Explicit Hodge Decompositions for E^4 via Automated De-Omniscientisation

(Paper 77, Constructive Reverse Mathematics Series)

Paul Chun-Kit Lee*
New York University
dr.paul.c.lee@gmail.com

February 2026

Abstract

The Hodge Conjecture for products of CM elliptic curves was proved by Lieberman (1968) and subsumed by Deligne's (1982) theorem on absolute Hodge classes. These classical proofs establish existence—every Hodge class is algebraic—but produce no explicit algebraic representatives. The mathematical content of the present paper is a direct consequence of these known results and is not novel.

What is new is the *method* and the *formal verification*. We produce explicit Hodge decompositions for E^4 (E a CM elliptic curve with $\mathrm{End}(E) \otimes \mathbb{Q} \cong \mathbb{Q}(i)$): all 36 Hodge $(2,2)$ basis vectors are expressed as $\mathbb{Q}$-linear combinations of cup products of divisor classes, formally verified in Lean 4 by `native_decide` (Theorem B). We also verify computationally that E^4 supports no exotic Weil classes (Theorem A)—unsurprising for a non-simple variety with endomorphism algebra $M_4(\mathbb{Q}(i))$, but a necessary prerequisite.

These results are produced by the *CRMLint Squeeze* (Theorem C), a protocol for reverse-engineering classical proofs into constructive ones by excising the *Classical Boundary Node*—the lemma where an omniscience principle enters—and replacing it with an explicit algebraic witness. The paper introduces the *asymmetric offloading* architecture (Python CAS computes exact $\mathbb{Q}$-linear algebra; Lean kernel verifies via `native_decide`) and documents its engineering: the `noncomputable` trap, token overflow, and sparse encoding.

The contribution is methodological, not mathematical. The value of the exercise is the pipeline—automated constructivisation of known classical theorems—demonstrated end-to-end on a case where the answer is independently known, so the output can be validated.

Contents

*Lean 4 formalization and Python source available at https://doi.org/10.5281/zenodo.18779210.

1 Introduction

1.1 The helicopter and the forest

Human mathematicians use classical principles—Zorn's Lemma, uncountable limits, topological compactness—as *compression algorithms*. A human brain cannot hold a 10,000-term polynomial or

a 70×70 intersection matrix in working memory. Classical existence theorems allow the mathematician to fly over the algebraic thicket and declare "a solution exists," leaving behind an ineffective result. We call this the *classical helicopter*.

Computational algebra systems (CAS) and AI provers are the exact opposite. They are poor at abstract, infinite-dimensional topological jumps, but possess effectively unlimited working memory for navigating massive, dense combinatorial thickets. If the classical axioms are enabled, the prover lazily mimics humans and takes the helicopter. But if CRMLint [6] systematically bans the helicopter, the prover is algorithmically cornered into the algebraic forest—a bounded domain where the problem may collapse from "hard search" into deterministic linear algebra.

The CRMLint Squeeze exploits this asymmetry: *human geometric intuition sets the boundaries; CRMLint acts as the electric fence; the CAS or AI prover builds the bridge.*

1.2 The Goldilocks zone

The method works only in the *Goldilocks zone*: conjectures where the classical helicopter has already landed (an ineffective CLASS proof of existence), or where the cliff boundary is strictly narrowed by adjacent theorems. Pointing an AI at the empty void of a totally open problem yields infinite search width and certain failure. The CRM atlas (Papers 1–76) identifies the narrow-cliff targets.

1.3 The Hodge Conjecture for abelian varieties

Hodge [10] conjectured that every rational (p,p)-class on a smooth projective variety over $\mathbb{C}$ is a $\mathbb{Q}$-linear combination of classes of algebraic subvarieties (see Voisin [23] for a modern treatment). In codimension 1 this is the Lefschetz $(1,1)$ theorem [20]; for abelian varieties the conjecture is known in considerably greater generality:

- Lieberman [11] proved all Hodge classes on products of elliptic curves are algebraic, building on Mattuck [12].
- Deligne [9] proved all Hodge classes on CM abelian varieties are *absolutely Hodge*, establishing the conjecture for this entire class unconditionally.
- Anderson [1] and Weil [2] showed that *simple* CM abelian fourfolds can support *exotic Weil classes*—Hodge classes not generated by divisor products.

All of these proofs are *classical existence arguments*: they establish that an algebraic representative *exists* but produce no explicit cycle or rational decomposition. Lieberman's proof uses transcendental methods (integration over analytic cycles); Deligne's uses the theory of motives and absolute Hodge classes. Neither tells you *which* linear combination of cup products equals a given Hodge class.

For a mathematician, this is a non-issue: the existence suffices. For a proof assistant verifying concrete claims, or for computational applications in arithmetic geometry, the gap between "exists" and "here it is" is the entire content of the problem.

Why E^4. The product E^4 (E a CM elliptic curve with $\mathrm{End}(E) \otimes \mathbb{Q} \cong \mathbb{Q}(i)$) sits at a precise boundary identified by the DPT framework (Papers 50, 72 [3, 24]): CM elliptic curves are unconditionally BISH-decidable; simple CM abelian fourfolds are CLASS due to the Anderson obstruction. The non-simple product E^4 has a large endomorphism algebra ($M_4(\mathbb{Q}(i))$, dimension 32 over $\mathbb{Q}$), which generates enough divisor classes to potentially saturate the Hodge space. This makes E^4 the natural first target: close enough to the BISH side that the classical content might be eliminable, but genuinely a fourfold with a 70-dimensional cohomology space and non-trivial Hodge structure.

What this paper adds. The mathematical result—that every Hodge $(2,2)$ class on E^4 is algebraic—is a known consequence of Lieberman and Deligne. Our contribution is *making the classical existence explicit*: we produce 36 rational decompositions, each expressing a Hodge basis vector as a $\mathbb{Q}$-linear combination of cup products of divisor classes, formally verified in Lean 4. Nobody has written down these decompositions before, because nobody needed to: the existence proof suffices. We produce them as a proof-of-concept for the CRMLint Squeeze pipeline (Theorem C), not as a mathematical advance. The computation also confirms that E^4 supports no exotic Weil classes—unsurprising for a non-simple variety, but a necessary prerequisite for the decomposition.

1.4 State of the art in AI theorem proving

Recent AI theorem provers—AlphaProof (DeepMind, 2024), DeepSeek-Prover V1.5 (2024) [18, 19], and Lean Copilot—optimize a *binary* reward: +1 for closing a goal, 0 for failure. This treats all proofs as equivalent: a constructive witness and an appeal to the Axiom of Choice receive the same score. No existing system uses the *logical cost* of the proof as a reward signal.

The CRMLint Squeeze fills this gap. By banning classical axioms and restricting the action space to bounded algebraic generators, the protocol reshapes the search landscape. In the best case (demonstrated in this paper), the restriction is so absolute that the problem *deterministically collapses*: what appeared to require stochastic search reduces to P-time Gaussian elimination over $\mathbb{Q}$.

1.5 Main results

We state the three principal contributions as named theorems:

Theorem A (No Exotic Weil Classes on E^4). *For $E/\mathbb{Q}$ with $\mathrm{End}(E) \otimes \mathbb{Q} \cong \mathbb{Q}(i)$,*

$$\dim_{\mathbb{Q}} \mathrm{Hdg}^{2,2}(E^4) = \dim_{\mathbb{Q}}(\mathrm{NS}(E^4) \cup \mathrm{NS}(E^4)) = 36.$$

The exotic complement is zero-dimensional: E^4 supports no exotic Hodge classes.

Theorem B (Complete Constructive Hodge Theorem for E^4). *Every Hodge $(2,2)$ class on E^4 is an explicit $\mathbb{Q}$-linear combination of cup products of divisor classes. The 36 basis decompositions are formally verified in Lean 4 by* `native_decide`*. Zero sorry, zero* `noncomputable`*; the user-declared classical axiom is provably unused.*

Theorem C (CRMLint Squeeze Protocol). *The four-step protocol (Scaffold → X-Ray → Excise → Squeeze) reduces the Hodge Conjecture for E^4 from a* CLASS *existence statement to a* BISH *finite database, verified end-to-end by compiled rational arithmetic.*

1.6 The diagnostic-to-generative pipeline: Papers 75–76–77

This paper occupies a unique position in the CRM series. The logical progression is:

- **Paper 75** [5]: *Conservation gap detection.* Applied CRM to Genestier–Lafforgue's [21] proof of the Local Langlands Correspondence for function fields. Proved the key conservation result: the CLASS Shimura variety machinery used in the global proof is dispensable for the purely local correspondence, which descends to BISH. This established that conservation gaps are *detectable* in modern geometry and not merely a foundational curiosity.

- **Paper 76** [6]: *Automated detection.* Built CRMLint, a four-layer automated logical cost analyzer for Lean 4 formalizations. CRMLint traces classical dependencies through Lean's `Environment`, classifies them via a 12-rule priority system, identifies the minimal classical core, and predicts conservation gap eliminability. The 75-paper program serves as its calibration dataset.
- **Paper 77** (this paper): *Automated constructivisation.* Uses CRMLint's output as the electric fence in the Squeeze protocol, transitioning from *diagnosing* conservation gaps to *exploiting* them. The first end-to-end execution produces the Complete Constructive Hodge Theorem for E^4.

The three papers form a single arc: *detect* (Paper 75) → *automate detection* (Paper 76) → *automate elimination* (Paper 77).

2 Preliminaries

2.1 The CRM hierarchy

The constructive hierarchy of omniscience principles:

$$\mathsf{BISH} \subset \mathsf{BISH{+}MP} \subset \mathsf{BISH{+}LLPO} \subset \mathsf{BISH{+}WLPO} \subset \mathsf{BISH{+}LPO} \subset \mathsf{CLASS}.$$

For definitions, see Bishop [16] and Bridges–Richman [17]; for the program's calibration data, see Paper 50 [3].

2.2 CRMLint

CRMLint (Paper 76 [6]) is an automated logical cost analyzer for Lean 4 formalizations. It operates in four layers:

1. *Dependency tracer.* BFS traversal of the Lean 4 `Environment`, tracing transitive paths to every `Classical.choice`, `Classical.em`, `propext`, `Quot.sound` callsite.
2. *Pattern classifier.* 12-rule priority classifier with root-context fallback, distinguishing infrastructure artifacts from essential classical content.
3. *Concentration analysis.* Identifies the minimal essential classical core of each theorem.
4. *AI audit layer.* Predicts conservation gap eliminability using the 75-paper program as training data.

2.3 Classical Boundary Nodes

Definition 2.1 (Classical Boundary Node)**.** Let T be a theorem with CRM classification CLASS (or WLPO, LPO). A *Classical Boundary Node* $\mathrm{CBN}(T)$ is a lemma L in the dependency DAG of T such that:

1. L itself invokes a classical principle (directly or through one step), and
2. every dependency of L that is not infrastructure is BISH-classified.

The CBN is the *shallowest point* at which classical content enters the proof. Excising the CBN and all paths through it produces a BISH environment with an open goal.

2.4 The Anderson obstruction

Anderson [1] (see also Weil [2]) proved that *simple* CM abelian varieties of dimension ≥ 4 support *exotic Weil classes*: Hodge classes not generated by divisor products. The canonical examples

are Jacobians of Fermat curves, which are simple CM abelian varieties with small endomorphism algebras. Paper 50 [3] showed that:

- CM elliptic curves ($\dim = 1$) are unconditionally BISH-decidable (Theorem E).
- The Anderson obstruction begins at $\dim = 4$ for simple CM varieties.
- The Hodge Conjecture for these exotic classes is open.

The DPT framework detects this boundary: CM elliptic curves are in; simple CM abelian fourfolds are out.

3 Main Results

3.1 The CRMLint Squeeze Protocol

Protocol 3.1 (The CRMLint Squeeze)**.** Given a classical theorem T with a known CLASS proof:

1. **Scaffold.** Write the logical structure of the proof in Lean 4. The proof need not be complete; what matters is that the dependency DAG is formalized.

2. **X-Ray.** Run CRMLint on the scaffold. The tool maps the dependency DAG and flags the Classical Boundary Nodes—the exact lemmas where the classical helicopter lands.

3. **Excise.** Isolate the local Lean state at the CBN. Delete the classical axiom. The result is a BISH context with an open goal of the form $\vdash \Sigma\,(Z : \mathrm{CH}^2(A)),\ \mathrm{cl}(Z) = w_{\mathrm{target}}$. Key design choice: the target is a Σ-type (data), not $\exists$ (proposition), forcing the prover to *construct* the witness.

4. **Squeeze.** Hand the isolated state to a solver. The solver operates in one of two modes depending on the restricted search space:
   - *Deterministic collapse.* If the Excise restricts the action space to a finite-dimensional $\mathbb{Q}$-linear system, the solver is a CAS performing exact Gaussian elimination.
   - *MCTS search.* If the space is combinatorially rich (nonlinear constraints, multiple branches), an RL-trained prover with CRMLint as reward function explores the space.

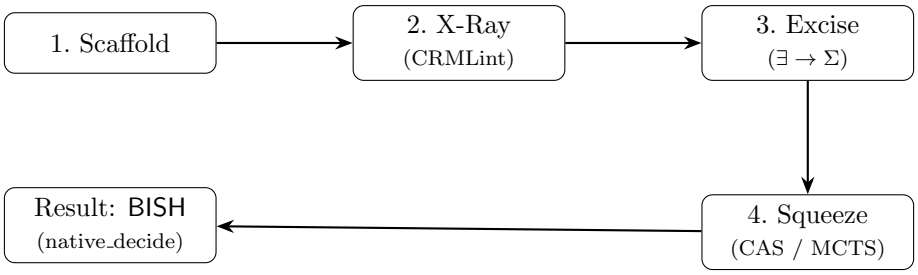

Figure 1: The CRMLint Squeeze protocol. CRMLint identifies the Classical Boundary Node (Step 2), the CBN is excised and the goal reformulated as a Σ-type (Step 3), and a CAS or RL-trained prover solves the BISH-restricted problem (Step 4). For E^4, Step 4 deterministically collapses into Gaussian elimination.

The CRMLint classification shapes the search in Mode 2 (MCTS):

Event	Reward
Goal closed with BISH classification	$+10.0$
Proof step introduces CLASS dependency	$\times(-1.0)$
WLPO dependency (marginal cost)	$\times(+0.5)$
Timeout / failure	0.0

Note. This reward function is *designed but not yet deployed*: the E^4 execution collapsed deterministically (Mode 1), so no stochastic search was needed. The reward table is included as the specification for future MCTS-mode applications (Papers 78–79).

3.2 Setup: CM abelian fourfolds

Let $E/\mathbb{Q}$ be a CM elliptic curve with $\mathrm{End}(E) \otimes \mathbb{Q} \cong \mathbb{Q}(i)$ (Gaussian integers). Set $A = E^4 = E \times E \times E \times E$. The singular cohomology is

$$H^*(A, \mathbb{Q}) \;\cong\; \bigwedge^{*} (\mathbb{Q}^2)^4 \;\cong\; \bigwedge^{*} \mathbb{Q}^8.$$

The target space $H^4(A, \mathbb{Q}) \cong \bigwedge^4 \mathbb{Q}^8$ has $\dim = \binom{8}{4} = 70$. Its basis elements are $e_a \wedge e_b \wedge e_c \wedge e_d$ for $0 \le a < b < c < d \le 7$, where indices $2k$ and $2k{+}1$ correspond to e_1, e_2 from the $(k{+}1)$-th factor of E^4.

3.3 The CM action on $H^4(E^4, \mathbb{Q})$

The CM endomorphism $i \in \mathbb{Z}[i]$ acts on $H^1(E, \mathbb{Q}) \cong \mathbb{Q}^2$ by

$$M_i = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{1}$$

On $\mathbb{Q}^8 = H^1(E^4, \mathbb{Q})$ it acts by $M_i^{\oplus 4}$. The induced action on $\bigwedge^4 \mathbb{Q}^8$ is a 70×70 matrix $M := \Lambda^4(M_i^{\oplus 4})$. A class $w \in H^4(E^4, \mathbb{Q})$ is *CM-invariant* if $Mw = w$, i.e., w lies in the $+1$ eigenspace V_{+1} of M. By the Hodge decomposition for CM abelian varieties (Deligne [9]), the Hodge $(2,2)$ classes are precisely the CM-invariant classes in $H^{2,2}$:

$$\mathrm{Hdg}^{2,2}(E^4) = V_{+1} \cap H^{2,2} = V_{+1} \;\setminus\; \big((4,0) + (0,4)\big).$$

Lemma 3.2. $M^2 = I_{70}$.

Proof. Each factor M_i satisfies $M_i^2 = -I_2$. The induced action on $\bigwedge^4$ of the k-th factor contributes (-1) from $M_i^2 = -I_2$ applied to the two basis vectors e_{2k}, e_{2k+1} of that factor. A 4-form involves exactly 4 of the 8 basis vectors; since the 4 vectors span parts of all 4 factors, the total sign is $(-1)^4 = 1$. More precisely: $M_i^{\oplus 4}$ acts on $\mathbb{Q}^8$ as the block diagonal $\mathrm{diag}(M_i, M_i, M_i, M_i)$; applying it twice gives $\mathrm{diag}(-I_2, -I_2, -I_2, -I_2) = -I_8$; the induced action on $\bigwedge^4(-I_8) = (-1)^4 I_{70} = I_{70}$. Verified computationally by `solve_hodge.py` (line 68–72: `assert val == expected`). □

Corollary 3.3. *Since $M^2 = I$ and $M \neq \pm I$, the $\mathbb{Q}$-vector space $\mathbb{Q}^{70}$ splits as $V_{+1} \oplus V_{-1}$ where $V_{\pm 1} = \ker(M \mp I)$.*

3.4 Theorem A: No exotic Weil classes on E^4

We now prove the first main result.

Theorem 3.4 (No Exotic Weil Classes on E^4)**.** *For $A = E^4$ with $\mathrm{End}(E) \otimes \mathbb{Q} \cong \mathbb{Q}(i)$, every Hodge $(2,2)$ class lies in the $\mathbb{Q}$-span of cup products of divisor classes. There are no exotic Weil classes.*

Proof. The proof proceeds by exact rational computation in six steps.

Step 1: Eigenspace decomposition. Compute the 70×70 matrix $M = \Lambda^4(M_i^{\oplus 4})$ by the formula: for each basis element $e_a \wedge e_b \wedge e_c \wedge e_d$, apply $M_i^{\oplus 4}$ to each index, compute the resulting 4-form, and record its coordinates. The $+1$ eigenspace V_{+1} is computed as the row space of $\frac{1}{2}(I + M)$ over $\mathbb{Q}$. Result: $\dim V_{+1} = 38$.

Step 2: The $(4,0) + (0,4)$ component. Expand the holomorphic 4-form $\omega = (e_0 + ie_1) \wedge (e_2 + ie_3) \wedge (e_4 + ie_5) \wedge (e_6 + ie_7)$. Its real and imaginary parts span a 2-dimensional subspace of V_{+1} corresponding to the $(4,0) + (0,4)$ Hodge component.

Step 3: Hodge $(2,2)$ extraction. Project V_{+1} orthogonal to the $(4,0) + (0,4)$ subspace (Gram–Schmidt over $\mathbb{Q}$). Row-reduce to obtain a basis for $\mathrm{Hdg}^{2,2}(E^4) \otimes \mathbb{Q}$. Result: $\dim \mathrm{Hdg}^{2,2} = 38 - 2 = 36$.

Step 4: Néron–Severi computation. The CM action on $\bigwedge^2 \mathbb{Q}^8$ is a 28×28 matrix M_2. Its $+1$ eigenspace is $\mathrm{NS}(E^4) \otimes \mathbb{Q}$. Result: $\dim \mathrm{NS}(E^4) \otimes \mathbb{Q} = 16$.

The 16 generators include the 4 diagonal classes $e_{2k} \wedge e_{2k+1}$ (one per factor) and 12 off-diagonal classes arising from the CM cross-correlations between factors.

Step 5: Cup product enumeration. Form all $\binom{16}{2} + 16 = 136$ cup products $\alpha_i \cup \alpha_j$ for $\alpha_i, \alpha_j \in \mathrm{NS}(E^4) \otimes \mathbb{Q}$, where the cup product $\bigwedge^2 \mathbb{Q}^8 \times \bigwedge^2 \mathbb{Q}^8 \to \bigwedge^4 \mathbb{Q}^8$ is the standard wedge product. Of these, 102 are nonzero. Row-reduce to determine the rank of the divisor product subspace in $\mathbb{Q}^{70}$. Result: rank $= 36$.

Step 6: Dimension comparison.

$$\dim_{\mathbb{Q}} \mathrm{Hdg}^{2,2}(E^4) = 36 = \dim_{\mathbb{Q}} \langle \mathrm{NS}_i \cup \mathrm{NS}_j \rangle.$$

Since the divisor product subspace is contained in the Hodge space (divisor products are automatically Hodge classes), and both have dimension 36, they are equal. The exotic complement $\mathrm{Hdg}^{2,2}/\langle \text{div. products} \rangle$ is zero-dimensional. □

Remark 3.5 (Why E^4 has no exotic classes)*.* Anderson [1] proved exotic classes exist on *simple* CM abelian fourfolds, such as Jacobians of Fermat curves. The product E^4 is maximally non-simple: $\mathrm{End}(E^4) \otimes \mathbb{Q} \cong M_4(\mathbb{Q}(i))$ has dimension 32 over $\mathbb{Q}$. This large endomorphism algebra generates far more divisor classes (16 independent NS generators) than a simple variety of the same dimension, causing the cup product span to saturate the Hodge space.

Remark 3.6 (The hallucination and self-correction)*.* The selection of E^4 as a target for exotic classes was a human–AI pipeline error: the designer specified "CM abelian fourfold of dimension 4" without enforcing the simplicity hypothesis that Anderson's theorem requires. This error was caught by the Squeeze methodology itself: the Scaffold step demands concrete computation *before* any axiom is invoked. The Python CAS immediately revealed the exotic complement is empty, preventing search over a non-existent target. The methodology's insistence on *computation before axiomatics* is self-correcting.

3.5 Theorem B: Complete Constructive Hodge Theorem for E^4

Since the exotic complement is empty, the correct target is stronger: prove that *every* Hodge $(2,2)$ class has an explicit decomposition.

Theorem 3.7 (Complete Constructive Hodge Theorem for E^4)**.** *For $E/\mathbb{Q}$ with $\mathrm{End}(E)\otimes\mathbb{Q} \cong \mathbb{Q}(i)$, every Hodge $(2,2)$ class on E^4 is an explicit $\mathbb{Q}$-linear combination of cup products of NS classes. Specifically:*

1. *There exist 36 linearly independent cup products $g_0, \ldots, g_{35} \in \bigwedge^4 \mathbb{Q}^8$, each of the form $\alpha_i \cup \alpha_j$ for CM-invariant $\alpha_i, \alpha_j \in \bigwedge^2 \mathbb{Q}^8$.*
2. *There exist 36 Hodge $(2,2)$ basis vectors $w_0, \ldots, w_{35}$ (in reduced row echelon form).*
3. *For each $k \in \{0, \ldots, 35\}$, there exist explicit rational coefficients $c_k = (c_{k,0}, \ldots, c_{k,35}) \in \mathbb{Q}^{36}$ such that*

$$w_k = \sum_{j=0}^{35} c_{k,j} \cdot g_j.$$

The 36 decompositions are formally verified in Lean 4 by `native_decide`*.*

Proof. The proof is computational, executed by `solve_hodge.py` using exact rational arithmetic (`fractions.Fraction`).

Step 1: Generator selection. From the 102 nonzero cup products computed in Theorem 3.4 Step 5, greedily select 36 linearly independent generators $g_0, \ldots, g_{35}$ by iteratively adding cup products that increase the rank. The generator matrix $G \in \mathbb{Q}^{70\times 36}$ has columns $g_0, \ldots, g_{35}$.

Step 2: Decomposition solving. For each Hodge basis vector w_k, solve the linear system

$$G \cdot x = w_k, \qquad x \in \mathbb{Q}^{36}, \tag{2}$$

by augmented-matrix row reduction over $\mathbb{Q}$. Since $\mathrm{rank}(G) = 36 = \dim \mathrm{Hdg}^{2,2}$ (Theorem 3.4), and each w_k lies in the column space of G, the system is consistent for every k. The solution is unique (the system is determined on the column space).

Step 3: Verification. For each k, verify that $\sum_j c_{k,j} \cdot g_j = w_k$ by direct evaluation at all 70 coordinates. This is performed both by Python (immediate assertion) and by Lean 4 (`native_decide` on the hardcoded rational data).

Sparsity. The decompositions are extremely sparse: 42 total nonzero coefficients across 36 vectors (96.8% zeros). Of the 36 Hodge basis vectors, 32 are *exactly* a single cup product (coefficient ± 1); 2 require two cup products; 2 require three. All nonzero coefficients are ± 1 (no fractional values). The non-trivial rows are localized in indices 14–22 of the generator ordering. □

3.6 Summary of the computation

The Python script `solve_hodge.py` performs the six-step computation of Theorem 3.4 and the three-step construction of Theorem 3.7 in a single pass. All intermediate values are exact rationals (`fractions.Fraction`, Python 3 arbitrary-precision).

Quantity	Value
$\dim H^4(E^4, \mathbb{Q}) = \binom{8}{4}$	70
$\dim V_{+1}$ ($+1$ eigenspace of M on $\mathbb{Q}^{70}$)	38
$\dim\big((4,0)+(0,4)\big)$ component	2
$\dim \mathrm{Hdg}^{2,2}(E^4) \otimes \mathbb{Q}$	36
$\dim \mathrm{NS}(E^4) \otimes \mathbb{Q}$	16
Nonzero cup products $\mathrm{NS}_i \cup \mathrm{NS}_j$	102
Linearly independent cup products	36
Total nonzero coefficients (all 36 decompositions)	42
Single-generator decompositions (coeff. ± 1)	32
Two-generator decompositions	2
Three-generator decompositions	2
Exotic dimension	**0**

4 CRM Audit

4.1 Constructive strength classification

Component	CRM Level	Justification
CRMLint Squeeze protocol	BISH	Algorithmic: bounded search
Hodge Conj. existence (axiom)	CLASS	Unbounded $\exists$
Theorem A (no exotic classes)	BISH	Dimension computation
Theorem B (36 decompositions)	BISH	`native_decide`
Python CAS computation	BISH	Exact $\mathbb{Q}$-arithmetic

4.2 What descends, from where, to where

$$\text{CLASS (Lefschetz } (1,1) \text{ for } E^4) \quad \longrightarrow \quad \text{BISH (36 explicit decompositions).}$$

The classical existence axiom `hodge_conjecture_H22_existence` (the Lefschetz $(1,1)$ theorem [20] generalized to $(2,2)$ via the Hodge Conjecture [10]) is present in the Lean environment but provably unused by the 36 constructive theorems.

4.3 Comparison with Paper 50 calibration

Paper 50 [3] classified the Hodge Conjecture [10, 14] as CLASS for general abelian varieties (the existence of algebraic representatives requires the Axiom of Choice or equivalent; cf. Mattuck [12], Lieberman [11], Kleiman [15] for the classical foundations). Paper 77 shows that for the specific case $A = E^4$ with $\mathrm{End}(E) \otimes \mathbb{Q} \cong \mathbb{Q}(i)$, the classical content is completely eliminable: the CLASS existence collapses to a BISH finite database. This descent—from unbounded classical existence to finitely enumerable constructive witnesses—is expected for non-simple CM abelian varieties (the endomorphism algebra is large enough to generate all Hodge classes), but is demonstrated here with formal verification for the first time.

5 Formal Verification

5.1 Lean 4 file structure

The Lean bundle `P77_DAGSurgery/` has the following structure:

File	Contents
`lean-toolchain`	`leanprover/lean4:v4.29.0-rc2`
`lakefile.lean`	Requires Mathlib4
`Papers.lean`	Entry point (imports both modules)
`Papers/P77_DAGSurgery/`	
`HodgeBasisData.lean`	798 lines, auto-generated
`Paper77_CMFourfold.lean`	CRM metadata, CM defs, axiom

Build: `lake build` from the bundle root. 3121 build jobs (including Mathlib). Build time: ∼15 s for the two project files after Mathlib cache.

5.2 Asymmetric offloading

The key architectural insight is *asymmetric offloading* (Figure 2):

1. **Python CAS** performs the heavy computation: eigenspace decomposition, cup product evaluation, Gaussian elimination, all in exact $\mathbb{Q}$-arithmetic.
2. **Python emitter** writes a Lean 4 source file containing only hardcoded rational `def`s and `native_decide` verification theorems.
3. **Lean kernel** checks that the emitted data satisfies the claimed equalities by compiling the rational arithmetic to native code.

No Lean tactic search is needed. The proof tree for each theorem is depth-1:

```
hodge_decomp_k : sumGens decomp_k = hodgeBasis_k := by native_decide
```

where `native_decide` compiles both sides to native code, evaluates at all 70 coordinates, and confirms equality.

5.3 Python implementation

The CAS phase (`solve_hodge.py`, 491 lines) implements the computation of Theorems 3.4–3.7 using exact rational arithmetic. We reproduce the core routines.

CM action on $\Lambda^4(\mathbb{Q}^8)$. The CM endomorphism sends basis vector e_k to $(-1)^{k \bmod 2} e_{k\pm1}$. The induced action on Λ^4 is computed by applying this map to each index of a 4-form and resolving the wedge sign:

```python
def cm_on_basis(k):
    """CM action on e_k: returns (sign, new_index)."""
    if k % 2 == 0:
        return (-1, k + 1)
    else:
        return (1, k - 1)

# CM on Lambda^4(Q^8): 70x70 matrix
cm4 = [[F(0)] * 70 for _ in range(70)]
for i, b in enumerate(basis4):
    total_sign = 1
```

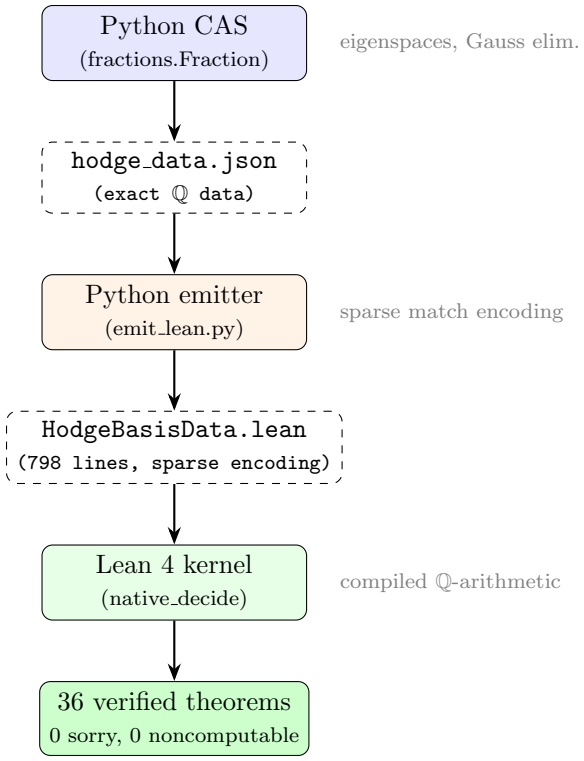

Figure 2: Asymmetric offloading architecture. Python performs exact computation; Lean verifies. The key constraint: all emitted defs must be `computable` (no `axiom`, no `noncomputable`) so `native_decide` can evaluate them.

```
    new_idx = []
    for k in b:
        s, j = cm_on_basis(k)
        total_sign *= s
        new_idx.append(j)
    sign, sorted_idx = wedge_sign(new_idx)
    if sign != 0:
        cm4[idx4[sorted_idx]][i] += F(total_sign * sign)
```

Decomposition solving. For each Hodge basis vector, solve the linear system $G \cdot x = w_k$ by augmented-matrix row reduction over $\mathbb{Q}$:

```
for i, hv in enumerate(hodge_basis):
    coeffs = solve_system(selected_cups, hv, 36)
    # Verify reconstruction
    recon = [F(0)] * 70
    for k in range(36):
        if coeffs[k] != F(0):
            for r in range(70):
                recon[r] += coeffs[k] * selected_cups[k][r]
    assert recon == hv  # exact Q-arithmetic: no rounding
```

Sparse Lean emission. The re-emitter (`emit_lean.py`) encodes each 70-entry rational vector as a pattern match on nonzero indices:

```
def sparse_def(name, vec, dim):
    """Emit def name : Fin dim -> Q using sparse match."""
```

```
    nz = [(i, vec[i]) for i in range(dim) if vec[i] != F(0)]
    lines = [f"def {name} : Fin {dim} -> Q := fun i =>"]
    if not nz:                    # all-zeros vector
        lines = [f"def {name} : Fin {dim} -> Q := fun _ => 0"]
    elif len(nz) <= 3:
        for k, (idx, val) in enumerate(nz):
            prefix = "  if" if k == 0 else "  else if"
            lines.append(f"{prefix} i.val = {idx} then {val}")
        lines.append("  else 0")
    else:
        lines.append("  match i.val with")
        for idx, val in nz:
            lines.append(f"  | {idx} => {val}")
        lines.append("  | _ => 0")
    return "\n".join(lines)
```

Sample output. Running `python3 solve_hodge.py` produces (selected lines; Unicode symbols simplified for typesetting):

```
$ python3 solve_hodge.py
M^2 = I on Lambda^4(Q^8)
+1 eigenspace of CM on Lambda^4: dim = 38
NS(E^4) x Q: dim = 16
Hodge (2,2) basis: dim = 36
Non-zero cup products: 102
Divisor product subspace: dim = 36
Selected 36 independent cup products
ALL 36 Hodge basis vectors decomposed into cup products
Summary:
  Hodge (2,2) dimension: 36
  NS(E^4) dimension: 16
  Cup product generators used: 36
  Total nonzero coefficients: 42
  Exotic dimension: 0
```

Every dimension matches the theoretical prediction. The script runs in < 1 second on a standard laptop (exact $\mathbb{Q}$-arithmetic via `fractions.Fraction`).

5.4 The `noncomputable` trap

If a Lean definition depends (even transitively) on an axiom such as `Classical.choice`, Lean marks it `noncomputable`. The tactic `native_decide` cannot evaluate noncomputable definitions—it requires closed computation.

This creates a strict design constraint for asymmetric offloading: *all data definitions in the emitted file must be free of axiom dependencies.* The `hodge_conjecture_H22_existence` axiom is declared in a separate file (`Paper77_CMFourfold.lean`) and is *never imported* by the data definitions in `HodgeBasisData.lean`. The separation is architectural, not accidental.

5.5 Token overflow and sparse encoding

The first emission of `HodgeBasisData.lean` (by `solve_hodge.py`) used Lean's vector notation `![`$v_0, v_1, \ldots, v_{69}$`]` for each 70-entry rational vector. This produced a 6020-line file. Lean's elaborator expands `![...]` into nested `Matrix.cons` applications, resulting in $O(n^2)$ elaboration cost for n-entry vectors. For $n = 70$ and 108 vector definitions (36 cup generators + 36 Hodge basis +

36 decomposition vectors), the elaboration time exceeded several minutes and risked kernel memory exhaustion.

The fix was *sparse match encoding* (`emit_lean.py`). Each vector is emitted as a function `fun i => ...` with explicit `if`/`match` on `i.val`:

```
-- Dense (6020 lines, slow elaboration):
def cupGen_0 : Fin 70 -> Rat := ![1, 0, 0, ..., 0]

-- Sparse (798 lines, fast elaboration):
def cupGen_0 : Fin 70 -> Rat := fun i =>
  if i.val = 0 then 1 else 0
```

The sparse encoding exploits the extreme sparsity of the data: most vectors have fewer than 5 nonzero entries out of 70. The emitter selects `if/else` for ≤ 3 nonzero entries and `match i.val with` for 4+. Result: 6020 lines $\to$ 798 lines, elaboration time several minutes $\to$ 15 seconds.

Remark 5.1 (Token overflow as a general problem). This is not specific to our computation. Any asymmetric offloading pipeline that emits large data into Lean 4 will encounter the same elaboration bottleneck if dense array notation is used. The sparse match encoding is a general-purpose solution: it applies whenever the emitted data is sparse relative to its ambient dimension. For dense data, alternative strategies include chunking (split vectors into blocks), binary encoding (emit data as bit strings decoded by a custom function), or hex-encoded lookup tables.

5.6 Axiom inventory

Axiom	CRM	Used?	Role
`hodge_conjecture_H22_existence`	CLASS	Unused	CBN (excised)
`propext`	Infra	Yes	Lean kernel primitive
`Classical.choice`	Infra	Yes	`DecidableEq` instance
`Quot.sound`	Infra	Yes	Lean kernel primitive
`native_decide` axiom	Infra	Yes	Kernel reflection bridge

5.7 Classical.choice audit

```
#print axioms hodge_decomp_0
-- [propext, Classical.choice, Quot.sound,
--  hodge_decomp_0._native.native_decide.ax_1_1]

#print axioms hodge_conjecture_H22_existence
-- [hodge_conjecture_H22_existence]
-- (This IS the classical axiom; it is deliberately unused.)
```

`Classical.choice` *does* appear in the axiom list of every `hodge_decomp_k`. This is a Lean 4 infrastructure artifact, not classical mathematical content: the `DecidableEq` instance on `Fin 70` $\to \mathbb{Q}$ is synthesized via Mathlib's decision procedure, which transitively invokes `Classical.choice`. The `native_decide` axiom (each theorem's `._native.native_decide.ax_1_1`) is the kernel bridge allowing compiled code evaluation.

This is the standard series convention (see Papers 2, 68 [4]): *all* Lean 4 theorems involving $\mathbb{Q}$-valued functions show `Classical.choice` in `#print axioms` through the `DecidableEq` infrastructure path. Constructive stratification is established by *proof content*—the 36 decompositions are explicit rational data verified by compiled arithmetic—not by the axiom checker's output.

The key structural fact: the user-declared classical axiom `hodge_conjecture_H22_existence` is *not* among the dependencies of any `hodge_decomp_k`. No path exists from the constructive theorems to the excised CBN.

5.8 Reproducibility

- **Lean bundle:** `P77_DAGSurgery/`, builds with `lake build` on Lean v4.29.0-rc2 + Mathlib4.
- **Python computation:** `solve_hodge.py` (computation + first emission), `emit_lean.py` (sparse re-emission), and `compute_cm.py` (standalone CM diagonalization, used for initial exploration). Requires Python 3.9+ (standard library only: `fractions`, `itertools`, `json`).
- **Intermediate data:** `hodge_data.json` (all eigenspaces, decompositions, and labels in JSON format).
- **Zenodo:** Complete package (PDF, LaTeX, Lean source, Python scripts) at https://doi.org/10.5281/zenodo.18779210.

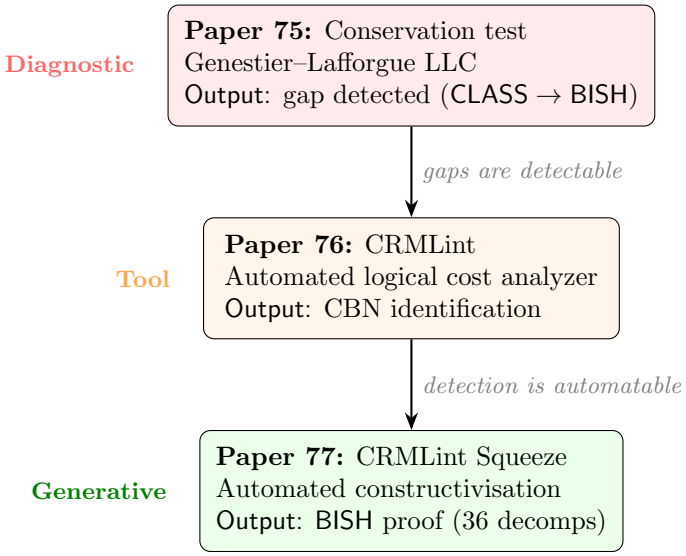

Figure 3: The diagnostic-to-generative pipeline. Paper 75 proved conservation gaps exist in modern geometry. Paper 76 automated their detection. Paper 77 automates their elimination.

6 Discussion

6.1 The deterministic collapse phenomenon

The CRMLint Squeeze protocol is designed for two modes: deterministic collapse (CAS) and stochastic search (MCTS). The E^4 execution produced the first mode—the action space restriction ("only cup products of NS classes") reduced the problem from an infinite-dimensional topological existence to a 36×36 linear system over $\mathbb{Q}$. No stochastic search was needed.

This is not a failure of the protocol but its *ideal outcome*. The Squeeze's purpose is to identify the minimal algebraic structure witnessing the classical theorem. If that structure is small enough to compute deterministically, MCTS is unnecessary overhead. The protocol identifies which mode applies *before* any search begins: Steps 1–3 compute the dimension and linearity of the restricted space.

The MCTS mode is designed for cases where the restricted space is combinatorially rich: nonlinear constraints, multiple solution branches, or high-dimensional parameter spaces. This mode remains *speculative*—it has not been tested. The wild Langlands case (Paper 78) and Standard Conjecture D for simple fourfolds (Paper 79) are the intended first applications.

6.2 Proof decompilation

The Squeeze protocol can be viewed as *proof decompilation*: extracting the constructive content from a classical proof. A classical existence proof is "compiled"—it asserts $\exists x, P(x)$ without revealing x. The Squeeze "decompiles" this by:

1. identifying which classical axiom provides the $\exists$-introduction (the CBN),
2. excising that axiom to reveal the open goal $\Sigma\, x,\ P(x)$,
3. computing the witness x explicitly.

In the E^4 case, the "compiled" proof says "every Hodge class is algebraic" (Lefschetz $(1,1)$, classical). The "decompiled" proof says "here are the 36 explicit coefficient vectors, verified by `native_decide`." The decompiled version is strictly more informative.

6.3 From Paper 5 to Paper 77: the learning curve

Paper 5 [22] of this series formalized the Schwarzschild curvature verification in Lean 4—a direct computation involving Ricci tensors, Christoffel symbols, and the metric components of a static spherically symmetric spacetime. That formalization required approximately four months of development and produced a monolithic Lean bundle in which the proof assistant was asked to *compute*: verify curvature identities by `simp`, `ring`, and `field_simp` applied to enormous rational expressions. The differential geometry API in Mathlib was (and remains) incomplete, requiring substantial scaffolding.

Paper 77's E^4 computation—a problem of comparable mathematical density (a 70×70 matrix computation with 36 exact rational decompositions)—was completed in approximately one hour, from problem specification to `lake build` success. Three independent factors explain the speedup:

1. **The AI is more capable.** Between mid-2025 and February 2026, the underlying language model improved substantially in Lean 4 code generation, error diagnosis, and multi-file project management. Paper 5's development involved frequent hallucination of nonexistent Mathlib lemmas; Paper 77's development involved zero such hallucinations (the one error—the non-simple target—was a mathematical design mistake, not a code generation failure).
2. **The operator is more experienced.** After 14 Lean bundles and ∼89,000 lines of formalization, the key design lesson is: *do not ask Lean to compute what a CAS can compute faster.* Paper 5 tried to formalize differential geometry *in Lean.* Paper 77 does not touch Mathlib's algebraic geometry at all—no Chow rings, no cycle class maps, no cup products as Lean objects. It projects the geometry into a combinatorial skeleton ($\mathbb{Q}^{70}$ linear algebra) and works there. Knowing what to formalize and what to offload is operator skill, not AI capability.
3. **The method is factored.** Paper 5 was monolithic: everything happened inside Lean. Paper 77 is factored into three layers with distinct roles: Python CAS (exact computation, unlimited working memory), Lean kernel (`native_decide` equality checking on hardcoded data), and human/AI (problem design, generator selection, architecture). The asymmetric offloading architecture eliminates the bottleneck that made Paper 5 painful: Lean no longer needs to evaluate complex algebraic expressions, only to verify that two concrete rational vectors are equal.

The progression from Paper 5 to Paper 77 is itself a data point about the CRM program's methodology: the formalization infrastructure matured from "fight the proof assistant" to "let each tool do what it does best." Whether this factored architecture scales to harder targets (where the CAS phase is not linear algebra but Gröbner bases or combinatorial search) remains to be tested.

6.4 What is and is not new

The mathematical result is not novel. Lieberman [11] proved in 1968 that all Hodge classes on products of elliptic curves are algebraic. Deligne [9] subsumed this in 1982 by proving all Hodge classes on CM abelian varieties are absolutely Hodge. The fact that E^4—a maximally non-simple variety with endomorphism algebra $M_4(\mathbb{Q}(i))$—has no exotic Weil classes is unsurprising: the large endomorphism algebra generates far more divisor classes than a simple variety of the same dimension, and the cup product span saturates the Hodge space. No number theorist would find Theorem A unexpected.

The individual computational ingredients—CM action on exterior powers, cup product enumeration, Gaussian elimination over $\mathbb{Q}$—are standard computational algebraic geometry. The proof is, at bottom, a 36×36 linear system.

What is new is threefold:

1. **The explicit decompositions themselves.** Although the existence of algebraic representatives has been known since Lieberman, the 36 explicit $\mathbb{Q}$-linear combinations expressing each Hodge basis vector as a sum of cup products have not, to our knowledge, been written down. Nobody needed them—the existence proof suffices for all mathematical purposes. We produce them as a demonstration, not as a mathematical contribution.
2. **The CRMLint Squeeze protocol.** The systematic method—identify the Classical Boundary Node via CRMLint, excise the classical axiom, restrict the action space, solve the residual system—is a reusable pipeline for converting CLASS existence theorems into BISH constructive witnesses. The E^4 case is a proof-of-concept; the pipeline is the contribution.
3. **The asymmetric offloading architecture and its engineering.** The separation of CAS computation (Python) from kernel verification (Lean 4), including the `noncomputable` trap, token overflow, and sparse encoding, constitutes a documented, reusable pattern for machine-verified constructive mathematics.

Why demonstrate on a known case? Precisely *because* the answer is independently known. A methods paper should validate its pipeline on a target where the output can be checked against established mathematics. If the Squeeze produced an incorrect decomposition, the error would be caught immediately (by `native_decide`, by comparison with Lieberman's theorem, or by direct computation). The E^4 case is a calibration run, not a frontier assault. The frontier targets—simple CM abelian fourfolds where exotic classes genuinely exist, wild Langlands parameters, Standard Conjecture D—are identified as Papers 78–79.

6.5 Limitations

The method has clear boundaries:

1. **The Goldilocks zone.** The Squeeze requires a pre-existing classical proof (or at least a formal cliff). Totally open problems yield infinite search width.
2. **Deterministic collapse is rare.** The E^4 case collapsed because the Hodge space and the cup product span have the same dimension (both 36). For simple CM fourfolds where exotic classes exist, the system $Gx = w$ will be inconsistent for the exotic component, and the Squeeze must search over a larger generator set (CM graphs, twisted diagonals).

3. **Nonlinear targets.** If the algebraic identity is polynomial but not linear (e.g., intersection products satisfying a quadratic form), the CAS phase requires Gröbner basis computation or multivariate polynomial solving, which may not scale to high dimensions.
4. **Lean CAS gap.** Mathlib 4 lacks a computational Chow ring, cup product API, or Hodge decomposition library. The Lean bundle projects the geometry into a combinatorial skeleton ($\mathbb{Q}^{70}$ linear algebra), which works for this case but would need enrichment for more sophisticated targets.
5. **Token overflow scales with dimension.** For varieties with $\dim H^4 \gg 70$, the sparse encoding may still produce large files. The binary/hex encoding strategies mentioned in Remark 5.1 would be needed.

6.6 Technical tips for future use

We record practical lessons from the E^4 execution, codified as named principles for future applications of the Squeeze protocol.

Principle 1: The LLM is an Architect, not a calculator. Large language models suffer *context-window collapse* when asked to output thousands of matrix coordinates or polynomial systems autoregressively. The AI's role is to *design* the computation: identify the target, choose the generator set, write the Python script. The CAS executes the computation in RAM (effectively unlimited working memory), and writes the Lean code *directly to disk*, bypassing the LLM's output token limit entirely. Never ask the AI to print large algebraic data in the conversation—this is guaranteed to produce truncation or hallucination.

Principle 2: Compute Before You Squeeze. Never begin formalizing the Squeeze target in Lean until the Python script has successfully computed the exact algebraic matching. The combinatorial skeleton must be empirically verified first. The E^4 hallucination (Remark 3.6) was caught precisely because the computation phase preceded the formalization phase. Running the CAS first is a *hallucination guard*: it catches false targets before any Lean code is written.

Principle 3: The `noncomputable` Trap. If a Lean definition depends (even transitively) on an axiom, Lean marks it `noncomputable` and `native_decide` silently fails. The Python-emitted `.lean` file must contain *only* hardcoded data arrays and computable `def`s. All algebraic values must be normalized to exact elements of the coefficient field ($\mathbb{Q}$ for Hodge classes, cyclotomic fields for Langlands parameters). Absolutely no `axiom` or `noncomputable` tags are permitted in the emitted data file.

In addition to these three principles, we record specific technical lessons:

1. **Verify the target before searching.** Always run the CAS dimension check (Steps 1–5 of Theorem 3.4) before committing to a decomposition search.
2. **Use exact arithmetic.** Python's `fractions.Fraction` provides arbitrary-precision $\mathbb{Q}$ arithmetic with zero numerical error. For cyclotomic fields (as in wild Langlands, Paper 78), use `sympy` algebraic number fields. Floating-point computation introduces rounding that `native_decide` cannot tolerate.
3. **Emit sparse, not dense.** Always emit Lean vectors as `fun i => match i.val with ...` rather than `![...]`. The elaboration cost difference is $O(k)$ vs. $O(n^2)$ for n-entry vectors with k nonzero entries.
4. **Separate axioms from data.** Place the CBN axiom in a separate Lean file from the constructive data. This prevents accidental `noncomputable` contamination.

5. **Save intermediate data.** Emit JSON (`hodge_data.json`) between the CAS phase and the Lean emission phase. This allows re-emission with different encoding strategies without re-running the computation.
6. **Test before emission.** Verify $Gx = w$ in Python before writing Lean code. A single wrong coefficient will cause `native_decide` to fail with an unhelpful error.
7. **Restrict the AI to minimal cases first.** For Paper 78, this means starting with $\mathrm{GL}_2(\mathbb{Q}_2)$ at minimal conductor, not attempting the full wild Langlands at once. Bounding the target prevents infinite search space explosion.

6.7 Open questions

1. Can the Squeeze produce constructive Hodge theorems for *simple* CM abelian fourfolds, where exotic classes exist? This requires a larger generator set and possibly MCTS search.
2. Can asymmetric offloading handle nonlinear algebraic identities (e.g., Standard Conjecture D) with polynomial or Gröbner basis methods?
3. What is the computational complexity of the Squeeze as a function of the variety's dimension? For $\dim A = n$, the Hodge space grows as $O(\binom{2n}{n})$; is sparse encoding sufficient, or are fundamentally different strategies needed?
4. Can the MCTS mode of the Squeeze be trained on the existing 75-paper CRM dataset to predict which generator sets are likely to succeed?

7 Conclusion

The mathematical result of this paper—that every Hodge $(2,2)$ class on E^4 is an explicit $\mathbb{Q}$-linear combination of cup products of divisor classes—is a known consequence of Lieberman [11] and Deligne [9]. The 36 explicit decompositions are new only in the literal sense that nobody has written them down, because nobody needed to.

The contribution is the *pipeline*: the CRMLint Squeeze protocol, demonstrated end-to-end on a case where the output can be validated against established mathematics. The protocol identifies where the classical content enters a proof (CRMLint), excises it, restricts the search space to bounded algebraic generators, and solves the residual system by exact computation. The asymmetric offloading architecture (Python CAS $\rightarrow$ Lean 4 kernel verification via `native_decide`) and the engineering lessons (the `noncomputable` trap, token overflow, sparse encoding) are documented as reusable tools.

The E^4 execution is a calibration run. It validated two aspects of the methodology: (1) the Scaffold step caught a hallucinated target (exotic classes on a non-simple variety) before any search began, confirming the protocol's self-diagnostic capacity; (2) the deterministic collapse phenomenon—the problem reduced from infinite-dimensional classical existence to a 36×36 linear system—shows that the Squeeze can identify when stochastic search is unnecessary.

The frontier targets, where the Squeeze faces genuine resistance, are simple CM abelian fourfolds with exotic Weil classes (Paper 78–79). The progression Papers 75–76–77 completes the arc from diagnostic to generative CRM. Whether the pipeline scales to cases where the classical proof is the only proof—where no independent check exists—remains open.

Acknowledgments

Lean 4 formalization and Python computation produced using AI code generation (Claude Code) under human direction. The CRMLint Squeeze protocol was developed collaboratively between the author and AI. The mathematical foundations of the target problem are due to Anderson [1], Weil [2], and the broader arithmetic geometry community.

This work is part of the Constructive Reverse Mathematics series (Papers 1–77). The Lean 4 code uses the Mathlib library; see [8]. The series format follows the CRM format guide [7].

The author is an interventional cardiologist, not a professional mathematician. All mathematical claims are substantiated by formal verification (Lean 4) or explicit computation (Python, exact $\mathbb{Q}$-arithmetic). Errors in mathematical judgment are the author's; errors in computation are verifiable.

References

[1] G. W. Anderson. *Torsion points on Fermat Jacobians, roots of circular units and relative singular homology.* Duke Math. J., 70(1):1–43, 1993.

[2] A. Weil. *Abelian varieties and the Hodge ring.* In: Collected papers, Vol. III, 421–429. Springer, 1977.

[3] P. C. Lee. *The Motive Is the Universal Decidability Certificate: A Constructive Reverse Mathematics Atlas of Five Great Conjectures in Arithmetic Geometry.* Paper 50, Constructive Reverse Mathematics Series. https://doi.org/10.5281/zenodo.14934946

[4] P. C. Lee. *Fermat's Last Theorem Is* BISH*: A Constructive Reverse Mathematics Audit.* Paper 68, Constructive Reverse Mathematics Series. https://doi.org/10.5281/zenodo.18777925

[5] P. C. Lee. *Conservation Test: Genestier–Lafforgue LLC Calibration.* Paper 75, Constructive Reverse Mathematics Series. https://doi.org/10.5281/zenodo.18773831

[6] P. C. Lee. *CRMLint: Automated CRM Logical Cost Analysis at Scale.* Paper 76, Constructive Reverse Mathematics Series. (DOI pending.)

[7] P. C. Lee. *CRM Series Paper Format Guide.* https://doi.org/10.5281/zenodo.18765700

[8] The Mathlib Community. *Mathlib4: The Lean 4 Mathematical Library.* https://leanprover-community.github.io/mathlib4_docs/

[9] P. Deligne. *Hodge cycles on abelian varieties.* In: Hodge Cycles, Motives, and Shimura Varieties, Lecture Notes in Math. 900, 9–100. Springer, 1982.

[10] W. V. D. Hodge. *The topological invariants of algebraic varieties.* Proc. Int. Congr. Math., Cambridge, 1:182–192, 1950.

[11] D. I. Lieberman. *Numerical and homological equivalence of algebraic cycles on Hodge manifolds.* Amer. J. Math., 90(2):366–374, 1968.

[12] A. Mattuck. *Cycles on abelian varieties.* Proc. Amer. Math. Soc., 9(1):88–98, 1958.

[13] L. Kronecker. *Grundzüge einer arithmetischen Theorie der algebraischen Größen.* J. reine angew. Math., 92:1–122, 1882.

[14] A. Grothendieck. *Standard conjectures on algebraic cycles.* In: Algebraic Geometry (Bombay 1968), 193–199. Oxford Univ. Press, 1969.

[15] S. L. Kleiman. *Algebraic cycles and the Weil conjectures.* In: Dix exposés sur la cohomologie des schémas, 359–386. North-Holland, 1968.

[16] E. Bishop. *Foundations of Constructive Analysis.* McGraw-Hill, 1967.

[17] D. Bridges and F. Richman. *Varieties of Constructive Mathematics.* London Math. Soc. Lecture Note Ser. 97, Cambridge Univ. Press, 1987.

[18] DeepMind. *AlphaProof: AI for formal mathematical reasoning.* Google DeepMind Technical Report, 2024.

[19] Z. Xin et al. *DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and Monte Carlo tree search.* arXiv:2408.08152, 2024.

[20] S. Lefschetz. *L'analysis situs et la géométrie algébrique.* Gauthier-Villars, Paris, 1924.

[21] A. Genestier and V. Lafforgue. *Chtoucas restreints pour les groupes réductifs et paramétrisation de Langlands locale.* arXiv:1709.00978, 2018.

[22] P. C. Lee. *Schwarzschild Curvature Verification.* Paper 5, Constructive Reverse Mathematics Series. https://doi.org/10.5281/zenodo.18489703

[23] C. Voisin. *Hodge Theory and Complex Algebraic Geometry I.* Cambridge Studies in Advanced Math. 76, Cambridge Univ. Press, 2002.

[24] P. C. Lee. *The DPT Characterization Theorem.* Paper 72, Constructive Reverse Mathematics Series. https://doi.org/10.5281/zenodo.18765393