, \nu_2)$.
$L \in \text{Child}(\mathsf{root}(\mathcal{T}))$, $L \neq C$.
$\texttt{bool}_{L,C} == L \in \texttt{toProcess}_C$? is the boolean defined in Def. 1
The prim-dec tree $\mathcal{T}(I)$
Assumption: For all node K' descendant of L in $\mathcal{T}(J)$, the trees $\mathcal{E}_J(K'), \mathcal{N}_J(K'), \mathcal{R}_J(K')$ are filled, except at their leaves if K' is not passed through the for Loop 5 of Algo. 2
Output: As specified by Definition 2

```
1  for $k = \ell, \dots, 1$ do                                   // bottom-up from depth $\ell$ to 1
2      for all nodes $S$ at depth $k$ of $\mathcal{T}$, descendant of $L$ do
3          for all paths $(\sigma_{k+1}, \dots, \sigma_2)$ in $\mathcal{E}_{I_{\leq k}}(S)_{\leq k-1}$ do
4              bool ← $(\sigma_{k+1}, \dots, \sigma_2) \leq_{mon} (\nu_{k+1}, \dots, \nu_2)$ ?
5              if $k = \ell$ and not $\texttt{bool}_{L,C}$ and bool then
6                  if $\boldsymbol{\sigma} == \boldsymbol{\nu}$ then
7                      return False , __ , __
8              if $k < \ell$ and bool then
9                  return True , $S$ , $(\sigma_{k+1}, \dots, \sigma_2)$
10 return False , __ , __
```

At depth 3, from left to right: $(0, 0, 10)$, $(0, 2, 8)$, $(2, 5, 1)$, $(3, 0, 7)$, $(3, 4, 5)$, $(0, 5, 5)$, $(0, 6, 3)$.

It will therefore be sufficient to compute only polynomials (next section) for these leading exponents.





Let us run the algorithm 1 "assignExp" on this example:

Line 1: $\mathcal{E}(\mathsf{root}(\mathcal{T}(I))) = [\delta_1(G) + \delta_1(H)] = [5]$. This exponent tree has depth 0, therefore it is already fully filled.

Line 5: $\text{Child}(\mathsf{root}(\mathcal{T}(I))) = \{G, H\}$. First $C = G$. Enter

assignExp$(\mathcal{T}(I_G))$ ==

Line 1: $\mathcal{E}_{I_G}(\mathsf{root}(\mathcal{T}(I_G))) = \mathcal{E}_{I_G}(G)_{\leq 0}$ (See. Prop. 2). $\mathsf{root}(\mathcal{E}(G)) = [\delta_2(A) + \delta_2(B) + \delta_2(C)] = [7]$.

Enter recursive call on $\text{Child}(G) = \{A, B, C\}$. First with A:

assignExp$(\mathcal{T}(I_A))$ ==

Line 1: $\mathcal{E}_{I_A}(\mathsf{root}(\mathcal{T}(I_A))) = \mathcal{E}_{I_A}(A) = \mathcal{E}(A)_0$ (See. Prop. 2).

$\mathsf{root}(\mathcal{E}(A)) = [\sum_{a \in \text{Child}(A)} \delta_3(a)] = [10]$. Exit assignExp$(\mathcal{T}(I_A))$.

Next child of G is B.

assignExp$(\mathcal{T}(I_B))$ ==

Line 1: $\mathcal{E}_{I_B}(\mathsf{root}(\mathcal{T}(I_B))) = \mathcal{E}_{I_B}(B) = \mathcal{E}(B)_0$ (See. Prop. 2).

$\mathsf{root}(\mathcal{E}(B)) = [\sum_{b \in \text{Child}(B)} \delta_3(b)] = [8]$. Exit assignExp$(\mathcal{T}(I_B))$.

Next child of G is C.

assignExp$(\mathcal{T}(I_C))$ ==

Line 1: $\mathcal{E}_{I_C}(\mathsf{root}(\mathcal{T}(I_C))) = \mathcal{E}_{I_C}(C) = \mathcal{E}(C)_0$ (See. Prop. 2).

$\mathsf{root}(\mathcal{E}(C)) = [\sum_{c \in \text{Child}(C)} \delta_3(c)] = [1]$. Exit assignExp$(\mathcal{T}(I_C))$.

Back to assignExp$(\mathcal{T}(I_G))$ at Line 7 of Algo. 1.

Enter lastExpAssign$(\mathcal{T}(I_G))$ ==

Line 1: $\texttt{toProcess} \leftarrow \text{Child}(\mathsf{root}(\mathcal{T}(I_G))) = \text{Child}(G) = \{A, B, C\}$. Enter While loop (Line 2), and assume that the node A is popped first:

Line 4: $\ell = 1$. Line 5: since $\mathcal{T}((I_G)_{\leq 2}) = \mathcal{T}(I_G)$, preleaves are A, B, C. The only descendant of A is A.

Enter Line 6 with node A. Note that $\mathcal{E}_{(I_G)\leq 2}(A) = \mathcal{E}_{I_G}(A) = \mathcal{E}_I(A)_{\leq 1}$. At this stage only $\mathcal{E}_I(A)_{\leq 0}$ is filled with (10).

Line 8: $\text{sibling}(A) = \{B, C\}$. isDivided$(A, 1, (10), B, \texttt{toProcess}, \mathcal{T}(I_G))$ == $\texttt{True}, B, (8)$, similarly isDivided$(A, 1, (10), C, \texttt{toProcess}, \mathcal{T}(I_G))$ == $\texttt{True}, C, (1)$. Therefore $\nu_1 = 0$ and the children of [10] [0,1] since $\delta_2(A, \mathcal{T}(I)) = \delta_1(A, \mathcal{T}(I_G)) = 2$. Exit For loops 6 , 5, 4 (because $\text{depth}(\mathcal{T}(I_G) - 1 = 1)$.

Line 1 Next child is B. Line 4, $\ell = 1$. Enter For loop 5 with B (only preleaf of $\mathcal{T}(I_G)$, descendant of B).

Line 8: $\text{sibling}(B) = \{A, C\}$. isDivided$(B, 1, (8), A, \texttt{toProcess}, \mathcal{T}(I_G))$ == $\texttt{False}, \_, \_$, and isDivided$(B, 1, (8), C, \texttt{toProcess}, \mathcal{T}(I_G))$ == $\texttt{True}, C, (1)$. Therefore $\nu_1 = \delta_1(A, \mathcal{T}(I_g)) = \delta_2(A, \mathcal{T}(I)) = 2$ and Child((8)) in $\mathcal{E}(B)_{\leq 1}$ are $[2, 3, 4]$, since $\delta_1(B, \mathcal{T}(I_G)) = \delta_2(B, \mathcal{T}(I)) = 3$.

Exit for loops 6, 5 , 4 (because $\text{depth}(\mathcal{T}(I_G) - 1 = 1)$.

Line 1: Next child is C. Line 4, $\ell = 1$. Enter For loop 5 with C (only preleaf of $\mathcal{T}(I_G)$, descendant of C).

Line 8: $\text{sibling}(C) = \{A, B\}$. isDivided$(C, 1, (1), A, \texttt{toProcess}, \mathcal{T}(I_G))$ == $\texttt{False}, \_, \_$, and isDivided$(C, 1, (1), B, \texttt{toProcess}, \mathcal{T}(I_G))$ == $\texttt{False}, \_, \_$. Therefore $\nu_1 = \delta_2(A) + \delta_2(B) = 5$ and Child((1) in $\mathcal{E}(C)$ are $[5, 6]$ since $\delta_2(C) = 2$.

Exit for loops 6, 5 , 4 (because $\text{depth}(\mathcal{T}(I_G) - 1 = 1)$, exit while loop 1.

Exit lastExpAssign$(\mathcal{T}(I_G))$. Exit assignExp$(\mathcal{T}(G))$.

Back to assignExp$(\mathcal{T})$. Next child of root is H.

Enter assignExp($\mathcal{T}(I_H)$) ==

Line 1: $\mathcal{E}_{I_H}(\mathsf{root}(\mathcal{T}(I_H))) = \mathcal{E}_{I_H}(H) = \mathcal{E}(H)_{\leq 1}$ (See. Prop. 2). $\mathsf{root}(\mathcal{E}(H)) = [\delta_2(D) + \delta_2(E) + \delta_2(F)] = [8]$.

Enter recursive call on Child(H) $= \{D, E, F\}$. First with D:
assignExp($\mathcal{T}(I_D)$) ==
Line 1: $\mathcal{E}_{I_D}(\mathsf{root}(\mathcal{T}(I_D))) = \mathcal{E}_{I_D}(D) = \mathcal{E}(D)_0$ (See. Prop. 2).
$\mathsf{root}(\mathcal{E}(D)) = [\sum_{d \in \text{Child}(D)} \delta_3(d)] = [7]$. Exit assignExp($\mathcal{T}(I_D)$).
Next child of H is E. assignExp($\mathcal{T}(I_E)$) ==
Line 1: $\mathcal{E}_{I_E}(\mathsf{root}(\mathcal{T}(I_E))) = \mathcal{E}_{I_E}(E) = \mathcal{E}(E)_0$ (See. Prop. 2).
$\mathsf{root}(\mathcal{E}(E)) = [\sum_{e \in \text{Child}(E)} \delta_3(e)] = [5]$. Exit assignExp($\mathcal{T}(I_E)$).
Next child of H is F. assignExp($\mathcal{T}(I_F)$) ==
Line 1: $\mathcal{E}_{I_F}(\mathsf{root}(\mathcal{T}(I_F))) = \mathcal{E}_{I_F}(F) = \mathcal{E}(F)_0$ (See. Prop. 2).
$\mathsf{root}(\mathcal{E}(F)) = [\sum_{f \in \text{Child}(F)} \delta_3(F)] = [3]$. Exit assignExp($\mathcal{T}(I_F)$).
Back to assignExp($\mathcal{T}(I_H)$) at Line 7.
Enter lastExpAssign($\mathcal{T}(I_H)$) ==

Line 1: Child($\mathsf{root}(\mathcal{T}(I_H))$) = Child($H$) $= \{D, E, F\}$. Enter while loop 2 Assume that the first element popped a Line 3 is the node D:

Line 4: $\ell = 1$. Line 5: since $\mathcal{T}((I_H)_{\leq 2}) = \mathcal{T}(I_H)$, preleaves are D, E, F. The only one descendant of D is D.

Enter Line 6 with the partial path $(7) \in \mathcal{E}_{I_{H \leq 2}}(D) = \mathcal{E}_{I_H}(D) = E_I(D)_{\leq 1}$. At Line 8, sibling($D$) $= \{E, F\}$. isDivided($D, 1, (7), E,$ `toProcess`$, \mathcal{T}(I_H)$) == `True`, $E, (5)$, similarly isDivided($D, 1, (7), F,$ `to`
`True`, $F, (3)$. Therefore $\nu_1 = 0$ and Child((7)) in $\mathcal{E}(D)$) are $[0, 1, 2, 3]$ since $\delta_2(D) = 4$. Exit for loops 8, 5 , 6, 4 (because depth($\mathcal{T}(I_H)$) $- 1 = 1$).

Line 3 Next child is E. Line 4, $\ell = 1$. Enter For loop 5 with E (only preleaf of $\mathcal{T}(I_H)$, descendant of E).

Enter Line 6 with the partial path $(5) \in \mathcal{E}_{I_{H \leq 2}}(E) = \mathcal{E}_{I_H}(E) = E_I(E)_{\leq 1}$. At Line 8, sibling($E$) $= \{D, F\}$.

isDivided($E, 1, (5), D,$ `toProcess`$, \mathcal{T}(I_H)$) == `False`, __, __, and isDivided($E, 1(5), F,$ `toProcess`$, \mathcal{T}(I_H)$) == `True`, $F, (3)$. Therefore $\nu_1 = \delta_2(D) = 4$ and Child((8)) in $\mathcal{E}(E)$) are $[4, 5]$ since $\delta_2(E) = 2$.

Exit for loops 8, 6,= 5 , 4 (because depth($\mathcal{T}(I_H)$) $- 1 = 1$).

Line 3: Next child is F. Line 4, $\ell = 1$. Enter For loop 5 with F (only preleaf of $\mathcal{T}(I_H)$, descent of F).

Enter Line 6 with the partial path $(3) \in \mathcal{E}_{I_{H \leq 2}}(F) = \mathcal{E}_{I_H}(F) = E_I(F)_{\leq 1}$. At Line 8, sibling($F$) $= \{D, E\}$.

isDivided($F, 1, (3), D,$ `toProcess`$, \mathcal{T}(I_H)$) == `False`, __, __, and isDivided($F, 1, (3), E,$ `toProcess`$, \mathcal{T}(I_H)$) == `False`, __, __. Therefore $\nu_1 = \delta_2(D) + \delta_2(E) = 6$ and Child((3)) in $\mathcal{E}(F)$) are $[6, 7]$ since $\delta_2(F) = 2$.

Exit for loops 8, 6, 5 , 4 (because depth($\mathcal{T}(I_H) - 1 = 1$), exit while loop 1.

Exit lastExpAssign($\mathcal{T}(I_H)$). Exit assignExp($\mathcal{T}(I_H)$).

Back to assignExp($\mathcal{T}$), Line 7. Enter lastExpAssign($\mathcal{T}$) ==.
Line 3: `toProcess` $\leftarrow$ Child(root) $= \{G, H\}$. Assume that G is popped first.
Line 4: $\ell = 1$
Line 5. Preleaves of $\mathcal{T}(I_{\leq 2}) = \{G, H\}$, therefore the only preleaf descendant of G is G.
Enter Line 6 with the partial path $(7) \in \mathcal{E}_{I_{\leq 2}}(G) = \mathcal{E}_I(G)$. At Line 8, sibling($F$) $= \{H\}$.
isDivided($G, 1, (7), H,$ `toProcess`$, \mathcal{T}(I)$) == `False`, __, __. $\nu_1 = \delta_1(H) = 2$ and the children of (7) in $\mathcal{E}(G)$ are $[2, 3, 4]$ since $\delta_1(G) = 3$.

Exit for loops 8, 6, 5.

For loop 4: $\ell = 2$.

Line 5: prelaves of $\mathcal{T}(I_{\leq 3}) = \mathcal{T}(I)$ descendant of G are A, B, C. First take node A.

Enter Line 6 with the partial path $(10, 0) \in \mathcal{E}_{I_{\leq 3}}(A) = \mathcal{E}_I(A)$. At Line 8, sibling$(G) = \{H\}$.

isDivided$(A, 2, (10, 0), H,$ `toProcess`$, \mathcal{T}(I)) ==$ `True`$, D, (7, 0)$. Thus $\nu_1 = 0$ and the children of the path $(10, 0)$ in $\mathcal{E}(A)$ are $[0, 1, 2]$ since $\delta_1(G) = 3$.

Exit for loops 8. Enter Line 6 with the partial path $(10, 1)$ in $\mathcal{E}_I(A)$. We obtain isDivided$(A, 2, (10, 1), H,$ `toProc`
`True`$, D, (7, 1)$. Thus $\nu_1 = 0$ and the children of the path $(10, 1)$ in $\mathcal{E}(A)$ are $[0, 1, 2]$ since $\delta_1(G) = 3$.

Exit for loops 8, 6.

From now, all the steps are given with a fewer details since it always the same thing repeated

Next node in For loop 5 is B. isDivided$(B, 2, (8, 2), H,$ `toProcess`$, \mathcal{T}(I)) ==$ `True`$, D, (7, 0)$. $\nu_1 = 0$ and thus the children of $(8, 2)$ in $\mathcal{E}(B)$ are $[0, 1, 2]$ since $\delta_1(G) = 3$.

isDivided$(B, 2, (8, 3), H,$ `toProcess`$, \mathcal{T}(I)) ==$ `True`$, D, (7, 0)$. $\nu_1 = 0$ and the children of $(8, 3)$ in $\mathcal{E}(B)$ are $[0, 1, 2]$ since $\delta_1(G) = 3$.

isDivided$(B, 2, (8, 4), H,$ `toProcess`$, \mathcal{T}(I)) ==$ `True`$, D, (7, 0)$. $\nu_1 = 0$ and the children of $(8, 4)$ in $\mathcal{E}(B)$ are $[0, 1, 2]$ since $\delta_1(G) = 3$.

Next node in For loop 5 is C.

isDivided$(C, 2, (1, 5), H,$ `toProcess`$, \mathcal{T}(I)) ==$ `False`$, \_\_, \_\_$. $\nu_1 = \delta_1(H) = 2$ and the children of $(1, 5)$ in $\mathcal{E}(C)$ are $[2, 3, 4]$ since $\delta_1(G) = 3$.

isDivided$(C, 2, (1, 6), H,$ `toProcess`$, \mathcal{T}(I)) ==$ `False`$, \_\_, \_\_$. $\nu_1 = \delta_1(H) = 2$ and the children of $(1, 6)$ in $\mathcal{E}(C)$ are $[2, 3, 4]$ since $\delta_1(G) = 3$.

Exit For loops 5, 4.

In For loop 3, next child of Child(`root`) is H.

Line 4: $\ell = 1$

Line 5. Preleaves of $\mathcal{T}(I_{\leq 2}) = \{G, H\}$, therefore the only preleaf descendant of H is H.

isDivided$(H, 1, (8), G,$ `toProcess`$, \mathcal{T}(I)) ==$ `True`$, G, (7)$. $\nu_1 = 0$ and the children of (8) in $\mathcal{E}(H)$ are $[0, 1]$ since $\delta_1(H) = 2$.

Exit for loop 5.

$\ell = 2$ in For loop 4.

Line 5: preleaves of $\mathcal{T}(I_{\leq 3}) = \mathcal{T}(I)$ descendant of H are D, E, F. First enter node D.

isDivided$(D, 2, (7, 0), G,$ `toProcess`$, \mathcal{T}(I)) ==$ `False`$, \_\_, \_\_$. $\nu_1 = \delta_1(G) = 3$ and the children of $(7, 0)$ in $\mathcal{E}(D)$ are $[3, 4]$ since $\delta_1(H) = 2$.

Similarly for paths of $(7, 1), (7, 2), (7, 3)$ of $\mathcal{E}(D)_{\leq 2}$, the children are also $[3, 4]$.

Next node in For loop 5 is E.

isDivided$(2, (5, 4), H,$ `toProcess`$, \mathcal{T}(I)) ==$ `False`$, \_\_, \_\_$. $\nu_1 = \delta_1(H) = 3$ and the children of $(5, 4)$ in $\mathcal{E}(E)$ are $[3, 4]$ since $\delta_1(H) = 2$.

isDivided$(2, (5, 5), H,$ `toProcess`$, \mathcal{T}(I)) ==$ `True`$, C, (1, 5)$. $\nu_1 = 0$ and the children of $(5, 5)$ in $\mathcal{E}(E)$ are $[0, 1]$ since $\delta_1(H) = 2$.

Next node in For loop 5 is F.

isDivided$(2, (3, 6), , H,$ `toProcess`$, \mathcal{T}(I)) ==$ `True`$, C, (1, 5)$. $\nu_1 = 0$ and the children of $(3, 6)$ in $\mathcal{E}(F)$ are $[0, 1]$ since $\delta_1(H) = 2$.

Similarly the children of $(3, 7)$ in $\mathcal{E}(F)$ are $[0, 1]$ since $\delta_1(H) = 2$.

Exit For loops 5, 4. Exit lastExpAssign$(\mathcal{T})$. Exit assignExp$(\mathcal{T})$.

The main outcome of this construction is the following theorem:

Theorem 1. *Let $\mathscr{G}(I)$ be a minimal lexGb of I. The exponents of the leading monomial of the polynomials in $\mathscr{G}(I)$ are precisely the minimal (for the semi-order $\leq_{mon}$) exponents found among each exponent tree $\mathcal{E}(N)$, when N runs through all nodes not leaves of $\mathcal{T}(I)$.*

The proof of the theorem is performed in two steps:

Step 1. For each node N not a leaf, and each leaf $\boldsymbol{\nu}$ of $\mathcal{E}(N)$ construct explicitly a polynomial $f_{N,\boldsymbol{\nu}} \in I$, for which $\text{LM}(f_{N,\boldsymbol{\nu}}) = x_1^{\nu_1} \dots x_n^{\nu_n}$ (Algo. 4 "`computePoly`" and Theorem 2).

Step 2. The set of all such exponents ν contains the minimal exponents of the monomial ideal $\text{LM}(I)$ (Remark 9, Theorem 3, Corollary 3).

In order to achieve these two steps, the following Proposition is a basic but essential tool.

Proposition 2. *According to Algo. 1 and 2, we have:*

(i) *Given a node R at depth $\ell \leq n-1$ in $\mathcal{T}(I)$, the exponent tree $\mathcal{E}(R)$ (which by definition has depth ℓ) depends only on $\mathcal{T}(I_{\leq \ell+1}) = \mathcal{T}(I)_{\leq \ell+1}$:*

$$\mathcal{E}_{I_{\leq \ell+1}}(R) = \mathcal{E}_I(R).$$

(ii) *Given a preleaf N of $\mathcal{T}(I)$ the exponent tree $\mathcal{E}(N)_{\leq n-\ell-1}$ (where nodes at depth $> n-\ell-1$ are cut, see Definition 2) depends only on the subtree $\mathcal{T}_R$ of $\mathcal{T}(I)$ rooted at R, where $R = P_{n-\ell-1}(N)$ is the $(n-\ell-1)$-th ancestor of N, at depth ℓ in $\mathcal{T}(I)$.*

According to Proposition 1 the map

$$\psi_R : k[x_1, \dots, x_n] \to A_R[x_{\ell+1}, \dots, x_n]$$

induces a canonical embedding of $\mathcal{T}(I_R)$ into $\mathcal{T}(I)$ identified with the subtree $\mathcal{T}_R$ rooted at R. A consequence of (ii) is that ψ_R also yields the identification:

$$\mathcal{E}_{I_R}(N) = \mathcal{E}_I(N)_{\leq n-1-\ell}.$$

Remark 7. Since the discard tree $\mathcal{N}_I(R)$ and interpolate tree $\mathcal{R}_I(R)$ have nodes labelled by those of the exponent $\mathcal{E}_I(R)$, a consequence of (i) is that:

$$\mathcal{N}_{I_{\leq \ell+1}}(R) = \mathcal{N}_I(R) \quad \text{and} \quad \mathcal{R}_{I_{\leq \ell+1}}(R) = \mathcal{R}_I(R).$$

Similary, a consequence of (ii) is that:

$$\mathcal{N}_{I_R}(N) = \mathcal{N}_I(N)_{\leq n-1-\ell} \quad \text{and} \quad \mathcal{R}_{I_R}(N) = \mathcal{R}_I(N)_{\leq n-1-\ell}.$$

Proof. (i) is proved by increasing induction on $\ell = 0, \dots, n-1$.

When $\ell = 0$, $R = \mathsf{root}(\mathcal{T}(I))$ thus $\mathcal{E}(R) = \mathcal{E}(\mathsf{root}(\mathcal{T}(I))$ which has depth 0 hence is made up of one node, its root written $\mathsf{root}(\mathcal{E}(\mathsf{root}(\mathcal{T}(I))))$. Recall that the root value of all trees $\mathcal{E}(R)$ are computed at Line 1 of Algo. 1, by the formula: $\mathsf{root}(\mathcal{E}(\mathsf{root})) = \sum_{L \in \mathsf{Child}(\mathsf{root}(\mathcal{T}(I)))} \delta_1(L)$, which depends only on $\mathcal{T}(I)_{\leq 1}$.

Assume now $\ell > 0$. Let $C \in \mathcal{T}(I)$ be the ancestor of R at depth 1. Algo. 1 "assignExp" makes recursive calls at Line 6 with the rooted subtree $\mathcal{T}_C = \mathcal{T}(I_C)$ of $\mathcal{T}(I)$. Note that R is at depth $\ell - 1$ in $\mathcal{T}_R$ allowing to use the induction hypothesis: $\mathcal{E}_{I_C}(R)$ depends only on $(\mathcal{T}_C)_{\leq \ell} \hookrightarrow \mathcal{T}(I)_{\leq \ell+1}$.

Next we prove that $\mathcal{E}_{I_C}(R) \overset{(\times)}{=} \mathcal{E}_I(R)_{\leq \ell - 1}$. After the recursive call assignExp($\mathcal{T}(I_C)$) (Line 6) where trees $\mathcal{E}_{I_C}(R)$ are filled, and before entering "lastExpAssign" (Line 7), it is agreed (see *Assumption:* in Algo. 2) that all exponent trees $\mathcal{E}_I(R)$ are filled entirely, except the labels at their leaves. This is precisely what means Eq. $(\times)$ above, since the leaves of $\mathcal{E}_I(R)$ are at depth ℓ.

The labels of $\mathcal{E}_I(R)$ at leaves are computed in Algo. 2, precisely at Lines 13 and 15. All nodes manipulated within the outer loops 2, 4, 5, 6 belongs to $\mathcal{T}(I)_{\leq \ell+1}$: this is clear for the parameters of the loops; As for the nodes S returned by "isDivided" call at Line 9 when `bool == True`, they also belong to $\mathcal{T}(I)_{\leq \ell}$. This is because $\mathrm{depth}(S) \leq \mathrm{depth}(K) = \ell$ (notations therein) from the specification of the routine "isDivided′" defined in Definition 2. This achieves the proof of (i).

The proof of (ii) proceeds by (increasing) induction on the gap $g := \mathrm{depth}(\mathcal{S}) - 1 - \ell_R$, over all trees $\mathcal{S} = \mathcal{T}(J)$ built from an ideal J that verifies Assumption **(H)**, any node R at depth ℓ_R in $\mathcal{S}$, while N remains a preleaf in $\mathcal{S}$. In the present statement we have $\mathrm{depth}(\mathcal{T}(I)) = n$ hence $\mathrm{depth}(N) = n - 1$, $\mathrm{depth}(R) = \ell$ thus $g = n - 1 - \ell$.

The base case $g = 0$ occurs when $\ell = n - 1$ and corresponds to $\mathcal{E}(N)_{\leq n-\ell-1} = \mathcal{E}(N)_0 = \mathsf{root}(\mathcal{E}(N))$, and $R = P_0(N) = N$. The value at root of $\mathcal{E}(N)$ is computed at Line 1 of Algo. 1 when the stack of recursive calls reaches the input $\mathcal{T}(I_N) = \mathcal{T}_N$. Then $\mathsf{root}(\mathcal{E}(N)) = \sum_{L \in \mathrm{Child}(\mathsf{root}(\mathcal{T}_N))} \delta_1(L)$ viewed in $\mathcal{T}_N$, which is equal to $\sum_{L \in \mathrm{Child}(N)} \delta_{n-\ell}(L)$ when viewed in $\mathcal{T}(I)$. Hence $\mathcal{E}(N)_{\leq g} = \mathcal{E}(N)_0 = \mathcal{E}_{I_N}(N)$.

Next assume that $\mathcal{E}_{J_Q}(N) = \mathcal{E}_J(N)_{\leq g}$ for all ideal J verifying Assumption **(H)**, for any node Q at depth ℓ_Q in $\mathcal{S} := \mathcal{T}(J)$ such that $\mathrm{depth}(\mathcal{S}) - \ell_Q \leq g$ (here $g = n - \ell - 1$).

Let $R' = P_1(R) = P_{n-\ell}(N)$. This node is at depth $\ell_{R'} = \ell - 1$ in $\mathcal{T}$. Let $J = I_{R'}$ and $\mathcal{S} := \mathcal{T}(J)$. It has depth $n - \ell + 1$ and R is at depth $\ell_Q := 1$ in $\mathcal{S}$. Therefore $\mathrm{depth}(\mathcal{S}) - \ell_Q = n - \ell + 1 - \ell_Q = n - \ell - 1 \leq g$, the induction hypothesis applies to $\mathcal{S}$, R and the preleaf N yielding $\mathcal{E}_{J_R}(N) = \mathcal{E}_J(N)_{\leq g}$. On the other hand, $J_R = (I_{R'})_R \simeq I_R$ since $R' = P_1(R)$, implying $\mathcal{E}_{I_R}(N) \simeq \mathcal{E}_{I_{R'}}(N)_{\leq g}$. Besides, induction hypothesis gives $\mathcal{E}_{I_R}(N) \simeq \mathcal{E}_I(N)_{\leq g}$. Finally, $\mathcal{E}_I(N)_{\leq g} \simeq \mathcal{E}_{I_{R'}}(N)_{\leq g}$.

Consider the algorithm "assignExp" called with input $\mathcal{T}(I_{R'})$. The recursive call at Line 6 of Algo. 1 with the child R of node R', fills the "missing" labels of the leaves, which are at depth $n - \ell$, of the tree $\mathcal{E}_{I_{R'}}(N)$. Because $\mathcal{E}_I(N)_{\leq g} \simeq \mathcal{E}_{I_{R'}}(N)_{\leq g}$ as shown above, these values are also assigned by construction to be the values at depth $n - \ell = g + 1$ in $\mathcal{E}_I(N)$. Hence $\mathcal{E}_I(N)_{\leq g+1} \simeq \mathcal{E}_{I_{R'}}(N)_{\leq g+1} = \mathcal{E}_{I_{R'}}(N)$, achieving the proof by induction.

4. Recombination polynomials

Given the prim-dec tree $\mathcal{T}(I)$ with at each of its nodes not leaf its associated exponent, discard, interpolate trees, this section shows how to compute polynomials $f_{N,\boldsymbol{\nu}}$ introduced in Step 1. above. An essential component of this construction is a "recombination" from the generators of the input primary ideals, like in the (recombination part) of the Chinese Remainder Theorem (CRT). The CRT can be restated in term of *idempotents* in a suitable algebra; We will use and present only the minimal necessary properties of them, and externalize the algorithmic construction and proof which is the purpose of [18].

4.1. Idempotents

The case where input primary ideals are ideals of points, the problem reduces to classical "multivariate" Lagrange interpolation. It is well-known in this case how to compute interpolating polynomials: the "multivariate" ones are products of univariate Lagrange idempotents for which explicit formula exists (this is classical: see Example in [18] below).

These idempotents are less obvious to define for general triangular ideals. This stems from two difficulties. First that there is no explicit formula for more general ideals than that of points, even in the radical case; For the latter this can be overcome by computing cofactors (Bézout coefficients) with an Extended Remainder sequence over *a field.* Even in this radical case, previous works are seldom, if inexistent, for more complicated triangular sets than $(x_1 - \alpha_1, \dots, x_n - \alpha_n)$ ([8, 6] do give explicit construction, but are constrained to ideal of points).

A second difficulty is when a triangular set defines a primary non-maximal ideals ([3, 5] do not come with an explicit construction and relies on linear algebra to bypass the difficulty, and [9] has some drawbacks pointed out in § 1.1). The realization that this case is not fundamentally different than the radical one is apparently very recent: cofactors exist and can also be computed by an Extended Remainder sequence. This is the outcome of the articles [12, 18]. The work [12] highlights a unique factorization property of monic univariate polynomials defined over a primary ideal, and how to compute gcds in some special cases, in which fall the settings of this work as shown in [18]. This latter article [18] computes explicitly cofactors and thus idempotents.

Let L be a leaf of the prim-dec tree $\mathcal{T}(I)$ and let $R = P_{n-\ell-1}(L)$ be its ancestor at depth ℓ.

Denote by $t_\ell^{(L)}$ the ℓ-th polynomial of the triangular set $\mathbf{t}^{(L)}$ generating a primary component of I. Proposition 1 shows that the ring $A_R := k[x_1, \dots, x_\ell]/\langle \mathbf{t}_{\leq \ell}^{(R)} \rangle$ is *Henselian.* Let $\mathcal{D} \subset \text{Child}(R)$ and $T_{\ell+1}^{(\mathcal{D})} := \prod_{C \in \mathcal{D}} t_{\ell+1}^{(C)} \bmod \langle \mathbf{t}_{\leq \ell}^{(R)} \rangle$. Write $A_{\mathcal{D}} := A_R[x_{\ell+1}]/\langle T_{\ell+1}^{(\mathcal{D})} \rangle$. We introduce a family of orthogonal idempotents of the algebra $A_{\mathcal{D}}$.

Proposition 3 (Prop. 1 [18]). *For each $C \in \mathcal{D}$, define:*
$\widetilde{e}(C) \equiv \prod_{C' \in \mathcal{D}, C' \neq C} t_{\ell+1}^{(C')} \bmod \langle \mathbf{t}_{\leq \ell}^{(R)} \rangle$.

1. *$\widetilde{e}(C)$ is a polynomial in $A_R[x_{\ell+1}]$*
2. *Orthogonality: Given $C \neq C' \in \mathcal{D}$,* $\widetilde{e}(C) \cdot \widetilde{e}(C') = 0$ *in $A_{\mathcal{D}}$.*
3. *$\widetilde{e}(C) \equiv 0 \bmod \langle \mathbf{t}_{\leq \ell+1}^{(C')} \rangle$, if $C' \neq C$ are in $\mathcal{D}$.*
4. *Bézout identity provides cofactors $u_{\ell+1}(C)$ and v such that $u_{\ell+1}(C)\,\widetilde{e}(C) + v\, t_{\ell+1}^{(C)} \equiv 1 \bmod \langle \mathbf{t}_{\leq \ell}^{(R)} \rangle$. Denote*
$$e(C) \equiv u_{\ell+1}(C)\widetilde{e}(C) \bmod \langle \mathbf{t}_{\leq \ell}^{(R)} \rangle.$$
The family $\{e(C)\}_{C \in \mathcal{D}}$ is a complete family of orthogonal idempotents of the algebra $A_{\mathcal{D}}$:
$$\sum_{C \in \mathcal{D}} e(C) = 1. \tag{5}$$
5. *$e(C) \equiv 1 \bmod \langle \mathbf{t}_{\leq \ell+1}^{(C)} \rangle$, that is $u_{\ell+1}(C)$ is the inverse of $\widetilde{e}(C)$ modulo $\mathbf{t}_{\leq \ell+1}^{(C)}$.*

Before ending the section, let us emphasize an easy and "well-known" but fundamental property of the lexicographic order, which appears crucial so that a short proof is provided.

Lemma 4 (Property of lex order). *With the notations above, let a family of monic polynomials $\{f^{(C)} \in A_C[x_{\ell+2}, \dots, x_n] \ : \ C \in \mathcal{D}\}$, seen as embedded in $A_R[x_{\ell+1}, \dots, x_n]$, thereby* $\deg_{x_{\ell+1}}(f^{(C)}) < \delta_{\ell+1}(C) = \deg_{x_{\ell+1}}(t^{(C)}_{\ell+1})$*. Assume that they all have the same leading monomial* $\text{LM}(f^{(C)}) = x_{\ell+1}^{\alpha_{\ell+1}} \cdots x_n^{\alpha_n} = m$*. Because $f^{(C)}$ is monic over A_C, $\alpha_{\ell+1} = 0$ hence $m = x_{\ell+2}^{\alpha_{\ell+2}} \cdots x_n^{\alpha_n}$.*

Then the polynomial $f := \sum_{C \in \mathcal{D}} f^{(C)} e(C) \in A_R[x_{\ell+1}, \dots, x_n]$ has a leading monomial $\text{LM}(f) = m$ as well.

Proof. Let $m_C \neq m$ be a monomial occurring in $f^{(C)}$, hence $m_C \precneqq_{lex} m$, and $a_c m_C$ the corresponding term, with $a_C \in A_C \hookrightarrow A_R[x_{\ell+1}]$, $\deg_{x_{\ell+1}}(a_C) < \delta_{\ell+1}(C)$. Then

$$\text{LM}(\sum_{C \in \mathcal{D}} a_C m_C e(C)) \precneqq_{lex} m.$$

Indeed, $m_C \precneqq_{lex} m$ in $A_C[x_{\ell+2}, \dots, x_n]$ and both a_C and $e(C)$ are in $A_R[x_{\ell+1}]$, therefore by the property of the lexicographic monomial order, $\text{LM}(a_C m_C e(C)) \precneqq_{lex} m$ for all C. In particular $\text{LM}(\sum_{C \in \mathcal{D}} a_C m_C e(C)) \precneqq_{lex} m$.

On the other hand, thanks to Eq. (5) of Prop. 3 (4), $\sum_{C \in \mathcal{D}} m e(C) = m \sum_{C \in \mathcal{D}} e(C) = m$. This implies that m is a monomial occurring in f. And finally that $\text{LM}(f) = m$.

4.2. Computation of polynomials

The algorithm 4 "`computePoly`" constructs the polynomials mentioned in Step 1. (just after Theorem 1). Its proof of correctness lies the ground for the following theorem. The factorization pattern of Eq. (2), recalled in the theorem, is a byproduct of the algorithm, proved in Lemma 5.

Theorem 2. *Let N be a preleaf of $\mathcal{T}(I)$, and $\boldsymbol{\sigma} = (\sigma_n \dots, \sigma_1)$ be a path (from the root to a leaf) of the exponent tree $\mathcal{E}(N)$. Algorithm 4 "`computePoly`" with input $R = \mathsf{root}$ outputs a polynomial $f_{N,\boldsymbol{\sigma}} \in I$ having for leading monomial $\text{LM}(f_{N,\boldsymbol{\sigma}}) = x_1^{\sigma_1} \cdots x_n^{\sigma_n}$.*

Moreover, it can also compute polynomials $\chi_j \in k[x_1, \dots, x_j]$ for $j = 1, \dots, n$ such that $\text{LM}(\chi_j) = x_j^{\sigma_j}$ and

$$f_{N,\sigma} \equiv \chi_1 \cdots \chi_n \bmod I_{\leq n-1}.$$

The proof of Theorem 2 is entirely based on the correctness of Algorithm 4, proved hereunder. The factorization pattern of the theorem is delayed to the next subsection 4.3, with the conclusion of the proof of Theorem 2 as well.

Proposition 4 (Correctness of Algo. 4). *The polynomial $h_{R,\boldsymbol{\sigma}} = \texttt{computePoly}(N, \boldsymbol{\sigma}, R)$ output by Algo. 4 verifies the required specifications: it belongs to I_R and $\text{LM}(h_{R,\boldsymbol{\sigma}}) = x_{\ell_N+1}^{\sigma_{\ell_N+1}} \cdots x_{\ell_R+1}^{\sigma_{\ell_R+1}}$.*

Proof. Following the recursive nature of the algorithm, the proof proceeds by induction on the gap $\ell_N - \ell_R$.

Step 1, the case $\ell_R = \ell_N$ (Lines 10-4). By assumption in this case (see "Input" part in the algorithm) $R \notin \mathcal{N}_I(N, \boldsymbol{\sigma})$. Thereofre R is a key of that array $\mathcal{R}_I(N, \boldsymbol{\sigma})$ by Specification (7); Write $\mathcal{R}_I(N, \boldsymbol{\sigma})[R] = [C, \boldsymbol{\tau}]$, C is a descendant of R with $\text{depth}(C) \leq \text{depth}(N) = \ell_N = \ell_R = \text{depth}(R)$. Hence, $C = R$. Write $\boldsymbol{\tau} = (\tau_{\ell_N+1})$. By definition in Specification 6 it verifies $\tau_{\ell_N+1} \leq \sigma_{\ell_N+1}$. Moreover $\boldsymbol{\tau} = (\tau_{\ell_N+1}) \in \mathcal{E}(R) = \mathcal{E}(C)$, being the label at $\mathsf{root}(\mathcal{E}(R))$ it is computed at Line 1 of Algo. 1, namely $\tau_{\ell_N+1} = \sum_{L \in \mathsf{Child}(\mathsf{root}(\mathcal{T}_R))} \delta_1(L)$ in the tree $\mathcal{T}_R$ which corresponds to

Algorithm 4: `computePoly` (Computation of polynomials)

```
Input: $N$ node at depth $\ell_N$
$\boldsymbol{\sigma} = (\sigma_{\ell_N+1}, \dots, \sigma_{\ell_R+1}) \in \mathcal{E}_I(N)_{\leq \ell_N - \ell_R}$
$R = P_{\ell_R - \ell_N}(N)$; depth of $R = \ell_R \leq \ell_N$
Assumption: if $\ell_R = \ell_N$ then $R \notin \mathcal{N}_I(N, \boldsymbol{\sigma})$
Output: $h_{R,\boldsymbol{\sigma}} \in I_R$ such that $\textsc{lm}(h_{R,\boldsymbol{\sigma}}) = x_{\ell_N+1}^{\sigma_{\ell_N+1}} \cdots x_{\ell_R+1}^{\sigma_{\ell_R+1}}$
1  if $\ell_R = \ell_N$ then                                   // $R$ and $N$ are at the same depth
2      $p(x_{\ell+1}) \leftarrow \prod_{C \in \mathrm{Child}(R)} t_{\ell_R+1}^{(C)} \bmod \langle \mathbf{t}_{\leq \ell_R}^{(R)} \rangle$
3      $w_{\ell_R+1} \leftarrow \sigma_{\ell_R+1} - \deg_{x_{\ell_R+1}}(p)$
4      return $x_{\ell_R+1}^{w_{\ell_R+1}} \cdot p$
5  $q \leftarrow \prod_{S \in \mathcal{N}_I(N,\boldsymbol{\sigma})} t_{\ell_R+1}^{(S)} \bmod \langle \mathbf{t}_{\leq \ell_R}^{(R)} \rangle$          // $\ell_N > \ell_R$
6  $w_{\ell_R+1} \leftarrow \sigma_{\ell_R+1} - \deg_{x_{\ell_R+1}}(q)$
7  $\chi_{\ell_R+1}^{(R)} \leftarrow x_{\ell_R+1}^{w_{\ell_R+1}} \cdot q$
8  $\mathcal{D} \leftarrow \mathrm{Child}(R) \setminus \mathcal{N}_I(N, \boldsymbol{\sigma})$
9  Compute the family of idempotents $\{e_{\mathcal{D}}(S)\}_{S \in \mathcal{D}}$
10 for $S \in \mathcal{D}$ do
11     $[C, \boldsymbol{\tau}] \leftarrow \mathcal{R}_I(N, \boldsymbol{\sigma})[S]$   // $C$ is a descendant of $S$, write $\boldsymbol{\tau} = (\tau_{\ell_C+1}, \dots, \tau_{\ell_S+1})$
12     $h_{S,\boldsymbol{\tau}} =$ computePoly$(C, \boldsymbol{\tau}, S)$                    // recursive call
13     $\boldsymbol{\sigma}' \leftarrow (\sigma_{\ell_N+1}, \dots, \sigma_{\ell_R+2})$              // last value of $\boldsymbol{\sigma}$ pruned
14     $w_j \leftarrow \sigma_j - \tau_j$ for $\ell_S + 1 = \ell_R + 2 \leq j \leq \ell_C + 1$
15     $h_{S,\boldsymbol{\sigma}'} \leftarrow x_{\ell_N+1}^{\sigma_{\ell_N+1}} \cdots x_{\ell_C+2}^{\sigma_{\ell_C+2}} \cdot x_{\ell_C+1}^{w_{\ell_C+1}} \cdots x_{\ell_R+2}^{w_{\ell_R+2}} \cdot h_{S,\boldsymbol{\tau}}$
16 return $\chi_{\ell_R+1}^{(R)} \left( \sum_{S \in \mathcal{D}} e_{\mathcal{D}}(S) h_{S,\boldsymbol{\sigma}'} \right) \bmod \langle \mathbf{t}_{\leq \ell_R}^{(R)} \rangle$
```

$\sum_{C\in\mathrm{Child}(R)}\deg_{x_{\ell_N+1}}(t^{(C)}_{\ell_N+1})$ viewed in the tree $\mathcal{T}(I)$ via the embedding $\mathcal{T}_R\hookrightarrow\mathcal{T}(I)$ of Prop. 2. It follows that $\tau_{\ell_N+1}=\deg_{x_{\ell_N+1}}(p)$ where p is the polynomial defined at Line 2, and in particular that $\deg_{x_{\ell_N+1}}(p)\le\sigma_{\ell_N+1}$ making the definition of w_{ℓ_N+1} at Line 3 positive. And consequently, $\deg_{x_{\ell_R+1}}(\chi^{(R)}_{\ell_R+1})=\sigma_{\ell_R+1}$ as required.

Besides, Corollary 1 affirms that $I_R\simeq\prod_{S\in\mathrm{Child}(R)}\langle\mathbf{t}^{(S)}_{\le\ell_N+1}\rangle$. By construction at Line 2, p is in the later product of ideals, hence $\chi^{(R)}_{\ell_R+1}$ being a multiple of p (line 4) is in I_R, as required.

Step 2, general case (Line 5 and onward), proof that $h_{R,\boldsymbol{\sigma}'}\in I_R$: Write $h:=\sum_{S\in\mathcal{D}}e_{\mathcal{D}}(S)h_{S,\boldsymbol{\sigma}'}$ and $h_{R,\boldsymbol{\sigma}}\overset{(\circ)}{:=}h\cdot\chi^{(R)}_{\ell_R+1}$ the output polynomial at Line 16.

Let us justify the recursive calls at Line 12 first, for insuring termination and correction. Write $\ell_C:=\mathrm{depth}(C)$. We have $\ell_S\le\ell_C\le\ell_N$ according to Sepcif. 6 ofthe interpolate tree $\mathcal{R}_I(N,\boldsymbol{\sigma})$. Besides note that $\mathrm{depth}(S):=\ell_S=\ell_R+1$. We obtain: $\ell_C-\ell_S=\ell_C-\ell_R-1<\ell_N-\ell_R$; The gap $\ell_C-\ell_S$ in the recursive call is smaller than in the main call, and thus it is valid to try a proof by induction. Eventually the stack of recursive calls reaches inputs for which $\ell_C-\ell_S=0$, reducing to the case treated in Step 1.

By definition (Line 5) q is in $\langle t^{(R)}_1,\ldots,t^{(R)}_{\ell_R},t^{(S)}_{\ell_R+1}\rangle$ for all $S\in\mathcal{N}_I(N,\boldsymbol{\sigma})$. Therefore $q\in\langle\mathbf{t}^{(M)}\rangle$ for all leaves M of $\mathcal{T}$ descendant of an $S\in\mathcal{N}_I(N,\boldsymbol{\sigma})$. Hence, $\chi^{(R)}_{\ell_R+1}$ also, as a multiple of q by Line 7, as well as for $h_{R,\boldsymbol{\sigma}}$, multiple of $\chi^{(R)}_{\ell_R+1}$ by Eq. ($\circ$) just above.

Let $S\in\mathcal{D}$. By definition of idempotents Prop. 3, (3), (5), $h\overset{(*)}{\equiv}c\cdot h_{S,\boldsymbol{\tau}}\bmod\langle\mathbf{t}^{(S)}_{\le\ell_R+1}\rangle$ for a constant $c\in A_S$. By induction hypothesis, $h_{S,\boldsymbol{\tau}}\in I_S\simeq\prod_{M\in\text{ Leaves of }S}\langle\mathbf{t}^{(M')}\rangle$ for all leaves M' of $\mathcal{T}$ descendant of S. Thanks to the congruence $(*)$ above, $h\in\prod_{M\in\text{ Leaves of }S}\langle\mathbf{t}^{(M')}\rangle$ as well.

Therefore, $h_{R,\boldsymbol{\sigma}}\in\langle\mathbf{t}^{(M)}\rangle$ for all leaves M which are descendant of a node in $\mathcal{N}_I(N,\boldsymbol{\sigma})$, thanks to the factor $\chi^{(R)}_{\ell_R+1}$ of $h_{R,\boldsymbol{\sigma}}$. Moreover $h_{R,\boldsymbol{\sigma}}\in\langle\mathbf{t}^{(M')}\rangle$ and for all descendant leaves M' of a node S in $\mathcal{D}$, thanks to the factor h of $h_{R,\boldsymbol{\sigma}}$. Regarding that $\mathcal{D}=\mathrm{Child}(R)\setminus\mathcal{N}_I(N,\boldsymbol{\sigma})$ (Line 8), we obtain that $h\in\prod_{S\in\mathrm{Child}(R)}\langle\mathbf{t}^{(S)}_{\le\ell_R+1}\rangle\simeq I_R$, the latter isomorphism being due to Corollary 1.

Step 3, general case (Line 5 and onward), proof about $\mathrm{LM}(h_{R,\boldsymbol{\sigma}})$: Write $\boldsymbol{\tau}=(\tau_{\ell_C+1},\ldots,\tau_{\ell_R+2})$. By Condition (c.1) of Def. 2 that governs how the tree $\mathcal{N}_I(N,\boldsymbol{\sigma})$ is built, $(0,\ldots,\tau_{\ell_C+1},\ldots,\tau_{\ell_R+2})\le_{mon}(\sigma_{\ell_N+1},\ldots,\sigma_{\ell_R+2})$ therefore there are some integers w_j such that $\tau_j+w_j=\sigma_j$ for $j=\ell_R+2,\ldots,\ell_C+1$. Thus according to Line 15,

$$\mathrm{LM}(x_{\ell_N+1}^{\sigma_{\ell_N+1}}\cdots x_{\ell_C+2}^{\sigma_{\ell_C}+2}\cdot x_{\ell_C+1}^{w_{\ell_C}+1}\cdots x_{\ell_R+2}^{w_{\ell_R}+2}\cdot h_{S,\boldsymbol{\tau}})=x_{\ell_R+2}^{\sigma_{\ell_R+2}}\cdots x_{\ell_N+1}^{\sigma_{\ell_N+1}}.$$

By Lemma 4, we obtain that

$$\sum_{S\in\mathcal{D}}e_{\mathcal{D}}(S)=1\ \Rightarrow\ \mathrm{LM}(\sum_{S\in\mathcal{D}}e_{\mathcal{D}}(S)h_{S,\sigma})=x_{\ell_R+2}^{\sigma_{\ell_R+2}}\cdots x_{\ell_N+1}^{\sigma_{\ell_N+1}}$$

And thus $\mathrm{LM}(h_{R,\boldsymbol{\sigma}})=\mathrm{LM}(\chi^{(R)}_{\ell_R+1})\mathrm{LM}(h)=x_{\ell_R+1}^{\sigma_{\ell_R+1}}\cdot x_{\ell_R+2}^{\sigma_{\ell_R+2}}\cdots x_{\ell_N+1}^{\sigma_{\ell_N+1}}$ as required.

Remark 8. The proof remains valid if:

1/ At Line 4 one replaces $x_{\ell_R+1}^{w_{\ell_R+1}}$ in "$x_{\ell_R+1}^{w_{\ell_R+1}}\cdot p$" by *any* monic polynomial $a(x_{\ell_R+1})\in A_R[x_{\ell_R+1}]$ of degee w_{ℓ_R+1}

2/ Same at Line 7.

3/ At Line 11 another way of filling the interpolate node $\mathcal{R}(N, \boldsymbol{\sigma})$ was done (remember that the array $R(N, \boldsymbol{\sigma})$ depends on some ways of going through the nodes in Algo. 2 "lastExpAssign" and 3 "isDivided").

4/ At Line 15 one replaces $x_{\ell_N+1}^{\sigma_{\ell_N+1}} \cdots x_{\ell_C+2}^{\sigma_{\ell_C}+2} \cdot x_{\ell_C+1}^{w_{\ell_C}+1} \cdots x_{\ell_R+2}^{w_{\ell_R}+2}$ by any polynomial a in $A_S[x_{\ell_R+1}, \ldots, x_{\ell_N+1}]$ such that $\text{LM}(a) = x_{\ell_N+1}^{\sigma_{\ell_N+1}} \cdots x_{\ell_C+2}^{\sigma_{\ell_C}+2} \cdot x_{\ell_C+1}^{w_{\ell_C}+1} \cdots x_{\ell_R+2}^{w_{\ell_R}+2}$.

In other words, the results of Prop. 4 are independent of these three choices. The computed polynomials do though. One particular choice yields the reduced Gröbner basis, see Section 6.

4.3. Structure and specialization property

The proof of the factorization property in Theorem 2 builds upon the following Lemma. It uses the same notation as Algo. 4 "`computePoly`".

Lemma 5. *Write* $h_{R,\boldsymbol{\sigma}} = \texttt{computePoly}(N, \boldsymbol{\sigma}, R)$. *as in Prop. 4 and Algo. 4, and consider the polynomials* $h_{S,\boldsymbol{\tau}} := \texttt{computePoly}(S, \boldsymbol{\tau}, C)$ *computed at Line 12. By induction hypothesis, there are polynomials* $\rho_j^{(S)}$ *such that:*

$$\text{LM}(\rho_j^{(S)}) = x_j^{\tau_j} \ , \quad h_{S,\boldsymbol{\tau}} \equiv \rho_{\ell_R+2}^{(S)} \cdots \rho_{\ell_C+1}^{(S)} \bmod (I_S)_{\leq \ell_N - 1}.$$

As in Line 14, define $\widetilde{\rho}_j^{(S)} = x_j^{w_j} \rho_j^{(S)}$ *if* $\ell_R + 2 \leq j \leq \ell_C + 1$, *or simply* $\widetilde{\rho}_j^{(S)} := x_j^{\sigma_j}$ *if* $\ell_C + 2 \leq j \leq \ell_N + 1$. *Let* $\chi_j^{(R)} \equiv \sum_{S \in \mathcal{D}} e_{\mathcal{D}}(S) \widetilde{\rho}_j^{(S)} \bmod \langle \mathbf{t}_{\leq \ell_N}^{(R)} \rangle$ *for* $\ell_R + 2 \leq j \leq \ell_N + 1$, *and let* $\chi_{\ell_R+1}^{(R)}$ *be as defined in Line 7. We have:*

$$h_{R,\boldsymbol{\sigma}} \equiv \chi_{\ell_R+1}^{(R)} \cdots \chi_{\ell_N+1}^{(R)} \bmod (I_R)_{\leq \ell_N - 1},$$

Proof. Thanks to the property (3)-(5) of Proposition 3 concerning idempotents, it is a simple algebraic verification that for all $S \in \mathcal{D}$, one has:

$$\chi_{\ell_R+2}^{(R)} \cdots \chi_{\ell_N+1}^{(R)} \equiv \widetilde{\rho}_{\ell_R+2}^{(S)} \cdots \widetilde{\rho}_{\ell_N+1}^{(S)} \bmod (I_S)_{\leq \ell_N}. \tag{6}$$

Recall the congruences ($\circ$) and ($*$) in the proof of Prop. 4, rewritten hereafter in the first two congruences. The third one is the statement, satisfied for all $S \in \mathcal{D}$:

$$h_{R,\boldsymbol{\sigma}} \equiv \chi_{\ell_R+1}^{(R)} \cdot h \equiv c \cdot h_{S,\boldsymbol{\tau}} \equiv \chi_{\ell_R+1}^{(R)} \cdot x_{\ell_R+2}^{w_{\ell_R+2}} \rho_{\ell_R+2}^{(S)} \cdots x_{\ell_N+1}^{w_{\ell_N+1}} \rho_{\ell_N+1}^{(S)} \bmod (I_S)_{\leq \ell_N} \tag{7}$$

and thus, by combining Eqs. (6), (7):

$$h_{R,\boldsymbol{\sigma}} \equiv \chi_{\ell_R+1}^{(R)} \cdot \widetilde{\rho}_{\ell_R+2}^{(S)} \cdots \widetilde{\rho}_{\ell_N+1}^{(S)} \bmod (I_S)_{\leq \ell_N}$$

Now both the l.h.s and the r.h.s. of Eq (7) are zero modulo I_S for all $S \in \mathcal{N}_I(N, \boldsymbol{\sigma})$ because the factor $\chi_{\ell_R+1}^{(R)} \in I_S$ by definition. Therefore $h_{R,\boldsymbol{\sigma}}$ is thus equal to the r.h.s. of Eq (7) modulo I_U for all $U \in \text{Child}(R)$ yielding

$$h_{R,\boldsymbol{\sigma}} \equiv \chi_{\ell_R+1}^{(R)} \cdots \chi_{\ell_N+1}^{(R)} \bmod \prod_{U \in \text{Child}(R)} (I_U)_{\leq \ell_N} \quad (\simeq (I_R)_{\leq \ell_N}).$$

The latter isomorphism being due to Corollary 1.

Thanks to Lemma 5 and Prop. 4, the proof of the theorem is easy:

Proof (Proof of Theorem 2). The algorithm 4 with $R = \mathsf{root}$, N preleaf, outputs a polynomial $h_{\mathsf{root},\boldsymbol{\sigma}} := \texttt{computePoly}(N, \boldsymbol{\sigma}, \mathsf{root})$ that verifies all the requirements according to Proposition 4 and Lemma 5. It suffices thus to take $f_{N,\boldsymbol{\sigma}} = h_{\mathsf{root},\boldsymbol{\sigma}}$.

As for the factorization pattern, note that $\ell_N = n-1$ and $\ell_R = 0$ implying $I_R = I$, whence $(I_R)_{\leq \ell_N} = I_{\leq n-1}$. Thus the factorization stated in Prop. 4 with these partiular input rewrites:

$$h_{\mathsf{root},\boldsymbol{\sigma}} \equiv \chi_1^{(\mathsf{root})} \cdots \chi_{n-1}^{(\mathsf{root})} \bmod I_{\leq n-1},$$

which is the factorization stated in the theorem if one takes $\chi_j = \chi_j^{(\mathsf{root})}$.

The specialization property in Eq. (3) is a consequence of the factorization pattern:

Corollary 2. *The specialization property holds for lexGb of ideals verifying Assumption* **(H)***, or, equivalently, the property stated in Eq.* (3) *holds.*

Proof. As mentioned, it suffices to prove Eq. (3), in which only the implication $\Leftarrow$ is not trivial. The notations from therein are reused here. It is possible to assume w.l.o.g. that $g \in G$ not in $G_{\leq n-1}$: if it is not the case, it suffices to take $G_{\leq m}$ instead of G where m is the largest variable occurring in g.

Let $1 \leq \ell \leq n-1$ and $a \in \bar{k}^\ell$. If $\ell = n-1$ then the property is already known thanks to Gianni's theorem [14]. So we can assume w.l.o.g that $\ell \leq n-2$.

Write $C_g := \textsc{lc}_\ell(g)$ so that $g = C_g \cdot x_{\ell+1}^{\sigma_{\ell+1}} \cdots x_n^{\sigma_n} +$ tail terms, and assume that $\phi_a(C_g) = 0$. By Theorem 2, and with the same notations,

$$g \equiv \chi_1 \cdots \chi_n \bmod I_{\leq n-1}. \tag{8}$$

This same theorem gives $\textsc{lm}(\chi_j) = x_j^{\sigma_j}$, yielding $\textsc{lm}(\chi_{\ell+1} \cdots \chi_n) = x_{\ell+1}^{\sigma_{\ell+1}} \cdots x_n^{\sigma_n}$. Hence $C_g \equiv \chi_1 \cdots \chi_\ell \bmod I_{\leq n-1}$. Therefore if $\phi_a(C_g) = 0$ then $\phi_a(\chi_1 \cdots \chi_\ell) \in \phi_a(I_{\leq n-1})$.

On the other hand, $\phi_a(\chi_1 \cdots \chi_\ell) \in \bar{k}$. If $\phi_a(\chi_1 \cdots \chi_\ell) \neq 0$, then $\phi_a(I_{\leq n-1}) = \langle 1 \rangle$, and thus $\phi_a(g) \in \phi_a(I_{\leq n-1})$ necessarily. Now if $\phi_a(\chi_1 \cdots \chi_\ell) = 0$, it also implies that $\phi_a(g) \in \phi_a(I_{\leq n-1})$ by specializing the congruence in Eq. (8) by ϕ_a: $\phi_a(g) \equiv 0 \bmod \phi_a(I_{\leq n-1})$.

Regarding that $\langle G_{\leq n-1} \rangle = I_{\leq n-1}$, in any case we have proved that $\textsc{nf}(\phi_a(g)\,,\, \phi_a(G_{\leq n-1})) = 0$. Moreover by definition of G_a, $\phi_a(G \setminus G_a) = \{0\}$, thereby $\phi_a(G_{\leq n-1}) = \phi_a((G_a)_{\leq n-1})$. Hence $\textsc{nf}(\phi_a(g)\,,\, \phi_a((G_a)_{\leq n-1})) = 0$. But it was assumed that $g \in G \setminus G_{\leq n-1}$, therefore $\phi_a(g) \notin \phi_a(G_{\leq n-1})$. And consequently $\phi_a((G_a)_{\leq n-1}) = \phi_a((G_a \setminus \{g\})_{\leq n-1})$. In conclusion $\textsc{nf}(\phi_a(g)\,,\, \phi_a((G_a \setminus \{g\})_{\leq n-1})) = 0$.

5. Checking the Gröbner property

5.1. Foreword

This section is made of preliminaries.

Given an ideal I, a *standard monomial* for I (and implicitly for the monomial order $\prec_{lex}$) is a monomial $m \notin \langle \textsc{lm}(I) \rangle$. The set of standard monomials $\textsc{Sm}(I)$ verifies:

$$x_1^{\sigma_1} \cdots x_n^{\sigma_n} \in \textsc{Sm}(I) \Rightarrow \ [\forall i \mid \sigma_i > 0, \Rightarrow \ x_1^{\sigma_1} \cdots x_i^{\sigma_i - 1} \cdots x_n^{\sigma_n} \in \textsc{Sm}(I)\ ].$$

A monomial $m := x_1^{\mu_1} \cdots x_n^{\mu_n}$ is a *minimal monomial* for I iff:

$$m \notin \text{Sm}(I), \text{ and } \forall i,\ \mu_i > 0 \ \Rightarrow m/x_i \in \text{Sm}(I).$$

Definition 3. A monomial $m := x_1^{\beta_1} \cdots x_n^{\beta_n}$ is on the *n-border* of a set of standard monomials Sm, denoted $\mathcal{B}o_n(I))$ iff:

$$\mathcal{B}o_n(I) := x_n\text{Sm}(I) \setminus \text{Sm}(I) \quad \text{where} \quad x_n\text{Sm}(I) := \{x_n m \mid m \in \text{Sm}(I)\}.$$

Remark 9. Note that any minimal monomial $x_1^{\mu_1} \cdots x_n^{\mu_n}$ for I such that $\mu_n > 0$ is in the n-border for I.

Theorem 3. *With the above notation, and denoting for $E \subset \mathbb{Z}^n_{\geq 0}$ $\text{mon}(E) = \{x_1^{\varepsilon_1} \cdots x_n^{\varepsilon_n} \mid \boldsymbol{\varepsilon} \in E\}$, we have $\mathcal{B}o_n(I) = \text{mon}(\mathcal{A}\ell\ell_n(I))$ where:*

$$\mathcal{A}\ell\ell_n(I) := \cup_{\textit{preleaves}\, N \in \mathcal{T}(I)} \cup_{\boldsymbol{\sigma} \in \text{Leaves}((\mathcal{E}(N)))} \{x_1^{\sigma_1} \cdots x_n^{\sigma_n}\}.$$

It is clear that this theorem implies Theorem 1. The proof of the theorem is postponed to § 5.3.

Corollary 3. *The set of polynomials*

$$\cup_{\ell=1}^{n} \cup_{N,\ \text{preleaf of } \mathcal{T}(I_{\leq \ell})} \cup_{\boldsymbol{\nu} \text{ leaves of } \mathcal{E}(N)} \{f_{N,\boldsymbol{\nu}}\}$$

is a lexGb of I.

Proof. Define ℓ as the integer such that $f \in I_{\leq \ell}$ but $f \notin I_{\leq \ell - 1}$. The theorem then guarantees the existence of a preleaf Q of $\mathcal{T}(I_{\leq \ell})$, and of a path $\boldsymbol{\nu} = (\nu_\ell, \nu_{\ell-1}, \ldots, \nu_1)$ in the exponent tree $\mathcal{E}_{I_{\leq \ell}}(Q)$ such that $\nu_i \leq d_i(f)$ for $i = \ell, \ldots, 1$. If $f_{Q,\boldsymbol{\nu}}$ denotes a polynomial as constructed in § 4.2 (hence $f_{Q,\boldsymbol{\nu}} \in I_{\leq \ell}$, $\text{LM}(f_{Q,\boldsymbol{\nu}}) = x_1^{\nu_1} \cdots x_\ell^{\nu_\ell}$), this implies that $\text{LM}(f_{Q,\boldsymbol{\nu}}) | \text{LM}(f)$ which characterizes that a set of elements of an ideal I is a Gröbner basis of that ideal.

5.2. Preliminaries on exponent tries

Lemma 6. *Let N be a preleaf of $\mathcal{T} = \mathcal{T}(I)$, $\boldsymbol{\sigma} = (\sigma_n, \ldots, \sigma_1)$ a leaf of the exponent tree $\mathcal{E}(N)$. Write $P = P_{n-\ell-1}(N)$ the ancestor at depth $\ell < n-1$ of N and consider a leaf $\boldsymbol{\tau} = (\tau_{\ell+1}, \ldots, \tau_1)$ of $\mathcal{E}(P)$. Then $\tau_{\ell+1} > \sigma_{\ell+1}$.*

Proof. By definition $\tau_{\ell+1}$ is the value of $\mathsf{root}(\mathcal{E}(P))$ hence is computed at Line 1 of Algo. 1, as follows: when the stack of recursive calls reached the subtree $\mathcal{T}_P$ rooted at P, Line 1 of Algo. 1 rewrites as:

$$\tau_{\ell+1} = \sum_{L \in \text{Child}(P)} \delta_{\ell+1}(L).$$

On the other hand, according to Specification (5), and of Line 15 of Algo. 2,

$$\sigma_{\ell+1} \leq \delta_{\ell+1}(P) - 1 + \sum_{B \in \mathcal{N}(N, (\sigma_n, \ldots, \sigma_{\ell+2}))} \delta_{\ell+1}(B)$$

Now Specification 4 (applied with $\ell = n-1$ and $d = n-\ell-2$) $\mathcal{N}(N, (\sigma_n, \ldots, \sigma_{\ell+2})) \subset \text{sibling}(P_{n-\ell-2}(N)) \subsetneq \text{Child}(P)$. Therefore $\tau_{\ell+1} > \sigma_{\ell+1}$.

Lemma 7. *Let $C \neq D \in \mathrm{Child}(\mathsf{root}(\mathcal{T}))$, and N_C, N_D some preleaves of $\mathcal{T}$ descendant of C and D respectively. Let $\boldsymbol{\sigma} = (\sigma_n, \dots, \sigma_1)$ be a leaf of the exponent tree $\mathcal{E}(N_C)$ and $\boldsymbol{\tau} = (\tau_n, \dots, \tau_1)$ one of the exponent tree $\mathcal{E}(N_D)$. Write $\boldsymbol{\sigma'} = (\sigma_n, \dots, \sigma_2)$ and $\boldsymbol{\tau'} = (\tau_n, \dots, \tau_2)$. If $\boldsymbol{\sigma'} = \boldsymbol{\tau'}$ and if C is popped before D from the* `toProcess` *list, at Line 3 of Algo. 2, then:*

$$\mathcal{N}(N_C, \boldsymbol{\sigma'}) \cup \{C\} \subset \mathcal{N}(N_D, \boldsymbol{\tau'}).$$

(if D is popped first, then it suffices to exchange C with D).

Proof. Let $A \neq C, D$, assuming such a node exists, be in $\mathcal{N}(N_C, \boldsymbol{\sigma'}) \subset \mathrm{Child}(\mathsf{root}(\mathcal{T}))$. According to Lines 13, 14, 17 of Algo. 2 which build the discard trees $\mathcal{N}_I(\,.\,)$, this means that isDivided$(C, \boldsymbol{\sigma'}, n{-}1, A, \mathtt{bool}_{A,C}, \mathcal{T}) ==$ `False`. Let us prove isDivided$(D, \boldsymbol{\tau'}, n{-}1, A, \mathtt{bool}_{A,D}, \mathcal{T}) ==$ `False` (†), which similarly is equivalent to $A \in \mathcal{N}(N_D, \boldsymbol{\tau'})$.

Assume first that $\mathtt{bool}_{A,C} ==$ `True`. Then it is Condition (c.1) of Def. 2 that is not met. This means that there is no node at any depth $\ell \leq n-1$ that is a descendant of A and having an exponent tree carrying an exponent $\boldsymbol{\nu} = (\nu_{\ell+1}, \dots, \nu_2)$ such that $\nu_i \leq \sigma_i$ for all $2 \leq i \leq \ell+1$. Since $\boldsymbol{\tau'} = \boldsymbol{\sigma'}$, neither $\nu_i \leq \tau_i$ is possible for such i. This implies that both Conditions (c.1) and (c.2) are not satsified for the node D, namely: isDivided$(D, \boldsymbol{\tau'}, n-1, A, \mathtt{bool}_{A,D}, \mathcal{T}) ==$ `False`, independently of $\mathtt{bool}_{A,D}$.

Next, assume that $\mathtt{bool}_{A,C} ==$ `False`, that is $A \notin \mathtt{toProcess}_C$ (see Def. 1). Then Condition (c.2) is not met: if there is a descendant N_A of A having an exponent tree $\mathcal{E}(N_A)$ carrying $\boldsymbol{\nu} = (\nu_{\ell_A+1}, \dots, \nu_2)$ sucht that $\nu_i \leq \sigma_i$ for $i = \ell_{A+1}, \dots, 2$ then N_A is a preleaf and $\boldsymbol{\nu'} = \boldsymbol{\sigma'}$. By assumption, $\mathtt{toProcess}_D \subsetneq \mathtt{toProcess}_C$, therefore $\mathtt{bool}_{A,D} ==$ `False` as well. Therefore Condition (c.1) is not met for D. And Condition (c.2) neither for the same reason it is not for the node C. Thus isDivided$(D, \boldsymbol{\sigma'}, n-1, A, \mathtt{False}, \mathcal{T}) ==$ `False`. This achieves the proof of (†) since all cases have been treated. This implies that $\mathcal{N}(C, \boldsymbol{\sigma'}) \setminus \{D\} \subset \mathcal{N}(D, \boldsymbol{\sigma'})$.

Second, let us prove that $D \notin \mathcal{N}(N_C, \boldsymbol{\sigma'})$. This is because isDivided$(C, \boldsymbol{\sigma'}, n{-}1, D, \mathtt{bool}_{D,C}, \mathcal{T}) ==$ `True`$, S, \boldsymbol{\nu}$; for example $S = N_D$, $\boldsymbol{\nu} = \boldsymbol{\tau'}$ works: $D \in \mathtt{toProcess}_C$ by assumption hence $\mathtt{bool}_{D,C} ==$ `True`. Moreover $\boldsymbol{\nu} = \boldsymbol{\tau'} = \boldsymbol{\sigma'}$, and thus Condition (c.1) is verified. We have proved $\mathcal{N}(C, \boldsymbol{\sigma'}) \subset \mathcal{N}(D, \boldsymbol{\sigma'})$.

Last, we claim that $C \in \mathcal{N}(N_D, \boldsymbol{\tau'})$, or equivalently isDivided$(D, \boldsymbol{\tau'}, n-1, C, \mathtt{bool}_{C,D}, \mathcal{T}) ==$ `False`. Since $C \notin \mathtt{toProcess}_D$ by assumption, we have $\mathtt{bool}_{C,D} ==$ `False`, and Condition (c.1) is not fullfilled. But since N_C is a descendent of C whose exponent tree $\mathcal{E}(N_C)$ carries $\boldsymbol{\sigma'} = \boldsymbol{\tau'}$ by assumption, Condition (c.2) is not met neither. Hence, $C \in \mathcal{N}(N_D, \boldsymbol{\tau'})$ and finally the inclusion of the statement of the lemma holds.

Corollary 4. *No two distinct leaves of some exponent trees attached to some preleaves of $\mathcal{T}$ carry the same exponent.*

(Putting it differently: If (ν_1) and (τ_1) are the labels *of distinct* leaves *(but labels themselves may be equal: $\tau_1 = \nu_1$) of $\mathcal{E}(N_C)$ and $\mathcal{E}(N_D)$ for N_D, N_C some preleaves of $\mathcal{T}$, then the path $\boldsymbol{\nu} = (\nu_n, \dots, \nu_1)$ in $\mathcal{E}(N_C)$ and the path $\boldsymbol{\tau} = (\tau_n, \dots, \tau_1)$ in $\mathcal{E}(N_D)$ are not equal.)*

Proof. First Remark 6 affirms that no two leaves in $\mathcal{E}(N)$ have equal paths, therefore w.l.o.g. $\boldsymbol{\tau}$ and $\boldsymbol{\nu}$ need not be in same exponent tree, whence $N_C \neq N_D$.

We proceed by induction on the depth $n \geq 2$ of $\mathcal{T}$ starting with the case $n = 2$. We can assume w.l.o.g. that $\tau_2 = \nu_2$, otherwise it is clear that $\boldsymbol{\tau} \neq \boldsymbol{\nu}$. Morever here N_C and N_D are distinct nodes at depth 1 of $\mathcal{T}$. Assume that N_C was popped first at Line 3 of Algo. 2. Lemma 7 affirms that $\mathcal{N}(N_C, (\tau_2)) \cup \{N_C\} \subset \mathcal{N}(D, (\nu_2))$. It follows,

$$\tau_1 := \sum_{B \in \mathcal{N}(N_C,(\tau_2))} \delta_1(B) \leq \delta_1(N_C) + \sum_{B \in \mathcal{N}(N_D,(\nu_2))} \delta_1(B) := \nu_1 + \delta_1(N).$$

and thus $\tau_1 < \nu_1$ and $\boldsymbol{\tau} = (\tau_2, \tau_1) \neq (\nu_2, \nu_1) = \boldsymbol{\nu}$.

Assume now that $n \geq 3$. Let G the smallest common ancestor of N_C and N_D, say at depth ℓ_G, and write C and D the two distinct children of G that are respectively ancestors of N_C and N_D. Assume that $\ell_G \geq 1$. In the subree $\mathcal{T}_G$ rooted at G, N_C and N_D are preleaves.

Write $\boldsymbol{\tau}' := (\tau_n, \ldots, \tau_{\ell_G+1})$ and $\boldsymbol{\nu}' := (\nu_n, \ldots, \nu_{\ell_G+1})$. Induction hypothesis implies that $\boldsymbol{\tau}' = \boldsymbol{\nu}'$ iff they represent the same node, that is $N_C = N_D$ and $\boldsymbol{\tau}' = \boldsymbol{\nu}'$ represents the same leaf of $\mathcal{E}_{I_G}(N_C)$. This contradicts the assumption, therefore $\boldsymbol{\tau}' \neq \boldsymbol{\nu}'$ henceforth $\boldsymbol{\tau} \neq \boldsymbol{\nu}$ and the conlusion follows in this case.

Remains to consider the case $\ell_G = 0$, that is when $C, D \in \text{Child}(\mathsf{root}(\mathcal{T}))$. Assume w.l.o.g. that C is popped first at Line 3 of Algo. 2. Lemma 7 implies that

$$\mathcal{N}(N_C, \boldsymbol{\tau}') \cup \{C\} \subset \mathcal{N}(N_D, \boldsymbol{\nu}'), \tag{9}$$

and it follows,

$$\tau_1 := \sum_{B \in \mathcal{N}(N_C,(\boldsymbol{\tau}'))} \delta_1(B) \leq \delta_1(N_C) + \sum_{B \in \mathcal{N}(N_D,(\boldsymbol{\nu}'))} \delta_1(B) := \nu_1 + \delta_1(N).$$

Thus $\tau_1 < \nu_1$ and $\boldsymbol{\tau} \neq \boldsymbol{\nu}$.

Proposition 5 (Intermediate Values). *Let $\boldsymbol{\sigma}' := (\sigma_n, \ldots, \sigma_2)$ be an $n-1$-uplet of nonnegative integers. Associate to it:*

$$\Lambda(\boldsymbol{\sigma}') := \{M_1, \ldots, M_r\} := \{M \text{ is a preleaf of } \mathcal{T} \mid \boldsymbol{\sigma}' \in \mathcal{E}(M)_{\leq n-2}\}$$

and write $C_i := P_{n-2}(M_i)$ the ancestor at depth 1 of the preleaf M_i. Define

$$\begin{array}{rcl} a & := & \min\limits_{M \in \Lambda(\boldsymbol{\sigma}')} \{s_1 \mid \boldsymbol{\sigma}_{s_1} := (\sigma_n, \ldots, \sigma_2, s_1) \in \mathcal{E}(M)\} \\ b & := & \max\limits_{M \in \Lambda(\boldsymbol{\sigma}')} \{s_1 \mid \boldsymbol{\sigma}_{s_1} := (\sigma_n, \ldots, \sigma_2, s_1) \in \mathcal{E}(M) \\ a_i & := & \sum\limits_{B \in \mathcal{N}(M_i, \boldsymbol{\sigma})} \delta_1(B) \quad \textit{for } i = 1, \ldots, r. \end{array}$$

Then $a = a_1 m$ $b = a_r + \delta_1(C_r) - 1$ and $a_{i+1} = a_i + \delta_1(C_i)$.

Moreover, for all $a \leq c \leq b$, we can associate in one-one way a couple $(M_c, \boldsymbol{\sigma}_c)$ with $M_c \in \Lambda(\boldsymbol{\sigma}')$ and $\boldsymbol{\sigma}_c \in \mathcal{E}(N_c)$, such that $\boldsymbol{\sigma}_c = (\sigma_n, \ldots, \sigma_2, c)$.

Proof. Step 1, the C_i's are pairwise distinct: Otherwise take two elements of $\Lambda(\boldsymbol{\sigma}')$, say w.l.o.g. M_1, M_2 which yield $C_1 = C_2$. In the tree $\mathcal{T}_{C_1}$ two disctinct preleaves have their exponent tree that carries the same exponent $\boldsymbol{\sigma}'$, which Corollary 4 forbids.

Step 2, assume that $r = |\Lambda(\boldsymbol{\sigma}')| = 1$. In this case $\boldsymbol{\sigma}_a$, $\boldsymbol{\sigma}_b$ belong the unique exponent tree $\mathcal{E}(M_1)$ associated with the unique element $M_1 \in \Lambda(\boldsymbol{\sigma}')$. According to Line 15 of Algo. 2 we have

$b = a + \delta_1(C_1) - 1$. Moreover the children of $\boldsymbol{\sigma'}$ are $[a, \dots, b]$, hence there is a path $\boldsymbol{\sigma}_c$ for all $a \leq c \leq b$.

Step 3, assume $r \geq 2$, and that the numbering of the C_i's is such that C_1 is popped first, then C_2 etc. Write $\texttt{toProcess}_{C_i}$ the status of the list $\texttt{toProcess}$ right after C_i is popped (see Def. 1). We then have $\texttt{toProcess}_{C_{i+1}} \subsetneq \texttt{toProcess}_{C_i}$. We claim that:

$$\mathcal{N}(M_i, \boldsymbol{\sigma'}) \cup \{C_i\} = \mathcal{N}(M_{i+1}, \boldsymbol{\sigma'}). \tag{10}$$

Lemma 7 provides the inclusion $\subset$, it remains to check the other side $\supset$. To this aim, given $A \neq C_i, C_{i+1},\ A \notin \mathcal{N}(M_i, \boldsymbol{\sigma'})$, it suffices to show that $A \notin \mathcal{N}(M_{i+1}, \boldsymbol{\sigma'})$ $(\heartsuit)$. This is acheived in Steps 4-5 hereafter. Note first that since $A \notin \mathcal{N}(M_i, \boldsymbol{\sigma'})$, we have:

$$\text{isDivided}(C_i, \boldsymbol{\sigma'}, n-1, A, \texttt{bool}_{A,C_i}, \mathcal{T}) == \texttt{True}, S, \boldsymbol{\nu} \quad , \tag{11}$$

where $S, \boldsymbol{\nu}$ make Condition (c.1) or (c.2) true.

Step 4, assume first that $A \in \texttt{toProcess}_{C_i}$ that is $\texttt{bool}_{A,C_i}$ $==$ $\texttt{False}$ $(\Diamond)$. Then Condition (c.1) of Def. 2 is verified, namely

$$(\nu_{\ell_S+1}, \dots, \nu_2) \leq_{mon} (\sigma_{\ell_S+1}, \dots, \sigma_2) \tag{12}$$

where ℓ_S is the depth of S.

Step 4.1: if $A \in \texttt{toProcess}_{C_{i+1}}$, then clearly Condition (c.1) is verfied by C_{i+1} since isDivided$(C_{i+1}, \boldsymbol{\sigma'}, n-1, A, \texttt{bool}_{A,C_{i+1}}, \mathcal{T}) == \texttt{True}, S, \boldsymbol{\nu}$ thanks to Eqs. (11), (12). Hence $A \notin \mathcal{N}(M_{i+1}, \boldsymbol{\sigma'})$.

Step 4.2: if $A \notin \texttt{toProcess}_{C_{i+1}}$ $(\ddagger)$ then Condition (c.2) only must be checked, since (c.1) is not true. Under (c.2) and because of Eq. (11) which gives an S descendant of A carrying an exponent $\boldsymbol{\nu}$ such that Eq. (12) holds, "isDivided" returns $\texttt{False}$ $(\bowtie)$ iff there is a node M_A descendant of A whose exponent tree carries an exponent $\boldsymbol{\tau} = \boldsymbol{\sigma'}$. If this was true, then $M_A \in \Lambda(\boldsymbol{\sigma'})$, say $A = C_j$ for $j \neq i, i+1$ since $A \neq C_i, C_{i+1}$. According to the numbering of elements of $\Lambda(\boldsymbol{\sigma'})$, either $j < i$, in which case $\texttt{toProcess}_{C_j} \supsetneq \texttt{toProcess}_{C_i} \supsetneq \texttt{toProcess}_{C_{i+1}}$, or $j > i+1$, in which case $\texttt{toProcess}_{C_i} \supsetneq \texttt{toProcess}_{C_{i+1}} \supsetneq \texttt{toProcess}_{C_j}$. In the latter case, $A = C_j \in \texttt{toProcess}_{C_{i+1}}$, contradiction with $(\ddagger)$. In the former case, $A = C_j \notin \texttt{toProcess}_{C_i}$ a contradiction with $(\Diamond)$. Therefore $(\bowtie)$ is not true and thus isDivided$(C_{i+1}, \boldsymbol{\sigma'}, n-1, A, \texttt{bool}_{A,C_{i+1}}, \mathcal{T}) \neq \texttt{False}$.

Step 5, assume next that $A \notin \texttt{toProcess}_{C_i}$. Then $A \notin \texttt{toProcess}_{C_{i+1}}$ neither. Therefore Eq. (11) and isDivided$(C_{i+1}, \boldsymbol{\sigma'}, n-1, A, \texttt{False}, \mathcal{T})$ returns the same boolean, namley $\texttt{True}$. This concludes the proof of $(\heartsuit)$, and thus the proof of Eq. (10).

Step 6, from Eq. (10) we obtain $a_{i+1} = a_i + \delta_1(C_i)$ for $i = 1, \dots, r-1$. Moreover by definition $a_1 = \sum_{B \in \mathcal{N}(M_1, \boldsymbol{\sigma'})} \delta_1(N)$ is therefore the minimal value a in the statement. And $a_r + \delta_{C_r} - 1 = b$ from the same argument of Step 1.

Corollary 5. *Let $\boldsymbol{\sigma'}$ be as in Proposition 5. We refer to the notations $\Lambda(\boldsymbol{\sigma'})$, a and M_a used therein. Let also $\boldsymbol{\tau'} = (\tau_n, \dots, \tau_2)$ be a preleaf of an exponent tree $\mathcal{E}(N)$, where N is a preleaf of $\mathcal{T}(I)$. Assume that $\boldsymbol{\tau'} <_{mon} \boldsymbol{\sigma'}$. Then $\tau_1 \geq a$.*

Proof. Write $C_a := P_{n-2}(M_a)$ the ancestor at depth 1 of M_a, and similary write C the one for M. We prove that

$$\mathcal{N}(M_a, \boldsymbol{\sigma'}) \subset \mathcal{N}(M, \boldsymbol{\tau'}). \tag{13}$$

Since $a = \sum_{B \in \mathcal{N}(M_a, \boldsymbol{\sigma'})} \delta_1(B)$ and $\tau_1 = \sum_{B \in \mathcal{N}(M, \boldsymbol{\tau'})} \delta_1(B)$, Eq. (13) is indeed enough to conclude $\tau_1 \geq a$.

Let $A \neq C, C_a$ *not in* $\mathcal{N}(M, \boldsymbol{\tau'})$. Whether Conditions (c.1) or (c.2) is verified, there is node N_A descendant of A say at depth $1 \leq \ell_A \leq n-1$, and an exponent $\boldsymbol{\nu'} = (\nu_{\ell_A+1}, \ldots, \nu_2)$ being a preleaf in the exponent tree $\mathcal{E}(N_A)$, such that $\nu_i \leq \tau_i$ for $2 \leq i \leq \ell_A + 1$.

$$\text{isDivided}(C, \boldsymbol{\tau'}, n-1, A, \texttt{bool}_{A,C}, \mathcal{T}) == \texttt{True}, N_A, \boldsymbol{\nu'}. \tag{14}$$

By assumption, $\tau_i \leq \sigma_i$, consequently Condition (c.1) is verified for C_a if $A \in \texttt{toProcess}_{C_a}$. Hence isDivided$(C_a, \boldsymbol{\sigma'}, n-1, A, \texttt{True}, \mathcal{T}) == \texttt{True}, N_A, \boldsymbol{\nu'}$

Next if $A \notin \texttt{toProcess}_{C_a}$, we must look at Condition (c.2). It requires that no preleaf descendant of A has an exponent tree that carries $\boldsymbol{\sigma'}$ as a preleaf. Assume the contrary with M_A such a preleaf of $\mathcal{T}$ descendant of A. But then $M_A \in \Lambda(\boldsymbol{\sigma'})$, and thus $A \in\in P_{n-2}(\Lambda(\boldsymbol{\sigma'}))$. On the other hand, because a is minimal $\texttt{toProcess}_{C_a} \subsetneqq \texttt{toProcess}_A$, see Eq. (10) in the proof of Prop. 5. Therefore the preleaf N_A descendant of A makes Condition (c.2) true for C_a, that is isDivided$(C_a, \boldsymbol{\nu'}, n-1, A, \texttt{False}, \mathcal{T}) == \texttt{True}, N_A, \ldots$. This contradicts Eq. (14)

$A \notin \texttt{toProcess}_{C_i}$ for all $C_i := P_{n-2}(\Lambda(\boldsymbol{\sigma'}))$ (see proof of Prop. 5). Contradiction with $A \notin \texttt{toProcess}_C$. Thus Condition (c.2) is verified when $A \notin \texttt{toProcess}_C$. Hence in any cases $A \notin \mathcal{N}(N_a, \boldsymbol{\sigma'})$. We have proved $\mathcal{N}(M_a, \boldsymbol{\sigma'}) \setminus \{C\} \subset \mathcal{N}(M, \boldsymbol{\tau'})$.

If $C = C_a$ then since $C_a \notin \mathcal{N}(M_a, \boldsymbol{\sigma'})$ we have $\mathcal{N}(M_a, \boldsymbol{\sigma'}) \subset \mathcal{N}(M, \boldsymbol{\tau'})$ and Eq. (13) is proved.

If $C_a \neq C$, we claim that $C \notin \mathcal{N}(M_a, \boldsymbol{\sigma'})$, or equivalently isDivided$(C_a, n-1, \boldsymbol{\sigma'}, C, \texttt{bool}_{C_a,C}, \mathcal{T}) == \texttt{True}, S, \boldsymbol{\nu'}$ $(\diamondsuit)$, for some S descendant of C and $\boldsymbol{\nu'}$ a preleaf in its exponent tree $\mathcal{E}(S)$. If one takes $S = M$ and $\boldsymbol{\nu'} = \boldsymbol{\tau'}$, then $\boldsymbol{\tau'} <_{mon} \boldsymbol{\sigma'}$. Hence Condition (C.1) is verified, when $C \in \texttt{toProcess}_{C_a}$ (that is $\texttt{bool}_{C_a,C} == \texttt{True}$). Otherwise, Condition (c.2) requires additionnally that no preleaf descendant of C has an exponent tree that carries $\boldsymbol{\sigma'}$ as a preleaf. As seen above, $C \notin \texttt{toProcess}_{C_a}$ implies that $C \notin P_{n-2}(\Lambda(\boldsymbol{\sigma'}))$. Thus C cannot have a preleaf to whcih is attached an expeonent tree that carries $\boldsymbol{\sigma'}$. Thus Condition (c.2) is verfied, and finally $(\diamondsuit)$ is proved. This case achieves the proof of Eq. 13, hence of the Corollary.

5.3. Standard monomials and leading exponents

Now that prelimiaries are settled, this part focuses on the proof of Theorem 3. According to Remark 9, it implies Theorem 1. It proceeds by induction on the depth n of the tree $\mathcal{T}(I)$, that is the number of variables. It follows the plan hereunder. Note that Points 3, 4, 5 are definitions. Points 1, 9, 10 are statements allowing a short proof written directly here. Point 2 states the induction hypothesis. Finally the proofs of Points 6, 7, 8 are given thereafter.

1. Proof of the base case. When $n = 1$, $\mathcal{B}o_n(I) = \{x_1^{\sigma_1}\}$ where $\sigma_1 = \sum_{L \in \text{Child}(\mathsf{root}(\mathcal{T}(I)))} \delta_1(L)$. The prim-dec tree $\mathcal{T}(I)$ has depth 1, and $\mathsf{root}(\mathcal{T}(I))$ is the only preleaf. The exponent tree $\mathcal{E}(\mathsf{root})$ has depth 0, the only node is its root whose value is computed at Line 1 of Algo. 1 equal to $s = \sum_{L \in \text{Child}(\mathsf{root}(\mathcal{T}(I)))} \delta_1(L)$. $\sigma_1 = s$ implies $\text{mon}(\mathcal{A}\ell\ell_1(I)) = \{x_1^s\} = \mathcal{B}o_1(I)$.
2. Induction hypothesis supplies: $\text{mon}(\mathcal{A}\ell\ell_n(I_L)) = \mathcal{B}o_n(I_L)$ for all $L \in \text{Child}(\mathsf{root})$.
3. For all $m = x_2^{\mu_2} \cdots x_n^{\mu_n} \in \text{S}\textsc{m}(I_L)$ write $m^+ := x_2^{\mu_2} \cdots x_{n-1}^{\mu_{n-1}} \cdot x_n^{\mu_n^+}$ the unique monomial in $\mathcal{B}o_n(I_L)$, $\mu_n^+ > \mu_n$.
4. By induction hypothesis 2, $\boldsymbol{\mu}^+ := (\mu_n^+, \ldots, \mu_2) \in \mathcal{A}\ell\ell_n(I_L)$. Since the unions are disjoint in Corollary 3 denote by $N(L, m)$ the unique preleaf of $\mathcal{T}(I_L)$ such that $\boldsymbol{\mu}^+ \in \mathcal{E}_{I_L}(N(L, m))$.
5. Define $a(L, m) := \sum_{B \in \mathcal{N}(N(L,m), \boldsymbol{\mu}^+)} \delta_1(B)$ and for each L the maps:

$$\begin{array}{rcl} \phi_L : \text{S}\textsc{m}(I_L) & \rightarrow & \{\text{set of monomials in } x_1, \ldots, x_n\} \\ m & \mapsto & \{x_1^{a(L,m)} \cdot m, \ldots, x_1^{a(L,m)+\delta_1(L)-1} \cdot m\} \end{array}$$

6. Define $\mathcal{SM} := \cup_{L\in \mathrm{Child}(\mathsf{root}(\mathcal{T}))}\phi_L(\mathrm{Sm}(I_L))$. The union is disjoint.
7. $|\mathcal{SM}| = |\mathrm{Sm}(I)|$. (also equal to $\dim_k(k[x_1,\dots,x_n]/I)$ the dimension of the k-vector space)
8. $x_n\mathcal{SM}\setminus\mathcal{SM} = \mathrm{mon}(\mathcal{A}\ell\ell_n(I))$.
9. We deduce that $\mathrm{Sm}(I) = \mathcal{SM}$. *Proof:* Indeed, for each exponent $\boldsymbol{\sigma}$ in $\mathcal{A}\ell\ell_n(I)$, Theorem 2 constructs a polynomial $f_{\boldsymbol{\sigma}} \in I$ with $\mathrm{LM}(f) = x_1^{\sigma_1}\cdots x_n^{\sigma_n}$. Thus $\mathrm{mon}(\mathcal{A}\ell\ell_n(I)) \subset \langle \mathrm{LM}(I)\rangle$. Since $\mathcal{SM}\cap\mathrm{mon}(\mathcal{A}\ell\ell_n(I)) = \emptyset$ by 8, it follows that $\mathcal{SM}\subset\mathrm{Sm}(I) = \{\text{monomials } \notin \langle\mathrm{LM}(I)\rangle\}$. And thanks to 7, that $\mathcal{SM} = \mathrm{Sm}(I)$.
10. Finally, $\mathcal{B}o_n(I) := x_n\mathrm{Sm}(I)\setminus\mathrm{Sm}(I) = x_n\mathcal{SM}\setminus\mathcal{SM} := \mathrm{mon}(\mathcal{A}\ell\ell_n(I))$, achieving the induction.

Proof (Proof of 6). Assume that two monomials $m\in\mathrm{Sm}(I_L)$ and $p\in\mathrm{Sm}(I_K)$ for some $L,K\in$ Child(root), verify $m\neq p$. Then clearly $\phi_L(m)\cap\phi_K(p)=\emptyset$.

Using the notation of 3, write $m^+ = x_2^{\mu_2}\cdots x_{n-1}^{\mu_{n-1}}\cdot x_n^{\mu_n^+}$ and $p^+ = x_2^{\pi_2}\cdots x_{n-1}^{\pi_{n-1}}\cdot x_n^{\pi_n^+}$, the monomials in $\mathcal{B}o_n(I_L)$ and $\mathcal{B}o_n(I_K)$ respectively. Indeed, using the notation $N(L,m)$ and $N(K,p)$ of 4 deduced from the induction hypothesis, one has $\boldsymbol{\mu}^+\in\mathcal{E}_{I_L}(N(L,m))\hookrightarrow\mathcal{E}_I(N(L,m))$ and $\boldsymbol{\pi}^+\in\mathcal{E}_{I_N}(N(K,p))\hookrightarrow\mathcal{E}_I(N(K,p))$.

If $m^+=p^+$, one can consider that $K\neq L$. Otherwise if $K=L$, in the subtree $\mathcal{T}_L$ rooted at $K=L$, $\boldsymbol{\mu}^+=(\mu_n^+,\mu_{n-1},\dots,\mu_2)$ and $\boldsymbol{\pi}^+=(\pi_n^+,\pi_{n-1},\dots,\pi_2)$ are equal leaves of the exponent trees $\mathcal{E}_{I_L}(N(L,m))$ and $\mathcal{E}_{I_L}(N(L,p))$ contradicting Corollary 4.

Assume thus $m^+=p^+$ and $K\neq L$. Then Prop. 5 (applied with $\Lambda(\boldsymbol{\mu}^+)=\Lambda(\boldsymbol{\pi}^+)$) implies that $\mu_1\neq\pi_1$ for any children $\boldsymbol{\mu}=(\mu_n^+,\dots,\mu_1)$ of the preleaf $\boldsymbol{\mu}^+$ in the exponent tree $\mathcal{E}_I(N(L,m))$, and any children $\boldsymbol{\pi}=(\pi_n^+,\dots,\pi_1)$ of the preleaf $\boldsymbol{\pi}^+$ in the exponent tree $\mathcal{E}_I(N(K,p))$. W.l.o.g, we can assume that $\mu_1<\pi_1$ which implies according to Eq. (10) that $\mathcal{N}(N(L,m),\boldsymbol{\mu}^+)\cup\{L\}\subset\mathcal{N}(N(K,p),\boldsymbol{\pi}^+)$. In particular $a(L,m)\le a(K,p)+\delta_1(L)$ with the notation 5.

Therefore the largest monomial $x_1^{a(L,m)+\delta_1(L)-1}\cdot m$ in $\phi_L(m)$ is strictly smaller than the smaller exponent $x_1^{a(K,p)+\delta_1(L)-1}\cdot p$ in $\phi_K(p)$. Hence $\phi_L(m)\cap\phi_K(p)=\emptyset$.

Proof (Proof of 7). In Point 5 the union of the $\phi_L(\mathrm{Sm}(I_L))$ is disjoint, therefore $|\mathcal{SM}| = \sum_L|\phi_L(\mathrm{Sm}(I_L))|$. By construction, $|\phi_L(\mathrm{Sm}(I_L))| = \delta_1(L)|\mathrm{Sm}(I_L)|$. Hence, $|\mathcal{SM}| = \sum_L\delta_1(L)|\mathrm{Sm}(I_L)|$.

On the other hand, $I_L\simeq\prod_{M,\text{preleaf of }\mathcal{T}_L}\langle\mathbf{t}^{(M)}\otimes A_L\rangle$, where the product is irredundant (see Prop. 1). Thus, $R/I\simeq\prod_L A_L[x_2,\dots,x_n]/I_L$, and

$$\dim_k(R/I)=\sum_L\dim_k(A_L/I_L)=\sum_L\dim_k(A_L)\dim_{A_L}(A/I_L)=\sum_L\dim_k(k[x_1]/\langle t_1^{(L)}\rangle)|\mathcal{SM}(I_L)|.$$

Since $\dim_k(k[x_1]/\langle t_1^{(L)}\rangle)=\delta_1(L)$, we obtain $|\mathrm{Sm}(I)|=\dim_k(R/I)=\sum_L\delta_1(L)\mathrm{Sm}(I_L)$. Hence $|\mathrm{Sm}|=|\mathcal{SM}|$.

Proof (Proof of 8). Let $x_nm\in x_n\mathcal{SM}\setminus\mathcal{SM}$. By definition in 6, let L be the unique child of $\mathsf{root}(\mathcal{T})$ for which $\phi_L(\mathcal{SM}(I_L))$ contains m, and let $m_L\in\mathcal{SM}(I_L)$ be the unique monomial for which $m\in\phi_L(m_L)$. By definition of the map ϕ_L in 5 We have $m=x_1^{\mu_1}\cdot m_L$ where $a(L,m_L)\le\mu_1\le a(L,m_L)+\delta_1(L)-1$.

Assume that $x_nm_L\in\mathrm{Sm}(I_L)$, then $\phi_L(x_n\cdot m_L)=x_n\phi_L(m)$, and then $x_nx_1^{\mu_1}m_L=x_nm\in\mathcal{SM}$. Contradiction, therefore $x_nm_L\in x_n\mathrm{Sm}(I_L)\setminus\mathrm{Sm}(I_L)=\mathcal{B}o_n(I_L)$. By induction hypothesis 2, $x_nm_L\in\mathrm{mon}(\mathcal{A}\ell\ell_n(I_L))$. There is a unique preleaf N of $\mathcal{T}(I_L)$ and a unique exponent $\boldsymbol{\nu}\in\mathcal{E}_{I_L}(N)$ such that $x_2^{\nu_2}\cdots x_n^{\nu_n}=x_nm_L$. Note then that we have $x_nm=x_1^{\mu_1}x_2^{\nu_2}\cdots x_n^{\nu_n}$. Besides, we have $N=N(L,m_L)$ by definition 4. Therefore, by definition of $a(L,m_L)$, the labels of the children of the node $\boldsymbol{\nu} \in \mathcal{E}_{I_L}(N) \hookrightarrow \mathcal{E}_I(N)$ are $\{a(L, m_L), \ldots, a(L, m_L) + \delta_1(L) - 1\}$. An finally $(\nu_n, \ldots, \nu_2, \mu_1) \in \mathcal{E}_I(N)$. In particular, $x_n m \in \mathcal{A}\ell\ell_n(I)$, and $x_n \mathcal{SM} \setminus \mathcal{SM} \subset \mathcal{A}\ell\ell_n(I)$.

Next, consider $\boldsymbol{\sigma} = (\sigma_n, \ldots, \sigma_1) \in \mathcal{E}(N) \subset \mathcal{A}\ell\ell_n(I)$, N a preleaf of $\mathcal{T}(I)$. Write $\boldsymbol{\sigma'} = (\sigma_n, \ldots, \sigma_2) \in \mathcal{E}_{I_L}(N) \subset \mathcal{A}\ell\ell_n(I_L)$. Recall the notation $\Lambda(\boldsymbol{\sigma'})$, a, and b of Prop. 5. In particular $a \leq \sigma_1 \leq b$ and $N \in \Lambda(\boldsymbol{\sigma'})$. For all $C \in P_{n-2}(\Lambda(\boldsymbol{\sigma'}))$, induction hypothesis implies that $\mathrm{mon}(\mathcal{A}\ell\ell_n(I_C)) = \mathcal{B}o_n(I_C)$. Thus $m_C = x_2^{\sigma_2} \cdots x_n^{\sigma_n} \in x_n \mathrm{SM}(I_c) \setminus \mathrm{SM}(I_C)$. Note that $N(C, m_C) \in \Lambda(\boldsymbol{\sigma'})$. In particular By Prop. 5, Step 4 one has the following disjoint union

$$\{a, \ldots, b\} = \cup_{C \in \P_{n-2}(\Lambda(\boldsymbol{\sigma'}))} \{a(C, m_C), \ldots, a(C, m_C) + \delta_1(C)\}.$$

It implies that there is a unique C_{σ_1} above such that $a(C_{\sigma_1}, m_{C_{\sigma_1}}) \leq \sigma_1 \leq a(C_{\sigma_1}, m_{C_{\sigma_1}}) + \delta_1(C_{\sigma_1})$. And from the definiton of the map $\phi_{C_{\sigma_1}}$, $x_1^{\sigma_1} \cdots x_n^{\sigma_n} \in x_n \phi_{C_{\sigma_1}}(m_C) \subset x_n \phi_{C_{\sigma_1}}(\mathcal{SM}(I_{C_{\sigma_1}})) \subset x_n \mathcal{SM}$

6. Concluding Remarks

Specialization property. Corollary 2 was restricted to zero-dimensional ideals verifying Hypothesis **(H)**, when primary ideals have a lexGb are in purely triangular shape. It is very safe to think that this specialization property holds for a larger class of zero-dimensional primary ideals. Using the notation of Proposition 1, let h be a polynomial distinct from g_{i,s_i} in the lexGb of a general primary ideal.

$$h = p_1^{\ell_1} \cdots p_n^{\ell_n} + \text{ tail terms}$$

It is easy to see that the leading term of h may vanish after specialization at a point $(\alpha_1, \ldots, \alpha_\ell)$, while tail terms may not:

$$\mathscr{G} = \{x^2\ ,\ y^2 + x\ ,\ xyz + y\ ,\ z^2\}.$$

This is a primary ideal of radical $\langle x, y, z\rangle$. Write ϕ_0 the specialization map $x = 0$ from $k[x, y, z]$ to $k[y, z]$. Then $\phi_0(xyz + y) = y$, and $\mathrm{NF}(y\ ,\ \phi_0(\mathscr{G} \setminus G_0))) = \mathrm{NF}(y\ ,\ [y^2, z^2]) = y$ is not zero hence does not verify the specialization property of Corollary 2. On the other hand if one considers the primary ideal:

$$\mathscr{G}' = \{x^2\ ,\ y^2 + x\ ,\ xyz + (2y + 1)x\ ,\ z^2\},$$

then $\phi_0(xyz + yx) = 0$ hence Corollary 2 holds.

This is (likely) precisely what prevents the specialization to hold in full generality (see other counter-examples in [14, 19, 15, 5]); the most general possible result is thus probably:

specialization property holds for I if and only if it holds for each of the primary ideals of I.

A proof requires to understand how to recombine the lexGb's of these primary ideals into a lexGb of I. The data structure of tree used in the article does not generalize straightforwardly to such non-triangular lexGb.

Wrap-up algorithm. For sake of completeness pieces built so far are put together in Algo. 5 below.

Reduced lexGb. As already mentioned the algorithm does not compute the reduced lexGb in general. The reason why is in the definition of $\chi_\ell^{(R')}$ at Lines 7, 15 during recursive calls in the computation of Algo 4. Instead of x_ℓ^w, *any* degree w polynomial would do. In the case of ideal of points, Lederer [8] computes the reduced lexGb without normal forms. Instead of x^w, polynomials of "high" degree with only monomials of low degree with prescribed coefficients are constructed on-demand, in order to cancel monomials of too-high degree. This on-demand specification, and the number of such polynomials required make it difficult to analyze the cost of construction.

```
Algorithm 5: Final algorithm
  Input: triangular sets of the primary ideals of $I$
  Output: A minimal lexGb of $I$
1 Build the prim-dec tree $\mathcal{T}$
2 assignExp($\mathcal{T}$)                         // build exponent trees
3 $\mathscr{G} \leftarrow \emptyset$
4 for $\ell = 0$ to $n - 1$ do                                            // top-down
5     minPreleaves ← extract the nodes $N \in \mathcal{T}_{\leq \ell}$ and minimal exponents $\boldsymbol{\sigma}$ in $\mathcal{E}_{I_{\leq \ell}}(N)$ such
        that $\boldsymbol{\sigma}$ is minimal for $\leq_{mon}$
6     for $(N, \sigma) \in$ minPreleaves do
7         $\mathscr{G} \leftarrow \mathscr{G} \cup \{$computePoly$(N, \boldsymbol{\sigma}, \text{root})\}$
8 return $\mathscr{G}$
```

In this work instead we give a simple optimization that still does not yield the reduced one, but reduces furthermore the number of monomials. If one really needs the reduced lexGb, then taking normal from computations are less expensive.

[1] M. Kalkbrener, On the stability of Gröbner bases under specialization, J. Symbolic Comput. 24 (2) (1997) 51–58.

[2] L. Cerlienco, M. Mureddu, From algebraic sets to monomial linear bases by means of combinatorial algorithms, Discrete Mathematics 139 (13) (1995) 73 – 87.

[3] M. G. Marinari, T. Mora, A remark on a remark by Macaulay or enhancing Lazard structural theorem, Bull. Iranian Math. Soc. 29 (1) (2003) 1–45, 85.

[4] L. Cerlienco, M. Mureddu, Algoritmi combinatori per l'interpolazione polinomiale in dimensione ≥ 2, in: Actes du 24 me Séminaire Lotharingien, Vol. 461/S-24, Publ. I.R.M.A. Strasbourg, 1993, pp. 39–76.

[5] M. G. Marinari, T. Mora, Cerlienco-Mureddu correspondence and Lazard structural theorem, Investigaciones Mathematicas 27 (2006) 155–178.

[6] B. Felszeghy, B. Ráth, L. Rónyai, The lex game and some applications, J. of Symbolic Comput. 41 (6) (2006) 663 – 681.

[7] S. Lundqvist, Vector space bases associated to vanishing ideals of points, Journal of Pure and Applied Algebra 214 (4) (2010) 309–321.

[8] M. Lederer, The vanishing ideal of a finite set of closed points in affine space, J. of Pure and Applied Algebra 212 (2008) 1116–1133.

[9] N. Lei, X. Zheng, Y. Ren, The vanishing ideal of a finite set of points with multiplicity structures, in: Y. S. R. Feng, W.-L. Lee (Ed.), Computer Mathematics, 9th and 10th Asian Symposium on Computer Mathematics, Springer, 2014, pp. 275–296.

[10] J. Abbott, M. Kreuzer, L. Robbiano, Computing zero-dimensional schemes, Journal of Symbolic Computation 39 (1) (2005) 31–49.

[11] S. Gao, V. Rodrigues, J. Stroomer, Gröbner basis structure of finite sets of points, http://www.math.clemson.edu/~sgao/pub.html, preprint (16 pages) (2003).

[12] X. Dahan, Gcd modulo a primary triangular set of dimension zero, in: Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC '17, ACM, New York, NY, USA, 2017, pp. 109–116. pathdoi:10.1145/3087604.3087612.
URL `http://doi.acm.org/10.1145/3087604.3087612`

[13] D. Lazard, Ideal bases and primary decomposition: case of two variables, J. Symbolic Comput. 1 (3) (1985) 261–270.

[14] P. Gianni, Properties of Gröbner bases under specialization, in: J. Davenport (Ed.), In Proc. of EUROCAL'87, Lecture Notes in Computer Science (378), Springer, Berlin, 1987, pp. 293–297.

[15] T. Becker, Gröbner bases versus D-Gröbner bases, and Gröbner bases under specialization, Applicable Algebra in Engineering , Communications and Computing 5 (1994) 1–8.

[16] X. Dahan, M. Moreno Maza, É. Schost, W. Wu, Y. Xie, Lifting techniques for triangular decompositions, in: ISSAC'05, ACM, 2005, pp. 108–115.

[17] P. Gianni, B. Trager, G. Zacharias, Gröbner bases and primary decomposition of polynomial ideals, J. of Symbolic Comput. 6 (1988) 149–167.

[18] X. Dahan, On the bit-size of non-radical triangular sets, in: MACIS 2017, Vienna, Austria, November 15-17, 2017, Proceedings, Springer International Publishing, Cham, 2017, pp. 264–269.
URL `arXiv:1710.06396`

[19] M. El Kahoui, S. Rakrak, Structure of gröbner bases with respect to block orders, Mathematics of Computation 76 (260) (2007) 2181–2187.