Computational Algebra with Attention: Transformer Oracles for Border Basis Algorithms

Hiroshi Kera*[123] Nico Pelleriti*[3] Yuki Ishihara[4] Max Zimmer[3] Sebastian Pokutta[3]

[1]Chiba University [2]National Institute of Informatics [3]Zuse Institute Berlin [4]Nihon University

ZIB ZUSE INSTITUTE BERLIN

Berlin Mathematics Research Center MATH+

Motivation

Problem: Solving polynomial systems is fundamental but hard.

- Worst-case complexity exponential in # variables
- Border/Gröbner bases are the standard tool
- Most computation is **wasted** on unsuccessful reductions

Our Solution: Train a Transformer oracle to predict which reductions will succeed $\rightarrow$ **up to 3.5$\times$ speedup** with correctness guarantees.

Border Basis Algorithm (BBA)

Core Idea: Expand, then do Gaussian elimination.

Key Concepts:

- $\mathscr{L} = \{x^\alpha : \|\alpha\|_1 \le d\}$ — computational universe (monomials up to degree d)
- $\mathscr{V} \subseteq \mathrm{span}(\mathscr{L})$ — polynomial set with **pairwise distinct leading terms**
- $\mathscr{V}^+ = \{x_j v \mid v \in \mathscr{V}, j = 1, \ldots, n\}$ — expansion candidates

Algorithm 1: L-Stable Span in BBA and OBBA

Input: Polynomials $\mathscr{V}_0$, universe $\mathscr{L}$

$i \leftarrow 0$;

while *true* **do**

  $\mathscr{C}_i \leftarrow \mathscr{V}_i^+$;

  $\mathscr{C}_i \leftarrow \mathrm{Oracle}(\mathscr{L}, \mathscr{V}_i)$;

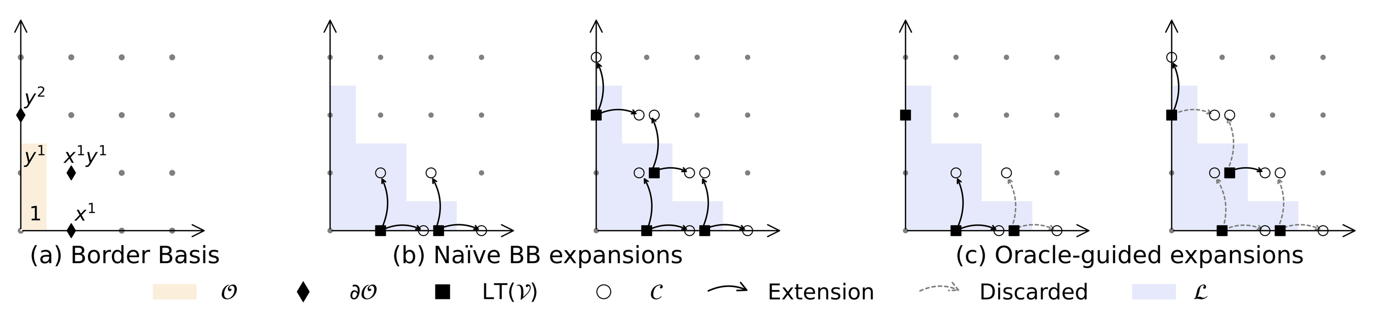
  $\mathscr{V}_{i+1} \leftarrow \mathrm{BasisExtension}(\mathscr{V}_i, \mathscr{C}_i, \mathscr{L})$;

  if $\mathscr{V}_{i+1} = \mathscr{V}_i$ **then**

    break;

  $i \leftarrow i+1$;

Example: How BBA Works



Input: $\mathscr{V} = \{x-1,\ x^2+y^2-1\}$, $\mathscr{L} = \{1, x, y, x^2, xy, y^2\}$

Step 1: Expand — Multiply each $v \in \mathscr{V}$ by each variable:

$$\mathscr{V}^+ = \{x(x-1), y(x-1), x(x^2+y^2-1), y(x^2+y^2-1)\}$$

Step 2: Reduce (Gaussian elimination) — After reduction mod $\mathscr{V}$:

- $x(x-1) = x^2 - x \xrightarrow{\text{reduce}} y^2 + x - 1$ ✓extends $\mathscr{V}$
- $y(x-1) = xy - y$ ✓extends $\mathscr{V}$
- $x(x^2+y^2-1), y(x^2+y^2-1)$ × vanish (wasted!)

Insight: BBA discovers useful candidates only in **hindsight**!

The Transformer Oracle

Task: Given current state $(\mathscr{L}, \mathscr{V})$, predict which expansions extend the basis.

$$\mathrm{Oracle} : (\mathscr{L}, \mathscr{V}) \mapsto \mathscr{S} \subset \{x_1, \ldots, x_n\} \times \mathscr{V}$$

Output: Set of pairs $\mathscr{S} = \{(x_\ell, v_m)\}$ where:

- $x_\ell \in \{x_1, \ldots, x_n\}$ is a **variable** (expansion direction)
- $v_m = \mathrm{LT}(p_m)$ is a **leading term** identifying polynomial $p_m \in \mathscr{V}$
- Each pair specifies candidate $x_\ell \cdot p_m \in \mathscr{V}^+$ to reduce

Architecture: Standard encoder-decoder Transformer, 6 layers, 8 heads.

Training: Supervised learning from BBA execution traces—record which expansions actually extended the basis at each iteration.

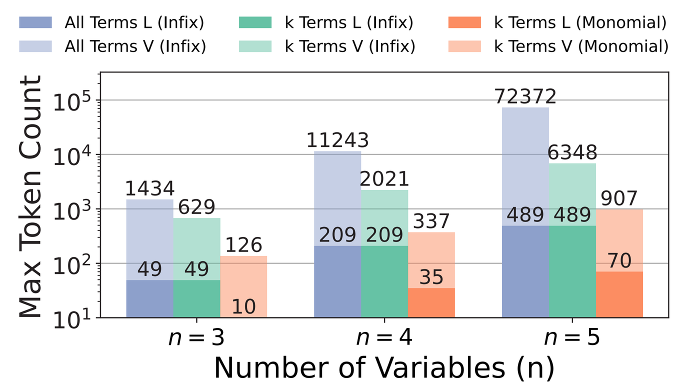
Correctness Guarantee: After k oracle calls, fall back to full BBA expansion $\rightarrow$ algorithm always terminates with correct output.

Efficient Monomial Embedding

Challenge: Polynomial sequences have **tens of thousands of tokens.**

Standard infix: $L = \{1, x, y\}$, $V = [x+2, y]$ becomes:

(C1,E0,E0,<sep>,C1,E1,E0,<sep>,C1,E0,E1,<supsep>, C1,E1,E0,+,C2,E0,E0,<sep>,C1,E0,E1,<eos>)

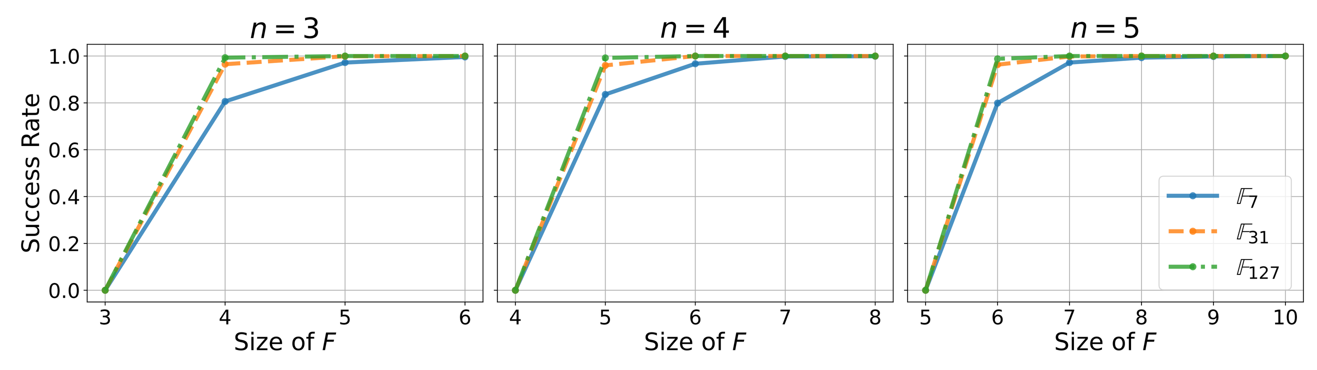


Ours: Embed each monomial as a **single token**:

$$\varphi_{\mathrm{m}}(t, \texttt{<*>}) = \varphi_{\mathrm{c}}(c) + \varphi_{\mathrm{e}}(\boldsymbol{a}) + \varphi_{\mathrm{f}}(\texttt{<*>})$$

Benefits: Tokens $\downarrow \mathscr{O}(n)$, memory $\downarrow \mathscr{O}(n^2)$

Dataset Generation



Success rate $\ge 1 - d'/p$

Challenge: Randomly sampling polynomials usually produces systems that do *not* have the necessary structure (i.e. generating a zero dimensional ideal).

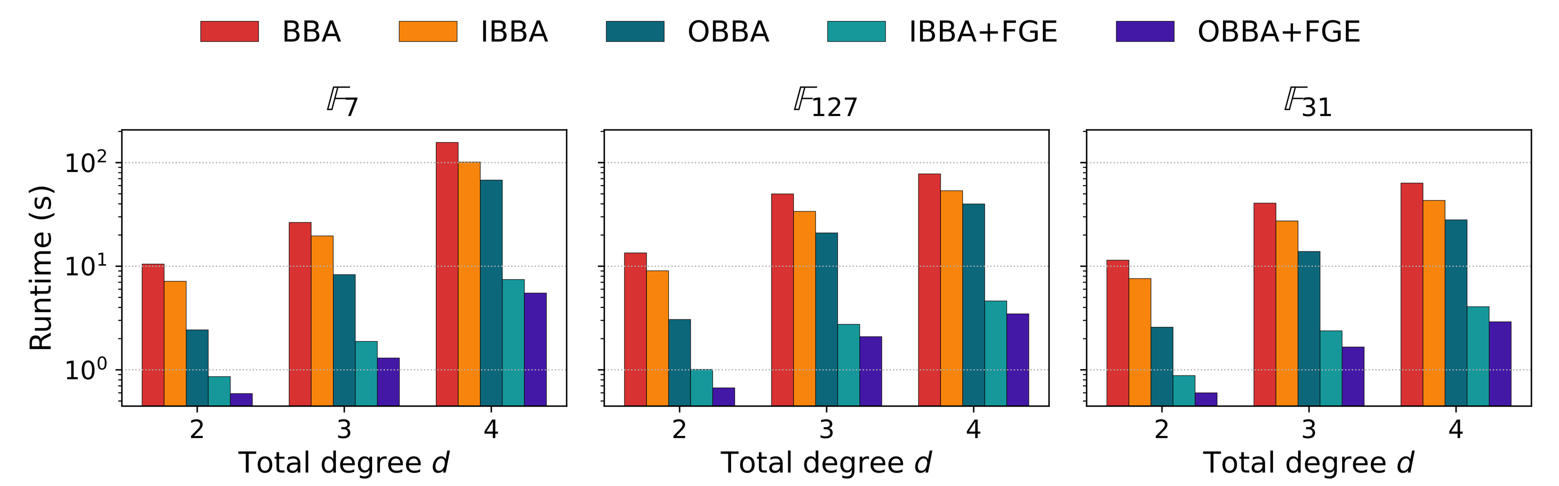
Solution: Sample border bases, transform backwards.

1. Sample order ideal $\mathscr{O}$, construct border basis G
2. Ideal-invariant transform: $F = AG$, $|F| > n$
3. Run BBA on F, collect pairs from last 5 expansions

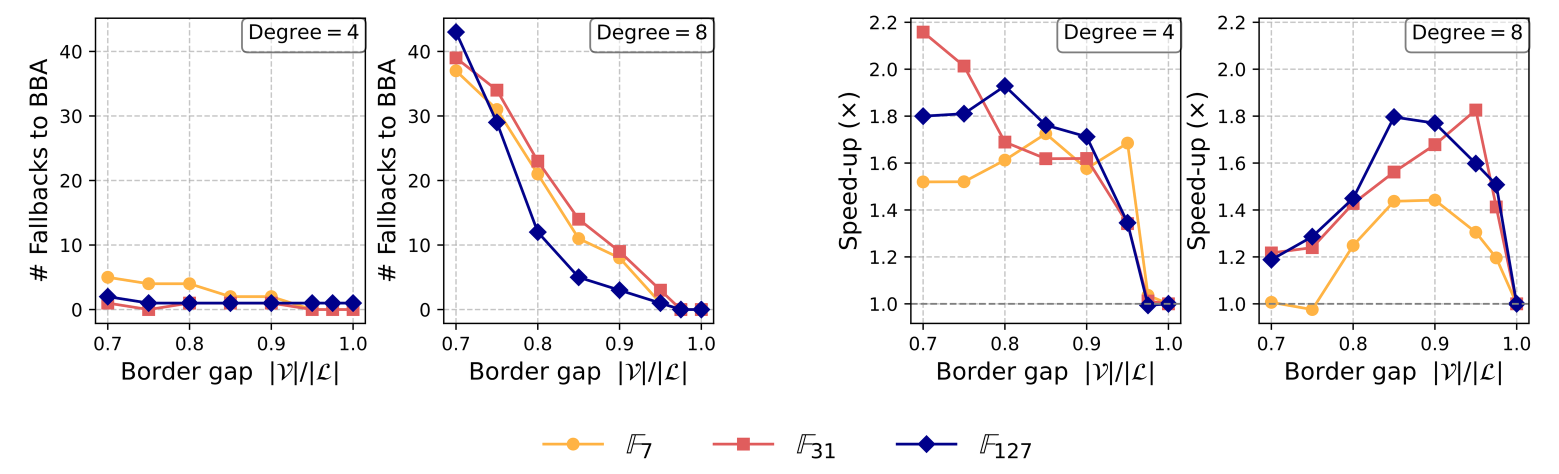
Result: 1M diverse samples per dataset.

Strong Out-of-Distribution Generalization

Key result: Oracle trained on simple cases generalizes to **much harder** problems.



Note: Y-axis is **log scale**. Degree-4 systems are **10–100$\times$ harder** than training data!



Speedup vs. relative border gap $\frac{|\mathcal{V}|}{|\mathcal{L}|}$ for $n = 4$. Invoking oracle earlier $\rightarrow$ more speedup but risk of fallback.

OOD Results: $n = 5$ Variables ($\mathbb{F}_{31}$)

Training: degree ≤ 2 $\rightarrow$ **Testing:** degree ≤ 4 (OOD)

	Baseline		Ours		
Deg	BBA	IBBA	OBBA	IBBA+FGE	OBBA+FGE
2	11.4s	7.6s	2.6s	0.88s	**0.60s**
3	40.7s	27.4s	13.9s	2.4s	**1.7s**
4	136.7s	97.8s	68.8s	7.4s	**5.6s**

Highlighted rows: Out-of-distribution instances (not seen in training).

Degree-4 is 14$\times$ harder than degree-2

Yet OBBA+FGE still achieves **17$\times$ speedup** over baseline!

Conclusion and Outlook

Summary:

- **First** deep-learning border basis algorithm with guarantees
- Diverse border basis sampling plus efficient $\mathscr{O}(n)$ monomial embedding
- Up to 3.5$\times$ speedup, strong OOD ability

Application: Sum-of-Squares (SOS) Programming (`arXiv:2510.13444`)

Next: Larger n, infinite fields, positive-dim. ideals

Code: `github.com/HiroshiKERA/OracleBorderBasis`