Certified CAS-assisted Polynomial Reasoning in Lean 4

Hao Shen

(This is a joint work with Junyu Guo, Junqi Liu and Lihong Zhi)

Hagenberg, Austria, July 6, 2026

Outline

Outline

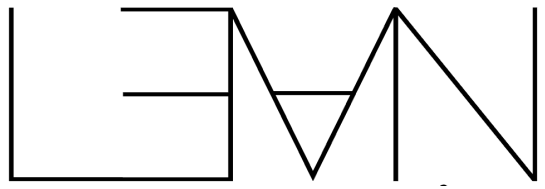
Interactive Theorem Proving with Lean 4

Interactive theorem prover

An interactive theorem prover lets us write mathematical statements and proofs in a precise formal language, with every proof step checked by a trusted kernel.

Lean 4

- A proof assistant and programming language based on dependent type theory.
- Combines human-guided proof development with automation.
- Mathlib is the community-driven mathematical library for Lean.



A small Lean proof

```
example (a b : Nat) :
    a + b = b + a := by
  rw [Nat.add_comm]
```

Polynomials in Lean 4

- Before 2019: Polynomials are **computable** in Lean. (It is **not efficient** as it is designed for reasoning rather than computing.)
- After 2019: Polynomials are changed to "noncomputable" by making it use "classical logic"[1].
- A proof can still be provided by "telling" Lean how to rewrite the polynomials.

```
example :
   ((1 / 2 : ℚ) • X + 1 : Polynomial ℚ) ^ 2 = (1 / 4 : ℚ) •X ^ 2 + X + 1 := by
 calc ((1 / 2 : ℚ) • X + 1 : Polynomial ℚ) ^ 2
   _ = ((1 / 2 : ℚ) • X) ^ 2  + 2 • (1 / 2 : ℚ) • X + 1 := by ring
   _ = (1 / 4 : ℚ) • X ^ 2 + 2 • (1 / 2 : ℚ) • X + 1 := by rw [smul_pow, div_pow];
   rfl
   _ = (1 / 4 : ℚ) • X ^ 2 + X + 1 := by rw [two_smul, ← add_smul, ← two_mul];
   simp
```

Listing 1: Show the equality in Lean 4 : $(\frac{1}{2}x+1)^2 = \frac{1}{4}x^2+x+1$

[1] the axiom of choice

Gröbner Bases

Definition (Gröbner Basis)

Fix a monomial order on $k[x_1, \ldots, x_n]$, where k is a field. A subset $G = \{g_1, \ldots, g_t\}$ of an ideal I is a **Gröbner basis** of I if

$$\langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_t) \rangle = \langle \mathrm{LT}(I) \rangle.$$

Gröbner basis theory (Buchberger, 1965) provides a unified algorithmic framework for:

- Ideal membership testing
- Solving systems of polynomial equations
- Many problems in computational algebra geometry

Algebraic Solver of Tactic `grind` in Lean 4

- It is inspired by Gröbner basis computation procedures and term rewriting completion without the formalization of Gröbner basis.
- It proves polynomial identities in commutative rings and fields via a polynomial-like structure with integral coefficients by reflection.
- It is designed to be efficient, e.g. it can verify

$$(X+1)^{20} = (X^2+2X+1)^{10}$$

in 261ms.

```
example (x : ℤ) : (x + 1) ^ 20 = (x^2 + 2*x + 1) ^ 10 := by grind
```

- It can handle several equations.

```
example (x y : ℤ) : x^2*y = 1 → x*y^2 = y → y*x = 1 := by grind
```

Gröbner Bases in Lean 4

The Problem

Gröbner basis theory has been formalized in Lean 4 by Guo et al. (2026), but:

- The formalization relies on `MvPolynomial`: a non-computable representation
- Direct computation of Gröbner bases inside Lean is impractical
- The generic design supports abstract development, not efficient symbolic computation

Our Goal

Combine external computer algebra system with formal verification in Lean 4 via a certificate-based approach.

Related Work: Combining ITP and CAS

Delegating computations to external CAS has a successful history:

- Coq + Maple (Delahaye & Mayero, 2005): field operations and algebraic simplification
- HOL Light + CAS (Seddiki et al., 2015): symbolic and numerical computation engines
- Lean 3 + Mathematica (Lewis & Wu, 2022): bidirectional extensible interface

Our Approach

- Use SageMath or SymPy as external solvers
- Design a computable polynomial representation for efficient verification
- Encapsulate the workflow into automated tactics

Computable Representation of Polynomial in Lean 4 (WIP)

```
example :
    letI X := (MvPolynomial.X 0 (R := ℚ)) -- the first variable
    letI Y := (MvPolynomial.X 1 (R := ℚ)) -- the second variable
    ((1 / 2 : ℚ) · X + Y) ^ 20 = ((1 / 4 : ℚ) · X ^ 2 + X * Y + Y ^ 2) ^ 10 := by
  -- they're equal iff their computable representations are equal
  rw [MvPolynomial.SortedRepr.eq_iff']
  decide +kernel -- decide in kernel, 2.23s
```

Listing 2: PIT of multivariate polynomials via computable representations

```
example :
    letI X := (MvPolynomial.X 0 (R := ℚ)) -- the first variable
    letI Y := (MvPolynomial.X 1 (R := ℚ)) -- the second variable
    ((1 / 2 : ℚ) · X + Y) ^ 20 = ((1 / 4 : ℚ) · X ^ 2 + X * Y + Y ^ 2) ^ 10 := by
  simp only [MvPolynomial.funext_iff, MvPolynomial.smul_eval,
     MvPolynomial.eval_add,
    MvPolynomial.eval_pow, MvPolynomial.eval_mul]
  grind -- 1.38s
```

Listing 3: PIT of multivariate polynomials via grind

(Tested on Intel Xeon Gold 6348.)

Computable Representation of Polynomial in Lean 4 (WIP)

```
example : (X 0 (R := ℚ)) + X 1 ≠ 2 * X 0 + 2 * X 1 := by
  rw [ne_eq, MvPolynomial.SortedRepr.eq_iff']
  decide +kernel
```

Listing 4: Proving inequality of multivariate polynomials

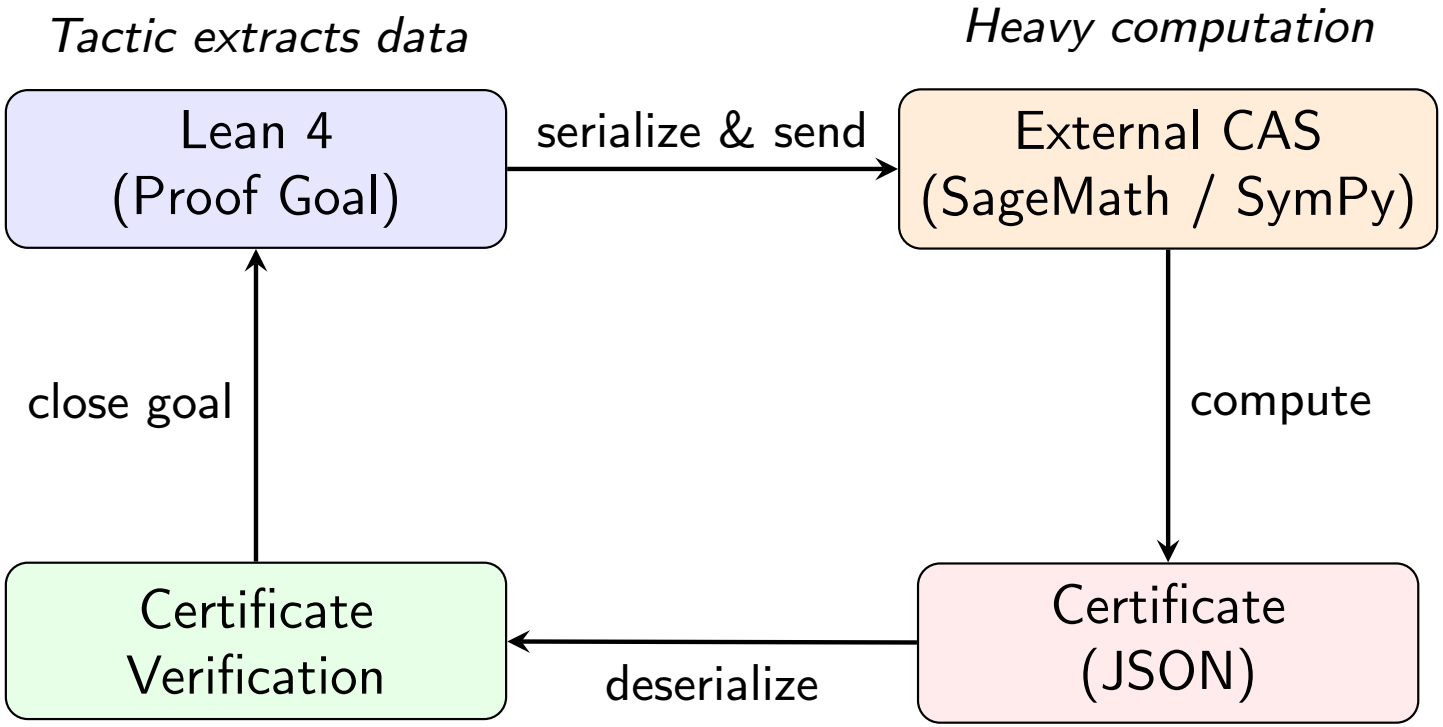
```
example :
    lex.degree (X 1 + X 2 : MvPolynomial (Fin 3) Int) ≼[lex]
      lex.degree (X 0 + X 1 ^ 2: MvPolynomial (Fin 3    ) Int) := by
  rw [MvPolynomial.SortedRepr.lex_degree_eq',
     MvPolynomial.SortedRepr.lex_degree_eq',
    SortedFinsupp.lexOrderIsoLexFinsupp.le_iff_le, ←
     Std.LawfulLECmp.isLE_iff_le (cmp := compare)]
  decide +kernel
```

Listing 5: Extracting degrees comparing monomials using lexicographic order

```
example :
    lex.leadingCoeff (3 * X 1 + X 2 : MvPolynomial (Fin 3) Int) = 3 := by
  rw [MvPolynomial.SortedRepr.lex_leadingCoeff_eq]
  decide +kernel
```

Listing 6: Extracting leading coefficient

Overview of the Framework



Outline

Stage 1: Lean to CAS

Extracting Algebraic Data from Lean

- In Lean 4, terms are represented as abstract syntax trees (`Expr`)
- We use the `quote4` library to match and extract Lean expressions against typed quotation patterns
- Extract polynomial data: variables, coefficients, and ring operations.

Serialization

- Extracted polynomials are serialized into a script
- The script is sent to an external solver (SageMath or SymPy)
- Execution is done via `IO.Process.spawn`

Stage 2: CAS to Lean

JSON Representation of Polynomials

The CAS result is returned as sparse certificate data. Each polynomial is a list of monomials, with rational coefficients and variable-exponent pairs.

```
[
  {"c": [3, 4], "e": [[1, 2], [2, 1]]},
  {"c": [-7, 1], "e": [[3, 5]]},
  {"c": [2, 1], "e": []}
]
```

Representation of $f = \frac{3}{4}x_1^2x_2 - 7x_3^5 + 2$

- `"c": [p, q]` encodes the rational number p/q.
- `"e": [[i, d], ...]` encodes the variable-power data x_i^d.
- This JSON is certificate data. it is not accepted as a proof.

Stage 2: Reconstructing Certificates from Structures

Serializable Certificate Structures

The CAS output is first decoded into small Lean structures:

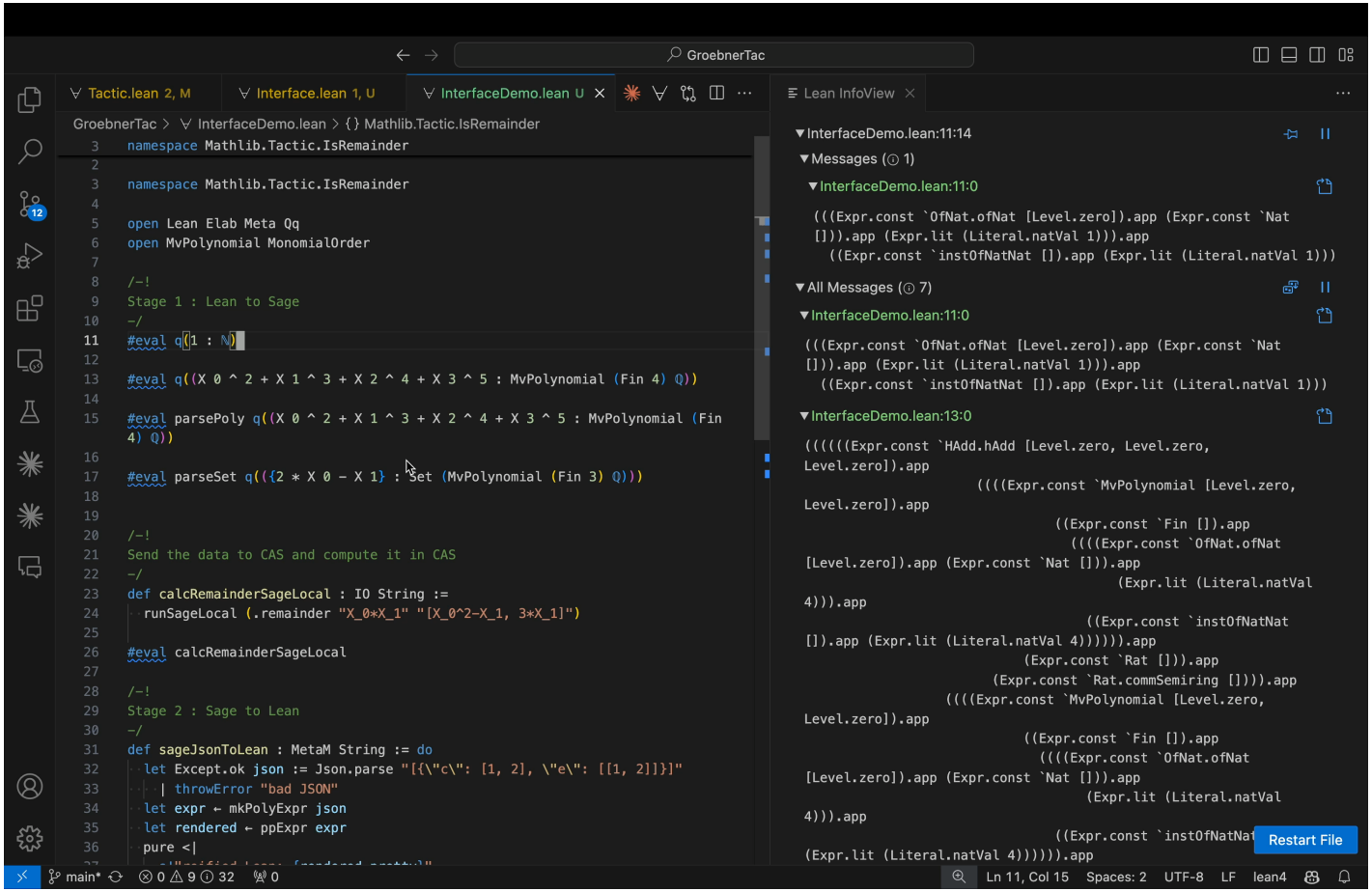
- Coefficients: `Poly.Q`, pairs (p, q)
- Variables: `Poly.V`, pairs (i, d)
- Monomials: `Poly.Mon`, coefficient plus variables
- Polynomials: `Poly.Polynomial`, lists of monomials

Reconstruction Functions

The certificate terms are rebuilt compositionally:

1. `Poly.Q.mkQ`: $(p, q) \mapsto p/q$
2. `Poly.V.mkQ`: $(i, d) \mapsto X_i^d$
3. `Poly.Mon.mkQ`: embed the reconstructed coefficient by `MvPolynomial.C`, then multiply reconstructed variables
4. `Poly.Polynomial.mkQ`: sum reconstructed monomials

Interface Demo



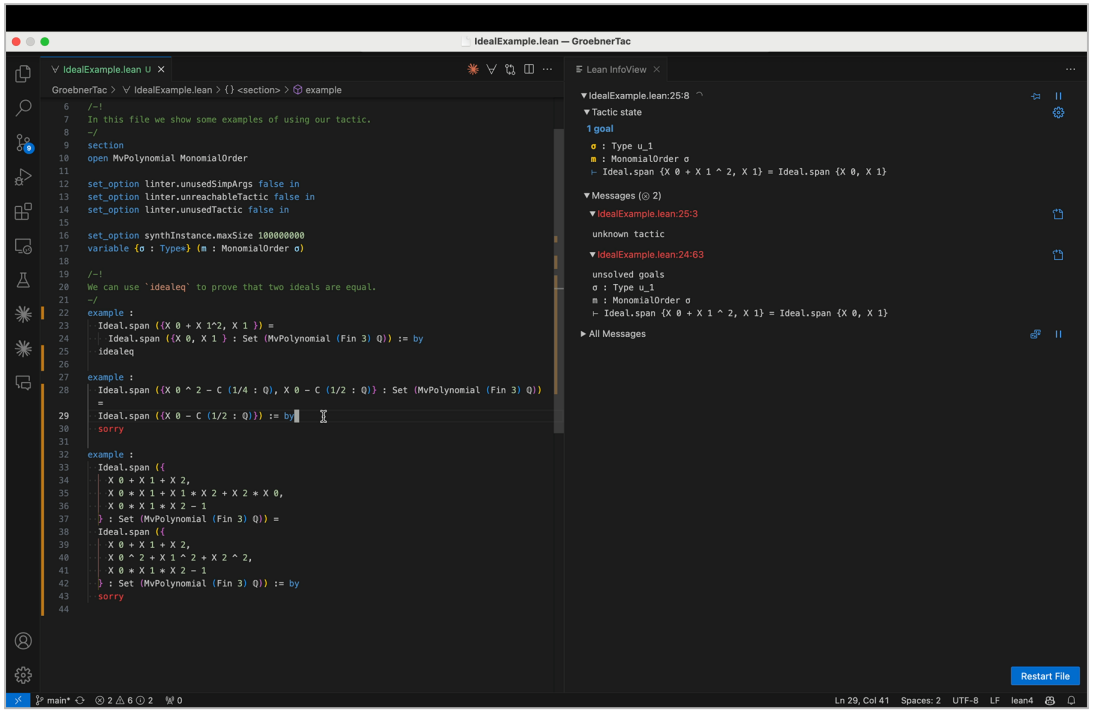
▸ Open Interface demo

Outline

Proving Ideal Equality: `idealeq`

Goal

Prove $I = J$ where $I = \langle f_1, \ldots, f_n \rangle$, $J = \langle g_1, \ldots, g_m \rangle$.



▸ Open demo video

Unified Tactic: `gb_solve`

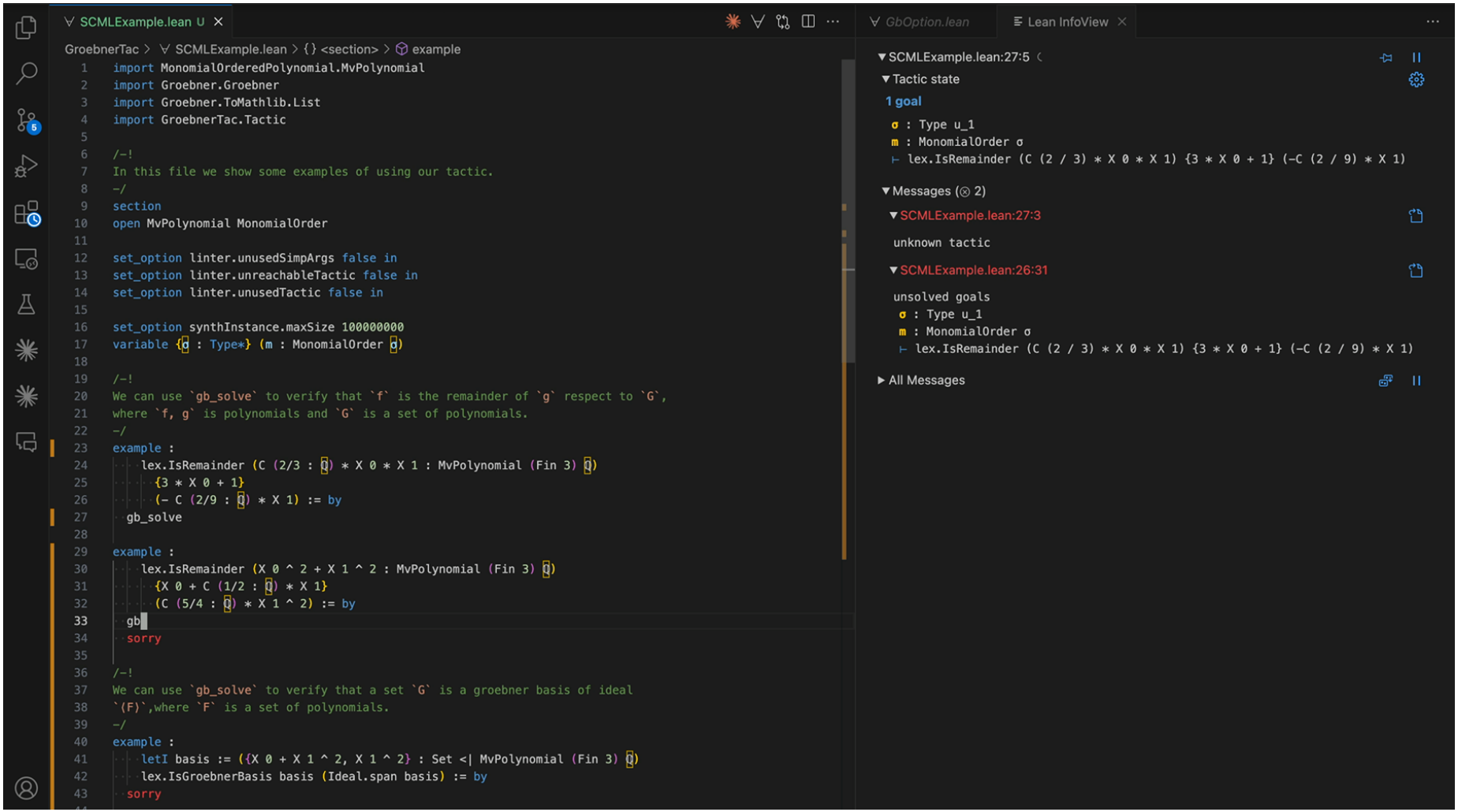
`gb_solve` inspects the proof goal and dispatches to the appropriate backend:

Goal Form	**Task**
`lex.IsRemainder f B r`	Remainder certification
`lex.IsGroebnerBasis G I`	Gröbner basis verification
`f ∈ Ideal.span B`	Ideal membership
`f ∉ Ideal.span B`	Ideal non-membership
`f ∈ (Ideal.span B).radical`	Radical membership
`f ∉ (Ideal.span B).radical`	Radical non-membership

Currently supports:

- Lexicographic monomial order
- Polynomial ring $\mathbb{Q}[x_0, \ldots, x_n]$

Demo of gb_solve

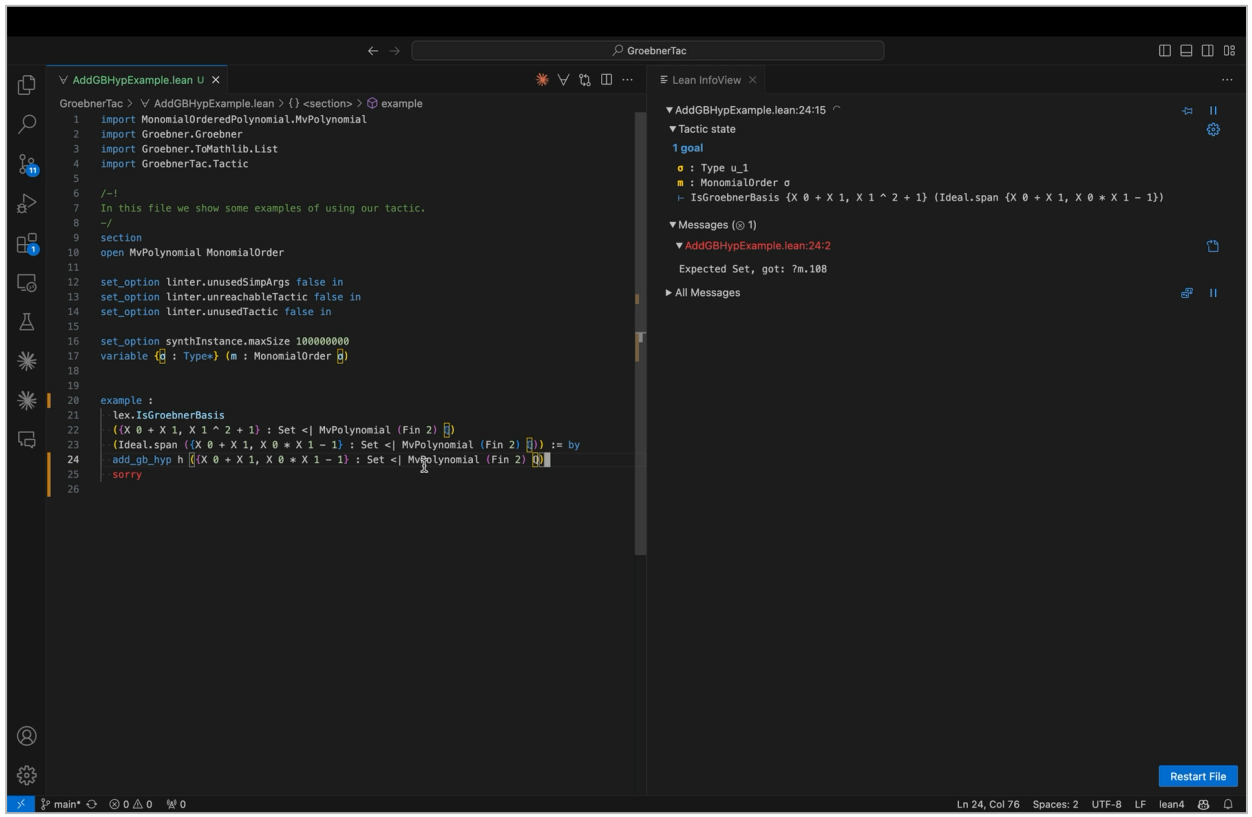


▸ Open demo video

Computing and Certifying a Gröbner Basis: `add_gb_hyp`

Motivation

The previous tactics assume a Gröbner basis G is already known. In practice, one starts with an arbitrary generating set B. Then we develop the tactic `add_gb_hyp`.



▸ Open demo video

Computation Modes

Value	Backend	Requirement
0 (default)	Local SageMath	SageMath installed
1	SageMath API	Internet access
2	Local SymPy	Python + SymPy

```
set_option gb_tactic.backend 1 in
example : Ideal.span ({X 0 + X 1^2, X 1^2} :
    Set (MvPolynomial (Fin 2) ℚ))
  = Ideal.span ({X 0, X 1^2} :
    Set (MvPolynomial (Fin 2) ℚ)) := by
  idealeq
```

Outline

Why Machine Learning Enters the Picture

A Division of Labor

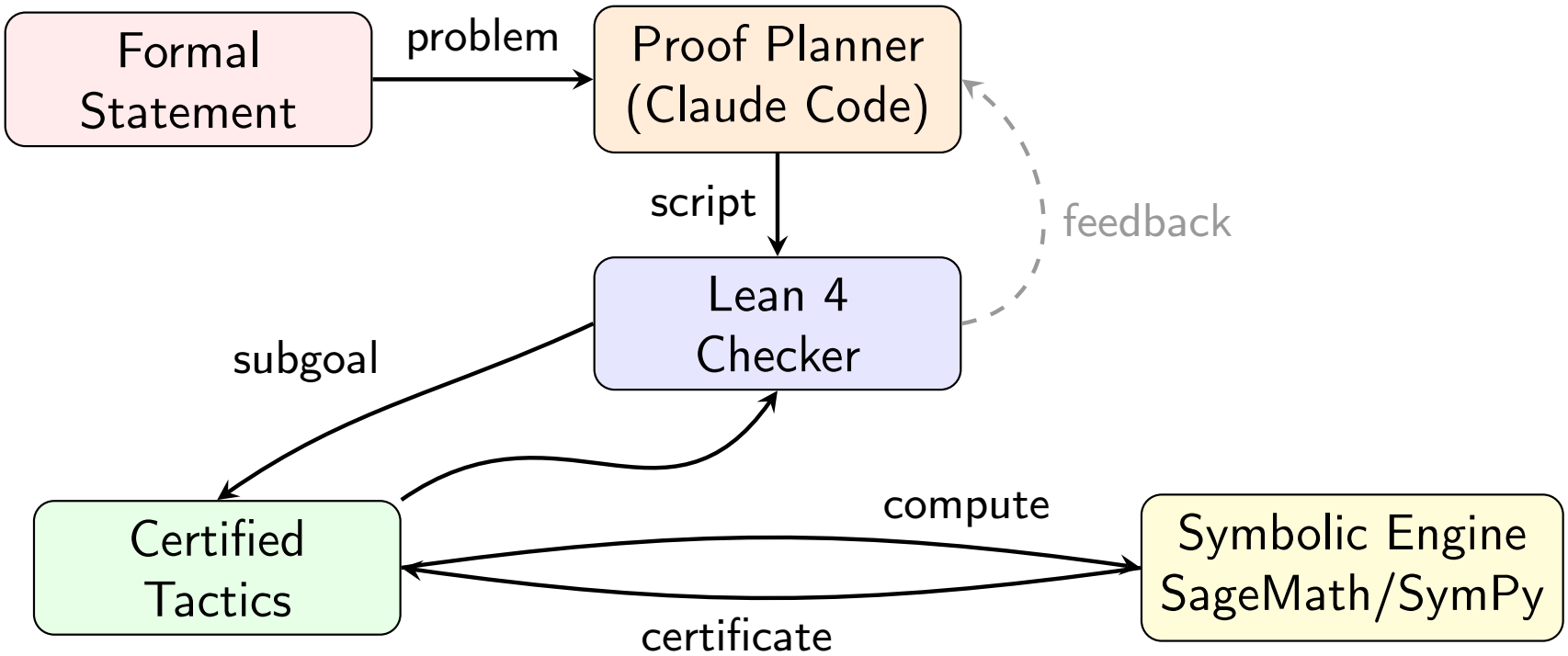
Our framework separates three different forms of intelligence:

- **Symbolic computation**: CAS backends perform exact algebraic computation
- **Formal verification**: Lean checks every imported certificate in its trusted kernel
- **Machine learning**: an LLM can guide proof search and select appropriate tactics

Key Principle

The LLM is used as a **proof orchestrator**, not as a trusted proof authority. It may propose proof steps, call tools, and adapt its strategy, but the final proof object must be accepted by Lean.

LLM-assisted Proof Workflow



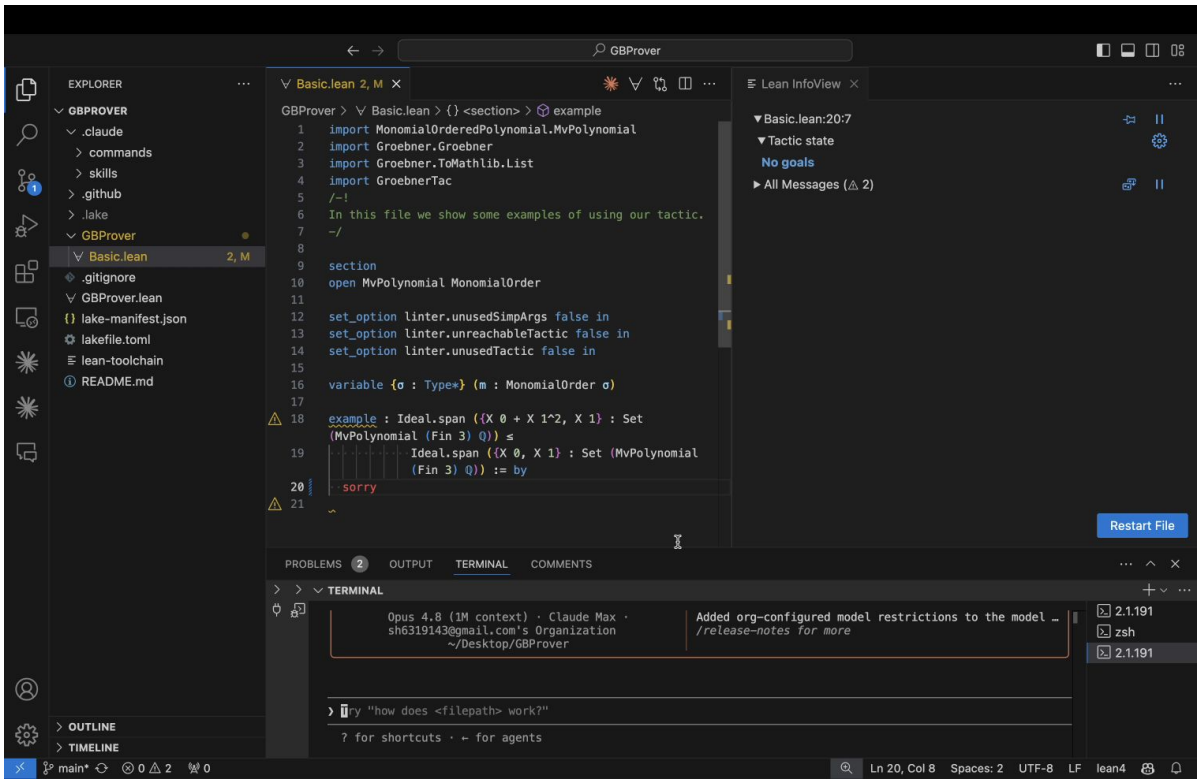
Only Lean's kernel is trusted; ML and CAS remain outside the trusted core.

- The LLM handles high-level proof planning and interaction with the Lean environment
- The CAS handles computationally intensive algebraic tasks
- Lean provides the final correctness guarantee

Claude Code Skill for Algebraic Proofs

Automation Interface

We developed a Claude Code skill that exposes the certified tactics as callable tools inside an interactive Lean workflow.



▸ Open video demo

Outline

Summary & Future Work

Main Message

Our system bridges efficient CAS computation and trustworthy Lean verification through verifiable certificates.

What This Enables

- External CAS provides efficient Gröbner-basis computation; Lean verifies certificates
- Tactics hide certificate handling behind a proof-oriented interface
- LLMs guide proof search, while Lean's kernel remains the trust anchor

Future Directions

- Support more methods, including Wu–Ritt and CAD
- Generalize coefficients and monomial orders
- Gradually replace external certificates with verified algorithms in Lean

Thank you!

Questions?

Email: shenhao24@amss.ac.cn

Source code: https://github.com/WuProver/GroebnerTactic

LLM skill: https://github.com/tsuki8/GBProver