Degroebnerization and its applications: a new approach for data modelling.

Michela Ceria
(joint work with T.Mora and A.Visconti)
Polytechnic of Bari

MEGA2021, June 2021

What we want to do?

We study a **new Computer Algebra-based approach** to give a polynomial model for data.

Specific problem

Reverse engineering for gene regulatory networks.

We want to find: polynomials in normal form with respect to the vanishing ideal of some given data points.

We achieve:

- no Gröbner bases Buchberger reduction: only linear algebra and combinatorics;
- complexity improvement.

Let's start! Reference for the problem

Previous paper

Laubenbacher, R., Stigler, B., *A computational algebra approach to the reverse engineering of gene regulatory networks*, Journal of theoretical biology, 229, 4, 523-537, Academic Press.

We will improve the method proposed in this paper.

Reverse engineering: the previous approach

Study of the DNA of mosquitos, by means of Computer Algebra.

Quoting Lauenbacher and Stigler

The authors *adopt the modeling framework of time-discrete multi-state dynamical systems*.

The proposed model

The system is composed by **states** (that are our **points**) whose *nodes* (the *variables*) represent the genes.

The states of each node, so the values each variable can take are the elements in $\mathbb{Z}_p$, with p prime.

The dynamical system is represented by the function

$$F : (\mathbb{Z}_p)^n \to (\mathbb{Z}_p)^n,$$

sending each point to the next one, so representing the transitions among the states.

The values of the prime p indicate *how that state changes at each transition*; for instance for $p = 3$, -1 means decreases, 0 means stability and $+1$ means increases.

More precisely, if $\mathbf{X} = \{P_1, ..., P_N\} \subseteq (\mathbb{Z}_p)^n$ are our points, it holds $F(P_i) = P_{i+1}$, for $i = 1, ..., N-1$.

We can describe F via the coordinate functions

$$f_j : (\mathbb{Z}_p)^n \to \mathbb{Z}_p, \quad 1 \leq j \leq n$$

so that $F(P_i) = (f_1(P_i), ..., f_n(P_i))$, for $i = 1, ..., N-1$.

They see these functions as *polynomial functions*

$$f_i \in \mathbb{Z}_p[x_1, ..., x_n].$$

How to solve the problem according to Lauenbacher and Stigler

They compute separately the functions f_j, $1 \leq j \leq n$, from which they reconstruct F.

In particular, the polynomials they want to find should be **in normal form**, namely in reduced form modulo the ideal $I(\mathbf{X})$.

Normal form

Let I be an ideal of $\mathcal{P} := \mathbf{k}[x_1, \ldots, x_n]$.
If $f \in \mathcal{P}$ is a polynomial, we call *normal form* of f with respect to I the unique polynomial

$$\mathrm{Nf}(f) := \sum_{t \in A} c_t t,$$

such that A is a basis of $\mathcal{P}/I$ as $\mathbf{k}$-vector space and $f - \mathrm{Nf}(f) \in I$.

The algorithm

The proposed algorithm is composed of three steps:

- compute a **particular solution** for f_j, $1 \leq j \leq n$;
- compute a **Gröbner basis** $\mathcal{G}$ of $I(\mathbf{X})$, via Buchberger-Möller algorithm, which interpolates on $\mathbf{X}$;
- perform **Buchberger reduction** on the particular solutions, modulo $\mathcal{G}$.

Focus on the particular solution: separator polynomials

Let us consider a finite set of simple points $\mathbf{X} := \{P_1, ..., P_N\} \subset \mathbf{k}^n$. A family of **separator polynomials** for $\mathbf{X}$ is a set $\mathcal{Q} := \{Q_1, .., Q_N\} \subset \mathcal{P}$, such that, for $i, j \in \{1, ..., N\}$, $Q_i(P_j) = 1$ if $i = j$ and $Q_i(P_j) = 0$ otherwise.

Denote $P_i = (a_{1i}, ..., a_{ni})$, $1 \leq i \leq N$; we can define, for $1 \leq i, j \leq N$,

$$c_{i,j} := \begin{cases} 0, & \text{if } i = j \\ \min\{h : 1 \leq h \leq n,\ a_{hi} \neq a_{hj}\} & \text{otherwise} \end{cases}$$

The building factors for separator polynomials are

$$p_{i,j}^{[c_{i,j}]} = \frac{x_{c_{i,j}} - a_{c_{i,j},j}}{a_{c_{i,j},i} - a_{c_{i,j},j}}$$

The particular solution is defined as

$$\sum_{i=1}^{N-1} a_{j,i+1} r_i(x_1, \ldots, x_n),$$

where $r_i(x_1, \ldots, x_n) = \prod_{k=1}^{N-1} p_{ik}^{[c_{i,k}]}$.
The polynomials r_i form a separator family for our points.

As they say...

In summary, the complexity of the algorithm is

$$O(n^2N^2)+O((N^3+N)(\log(p))^2+N^2n^2)+O(n(N-1))2^{cN+N-1}).$$

It is *quadratic* in the number n of *variables* and **exponential** in the number N of time **points**.

The turning point: Degroebnerization!

Why you should not even think...

Groebner bases are not efficient to be computed, therefore **Degroebnerization** is aimed to **limit only to the very necessary cases the use of Groebner bases**, finding **alternative solutions** every time it is possible.

The alternative tools come from **linear algebra** and **combinatorics**.

Lundvist's formula

Let $\mathbf{X} = \{P_1, ..., P_N\}$ be a finite set of simple points, $I := I(\mathbf{X}) \lhd \mathbf{k}[x_1, ..., x_n]$ the ideal of points and $\mathsf{N} = \{t_1, ..., t_N\} \subset \mathbf{k}[x_1, ..., x_n]$ such that $[\mathsf{N}] = \{[t_1], ..., [t_N]\}$ is a basis for $A := \mathbf{k}[x_1, ..., x_n]/I$. Then, for each $f \in \mathbf{k}[x_1, ..., x_n]$ we have

$$\mathrm{Nf}(f, \mathsf{N}) = (t_1, ..., t_N)(\mathsf{N}[\mathbf{X}]^{-1})^t (f(P_1), ..., f(P_N))^t,$$

where $\mathsf{N}[\mathbf{X}]$ is the matrix whose rows are the evaluations of N at the elements of $\mathbf{X}$ and $\mathrm{Nf}(f, \mathsf{N})$ is the normal form of f w.r.t. $I(\mathbf{X})$.

On Lundvist's formula

As soon as we have $\mathbf{X}$ and a basis of the quotient algebra $\mathcal{P}/I(\mathbf{X})$, computing the normal form of a polynomial modulo $I(\mathbf{X})$ is only a matter of **evaluations** and **linear algebra**.

In principle, we would need...

- the **particular solutions**, i.e. the polynomials of which we have to compute the normal form;
- the set of **points** $\mathbf{X}$;
- a **basis** of the **quotient algebra** $\mathcal{P}/I(\mathbf{X})$.

But we can even do better!

Reminder

To apply Lundvist's formula we need only the **evaluation** of the polynomial whose normal form is needed.

Reminder 2

The particular solutions are the f_i's for $i = 1, \ldots, N-1$; they describe the system since we have $F(P_i) = (f_1(P_i), \ldots, f_n(P_i))$, for $i = 1, \ldots, N-1$. Moreover, we know that $F(P_i) = P_{i+1}$, for $i = 1, \ldots, N-1$.

But we can even do better!

Since only the evaluation is needed and we know that $F(P_i) = P_{i+1}$, for $i = 1, \ldots, N-1$, **we can completely avoid the computation of the particular solution!**

Therefore...

...given **X** we only have to

- compute a **basis** of the **quotient algebra** $\mathcal{P}/I(\mathbf{X})$.
- apply **Lundvist's formula**.

A BASIS OF THE QUOTIENT ALGEBRA $\mathcal{P}/I(\mathbf{X})$

Given any ideal I of $\mathcal{P}$, we can define the set
$\mathsf{T}\{I\} := \{\mathsf{T}(g),\, g \in I\}$ and the semigroup ideal of leading terms
$\mathsf{T}(I) := \{t\mathsf{T}(g) |\, t \in \mathcal{T}, g \in I\}$.
$\mathsf{N}(I) := \mathcal{T} \setminus \mathsf{T}(I)$ is called **Gröbner escalier** of the ideal I.

If one has a Gröbner basis for $I(\mathbf{X})$ with respect to some term ordering, finding a basis for $\mathcal{P}/I(\mathbf{X})$ is trivial, since one such a basis is the Gröbner escalier of $I(\mathbf{X})$.
But computing a Gröbner basis is not easy at all, therefore we want to avoid it.

A BASIS OF THE QUOTIENT ALGEBRA $\mathcal{P}/I(\mathbf{X})$

There are some algorithms that employ only combinatorics on the given points in $\mathbf{X}$ to compute the Gröbner escalier, with **no polynomials involved**.

- **Cerlienco-Mureddu**: $O(n^2N^2)$
- **Lex Game**: $O(nN + N\min(N, nr))$
- **Iterative Lex Game**: $O(N^2 n \log(N))$

N points, n variables, r max. number of points with the first coordinates in common

The point trie

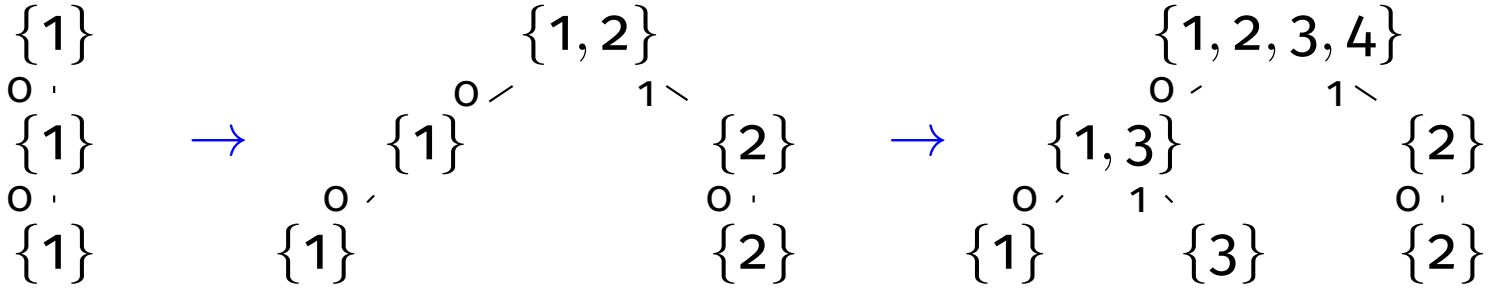
It is a trie representing the reciprocal relations among the coordinates of points.

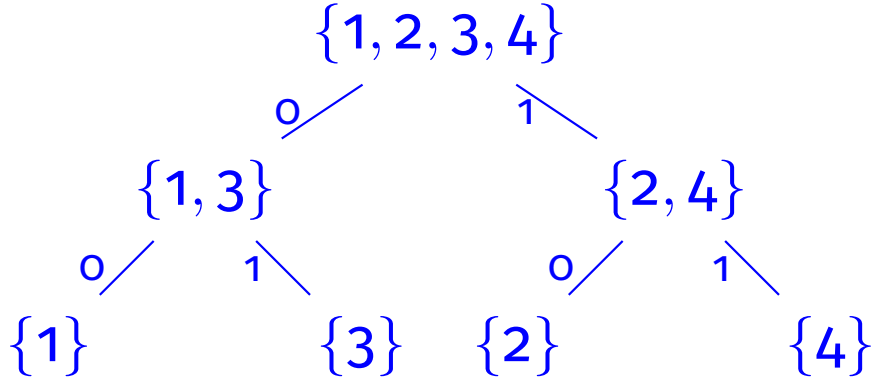
same path from level 0 to level i = same $1, \ldots, i$ first coordinates

It is constructed **iteratively** on the points.

Example

$\mathbf{X} := \{P_1 = (0,0), P_2 = (1,0), P_3 = (0,1), P_4 = (1,1)\}$





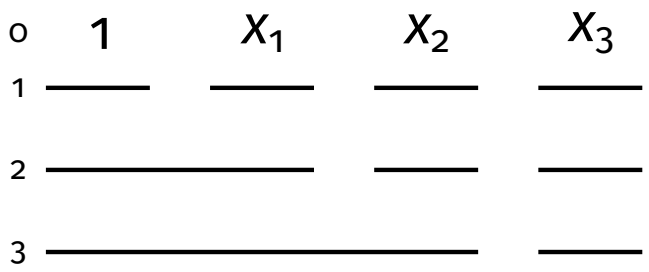
Bar Codes

Definition

A Bar Code B is a picture composed by segments, called *bars*, superimposed in horizontal rows, which satisfies

A. $\forall i, j, 1 \leq i \leq n-1, 1 \leq j \leq \mu(i), \exists! \bar{j} \in \{1, \ldots, \mu(i+1)\}$ s.t. $B_{\bar{j}}^{(i+1)}$ lies under $B_j^{(i)}$

B. $\forall i_1, i_2 \in \{1, \ldots, n\}, \sum_{j_1=1}^{\mu(i_1)} l_1(B_{j_1}^{(i_1)}) = \sum_{j_2=1}^{\mu(i_2)} l_1(B_{j_2}^{(i_2)})$; we will then say that *all the rows have the same length*.



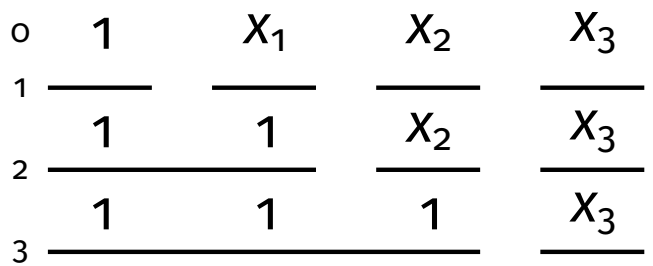
Associating monomials to bars

For $t = x_1^{\gamma_1} \cdots x_n^{\gamma_n} \in \mathcal{T}$, $\forall i \in \{1, \ldots, n\}$, $\pi^i(t) := x_i^{\gamma_i} \cdots x_n^{\gamma_n}$;

$M = \{t_1, \ldots, t_m\} \subset \mathcal{T}$, $M^{[i]} := \pi^i(M)$, $\overline{M}$, $\overline{M}^{[i]}$ increasingly ordered w.r.t. Lex.

$$\mathcal{M} := \begin{pmatrix} \pi^1(t_1) & \ldots & \pi^1(t_m) \\ \pi^2(t_1) & \ldots & \pi^2(t_m) \\ \vdots & & \vdots \\ \pi^n(t_1) & \ldots & \pi^n(t_m) \end{pmatrix}$$

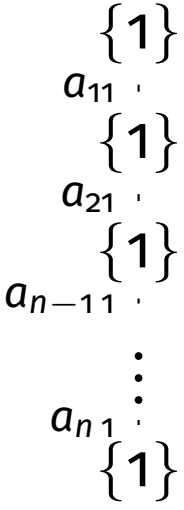
Bar Code: connecting with a bar the repeated terms

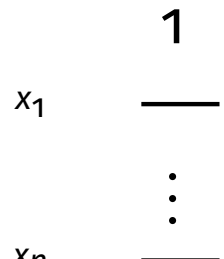


Our algorithm

Base step

$|\mathbf{X}| = N = 1$: set $\mathsf{N}(1) = \{1\}$ and construct the point trie $T(P_1) = \mathfrak{T}(\mathbf{X})$ and the Bar Code $\mathsf{B}(1)$ displayed below. The output is stored in the matrix M.





$$M = \begin{bmatrix} & \mathbf{x_n} & \mathbf{x_{n-1}} & \dots & \mathbf{x_1} \\ & \downarrow & \downarrow & \dots & \downarrow \\ \mathbf{1} \rightarrow & 0 & 0 & \dots & 0 \end{bmatrix}$$

Our algorithm: $|\mathbf{X}| = N > 1$

- update the point trie: forking level $s = \sigma$**-value**; leftmost label of the rightmost sibling $l = \sigma$**-antecedent**;
- find the $s-$bar of t_l: $\mathrm{B}_j^{(s)}$

Information on t_N:

- it lies over $\mathrm{B}_1^{(n)}, \mathrm{B}_1^{(n-1)}, \ldots, \mathrm{B}_1^{(s+1)}$ so t_N lies over the first $n, \ldots, s+1$ bars, i.e. $a_{s+1}^{(N)} = \ldots = a_n^{(N)} = 0$, so $x_n, \ldots, x_{s+1} \nmid t_N$;
- it should lie over $\mathrm{B}_{j+1}^{(s)}$: $a_s^{(N)} = a_s^{(l)} + 1$.

Our algorithm: $|\mathbf{X}| = N > 1$

We test whether $B_{j+1}^{(s)}$ lies over $B_1^{(n)}, B_1^{(n-1)}, \ldots, B_1^{(s+1)}$; two possible cases

A. **NO**: we construct a new s-bar of lenght one over $B_1^{(n)}, B_1^{(n-1)}, \ldots, B_1^{(s+1)}$, on the right of $B_j^{(s)}$, we label it as $B_{j+1}^{(s)}$ and we construct a $1, \ldots, s-1$ bar of length 1 over $B_{j+1}^{(s)}$: $t_N = x_s^{j+2}$; store the output in the N-th row of M.

B. **YES**: we must continue, repeating the procedure

Our algorithm: $|\mathbf{X}| = N > 1$

- *restrict* the point trie *to the points* whose corresponding terms lie over $\mathsf{B}^{(s)}_{j+1}$. The set containing these points is denoted by S and is *obtained reading* $\mathsf{B}^{(s)}_{j+1}$. More precisely, $S = \psi(\mathsf{B}^{(s)}_{j+1})$, where

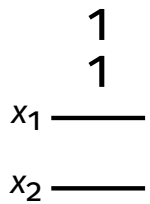
$$\psi : \mathsf{B} \to \mathcal{T}$$

  is the function sending each 1-bar $\mathsf{B}^{(1)}_{l}$ in the term t_l over it and, inductively, for $1 < u \leq n$, $\psi(\mathsf{B}^{(u)}_{h}) = \bigcup_{B \text{ over } \mathsf{B}^{(u)}_{h}} \psi(B)$
- *read P_N's path*, from level $s-1$ to level 1, *looking for the first forking level w.r.t. S* (σ-value/σ-antecedent as before).
- **repeat** the test

The procedure is repeated until we get to the 1-bars or if in the decision step we get case a.

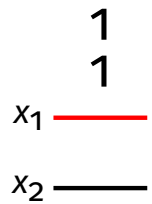
Example

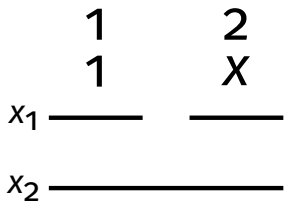
$\mathbf{X} := \{P_1 = (0,0), P_2 = (1,0), P_3 = (0,1), P_4 = (1,1)\}$;
lexicographical Gröbner escalier, $x < y$.



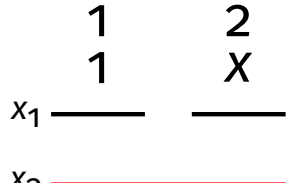
We start with $t_1 = 1$, associated to the first point, P_1.

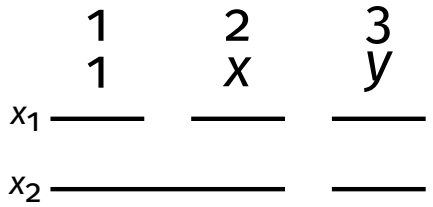
For $P_2 = (1,0)$, we have $f = 1$ and $l = 1$; $t_2 = x$:



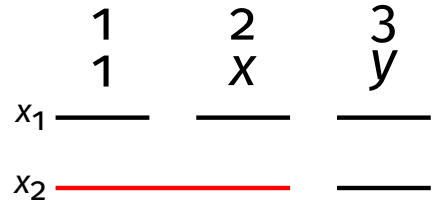


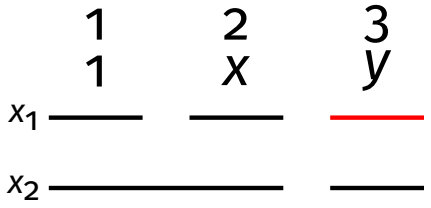
$\mathbf{X} := \{P_1 = (0,0), P_2 = (1,0), P_3 = (0,1), P_4 = (1,1)\}$; lexicographical Gröbner escalier, $x < y$. Continue...

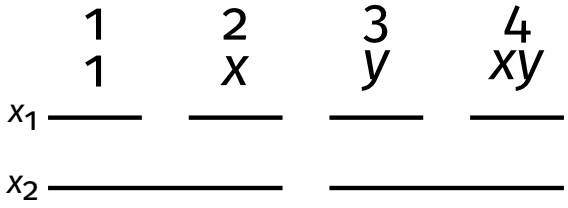




For $P_3 = (0,1)$, we have $f = 2$ and $l = 1$; $t_3 = y$.





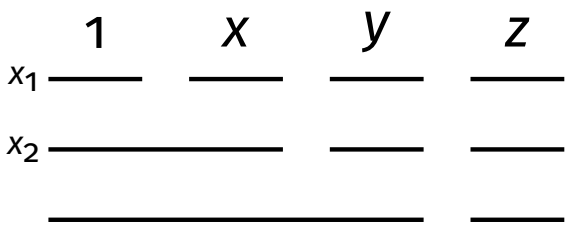


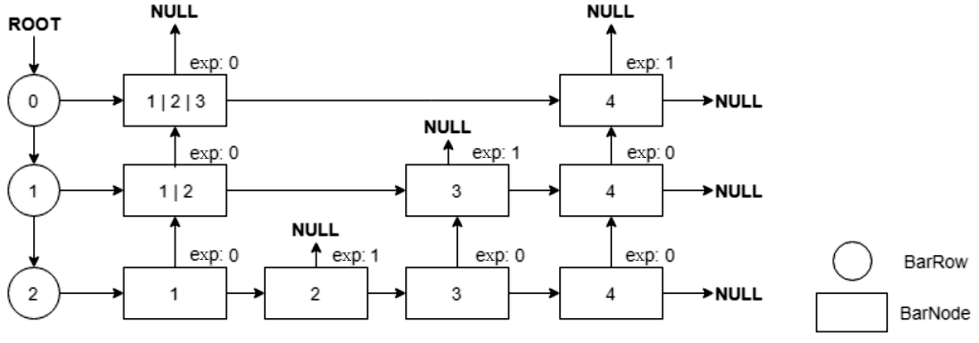
Finally, for $P_4 = (1,1)$, $f = 2$ and $l = 2$; this time there is a bar on the right of the red one. We then restrict to $\{P_3, P_4\}$, so we get the new forking level to be $f = 1$ and the new antecedent to be P_3. This time we see that there is no bar on the right, so we construct it and we get $t_4 = xy$.

Implementation

A Bar Code can be implemented in C using concatenated objects, implemented using a linked list of data structures. We need three different lists, namely one containing the terms, one related to the single bars and one containing the different levels of the Bar Code.

Bar Code referring to the set of terms $\{1, x, y, z\}$:





Testing

Tesing activity has been executed on a machine with an Intel(R) Core(TM) i7-8550U processor 64-bit architecture, CPU @ 1.80GHz equipped with 8 GB of RAM. The operating system installed on this machine was Linux Mint 19 (Tara) 64 bits.

$GF(2^m)$	Points	Coord	Singular	Iter LG
2^4	256	4	1.68s	0.13s
2^4	256	4	5.01s	0.11s
2^5	1024	3	16.77s	0.13s
2^6	4096	3	5m 27.04s	2.18s
2^5	1024	3	23.06s	0.18s
2^6	4096	3	6m 9.29s	2.04s
2^8	65536	3	27h 42m	49m 27s
2^{20}	4096	2	-	37.40s

Complexity of the whole procedure

- Lauenbacher and Stigler's complexity:

$$O(n^2N^2)+O((N^3+N)(\log(p))^2+N^2n^2)+O(n(N-1)2^{cN+N-1}).$$

- Lauenbacher-Stigler's procedure can be refined: complexity can become $O(N^2 nlog(N))+O(n^2N^3(\log(p))^2)$ within the FGLM complexity $O(n^2N^3(\log(p))^2)$

With Degroebnerization:

- no Buchberger reduction;
- only: **Iterative Lex Game** + **Lundvist's formula**:

$$O(nN^2\log(N))+O(nN^{O(1)}(\log(p))^2).$$

We reduce to **degrobnerization complexity** $O(nN^3(\log(p))^2)$.

Thank you
for your attention!