The Complexity of Decision-Relevant Uncertainty: Why Identifying What Matters Is Harder Than Knowing Everything

Tristan Simas
McGill University
tristan.simas@mail.mcgill.ca

January 3, 2026

Abstract

For decision problems with state space $S = X_1 \times \cdots \times X_n$, a coordinate set $I \subseteq \{1, \ldots, n\}$ is *sufficient* if states agreeing on I have identical optimal action sets. We prove:

Theorem (SUFFICIENCY-CHECK is coNP-complete). Given decision problem (A, S, U) and coordinate set I, determining whether I is sufficient is coNP-complete [4, 7].

Theorem (MINIMUM-SUFFICIENT-SET is coNP-complete). Finding the minimum sufficient coordinate set is coNP-complete. The problem has apparent Σ_2^{P} structure ($\exists I\ \forall s, s'$), but collapses to coNP because sufficiency equals "superset of relevant coordinates."

Theorem (ANCHOR-SUFFICIENCY is Σ_2^{P}-complete). Given fixed coordinate set I, determining whether there exists an assignment to I making the optimal action constant on the induced subcube is Σ_2^{P}-complete [14].

Theorem (Complexity Dichotomy). Sufficiency checking is polynomial-time when the minimal sufficient set has size $O(\log |S|)$, exponential-time when size is $\Omega(n)$.

All proofs are machine-checked in Lean 4 (3,500+ lines, 28 files, 0 sorry). The formalization uses a general Boolean Formula AST for the coNP and Σ_2^P reductions.

Keywords: computational complexity, decision theory, model selection, coNP-completeness, polynomial hierarchy, Lean 4

1 Introduction

Consider a decision problem with actions A and states $S = X_1 \times \cdots \times X_n$ (a product of n coordinate spaces). For each state $s \in S$, some subset $\text{Opt}(s) \subseteq A$ of actions maximize utility. A coordinate set $I \subseteq \{1, \ldots, n\}$ is *sufficient* if:

$$s_I = s'_I \implies \text{Opt}(s) = \text{Opt}(s')$$

where s_I denotes the projection of state s onto coordinates in I.

1.1 The Decision Quotient

Define the equivalence relation $s \sim s'$ iff $\mathrm{Opt}(s) = \mathrm{Opt}(s')$. The *decision quotient* is $Q = S/\!\sim$, the quotient of states by decision-equivalence. A coordinate set I is sufficient iff the quotient map $\pi : S \to Q$ factors through the projection $S \to S_I$.

The quotient size $|Q|$ measures decision-relevant complexity: when $|Q| = 1$, all states have identical optimal actions (no coordinates matter); when $|Q| = |S|$, every state requires independent consideration (all coordinates matter).

1.2 The Sufficiency-Checking Problem

Definition 1.1. SUFFICIENCY-CHECK: Given decision problem (A, S, U) and coordinate set $I \subseteq \{1, \ldots, n\}$, determine whether I is sufficient.

Definition 1.2. MINIMUM-SUFFICIENT-SET: Given decision problem (A, S, U) and integer k, determine whether there exists a sufficient set I with $|I| \leq k$.

1.3 Main Results

Theorem 1.3 (SUFFICIENCY-CHECK is coNP-complete)**.** *SUFFICIENCY-CHECK is coNP-complete [4, 7].*

Proof sketch. Membership: witness for non-sufficiency is a pair (s, s') with $s_I = s'_I$ but $\mathrm{Opt}(s) \neq \mathrm{Opt}(s')$, verifiable in polynomial time. Hardness: polynomial-time reduction from TAUTOLOGY. Given Boolean formula φ, construct decision problem where Opt is constant iff φ is a tautology. Full proof in Section 3.

Theorem 1.4 (MINIMUM-SUFFICIENT-SET is coNP-complete)**.** *MINIMUM-SUFFICIENT-SET is coNP-complete. The problem has apparent Σ_2^P quantifier structure ($\exists I(|I| \leq k)\ \forall s, s' : s_I = s'_I \implies Opt(s) = Opt(s')$), but collapses to coNP.*

Proof sketch. A coordinate i is *relevant* if there exist states s, s' differing only at i with $\mathrm{Opt}(s) \neq \mathrm{Opt}(s')$. The key structural result is: I is sufficient iff I contains all relevant coordinates (Theorem 1.8). Therefore the minimum sufficient set equals the set of relevant coordinates. Checking relevance is in coNP, so MINIMUM-SUFFICIENT-SET collapses to coNP. The $k = 0$ case reduces to SUFFICIENCY-CHECK, establishing hardness. Full proof in Section 3.

Theorem 1.5 (ANCHOR-SUFFICIENCY is Σ_2^{P}-complete)**.** *ANCHOR-SUFFICIENCY is Σ_2^P-complete [14].*

Proof sketch. Membership: $\exists \alpha\ \forall s : (s_I = \alpha \implies \mathrm{Opt}(s) = \mathrm{Opt}(s_0))$ for fixed s_0. Hardness: reduction from $\exists\forall$-SAT. Given $\exists x \forall y\, \varphi(x, y)$, construct decision problem where an anchor exists for the x-coordinates iff the formula is satisfiable. Full proof in Section 3.

Theorem 1.6 (Complexity Dichotomy)**.** *Let k^* be the size of the minimal sufficient set. Then:*

1. *If $k^* = O(\log |S|)$, sufficiency checking is polynomial-time*
2. *If $k^* = \Omega(n)$, sufficiency checking requires exponential time (assuming standard complexity conjectures)*

Theorem 1.7 (Tractable Subcases)**.** *Sufficiency checking is polynomial-time for:*

1. *Bounded action sets:* $|A| \leq k$ *for constant* k
2. *Separable utilities:* $U(a, s) = f(a) + g(s)$
3. *Tree-structured coordinate dependencies*

1.4 The Σ_2^{P} to coNP Collapse

MINIMUM-SUFFICIENT-SET has the syntactic form of a Σ_2^{P} problem: $\exists I\ \forall s, s'$. However, it is coNP-complete, not Σ_2^{P}-complete. The collapse occurs because sufficiency has an alternate characterization that eliminates the existential quantifier:

Theorem 1.8 (Sufficient iff Superset of Relevant)**.** *In a product space* $S = X_1 \times \cdots \times X_n$*, coordinate set* I *is sufficient if and only if* $I \supseteq$ *Relevant, where Relevant* $= \{i : \exists s, s'$ *differing only at* i *with* $Opt(s) \neq Opt(s')\}$*.*

This theorem (proven in Lean as `minimalSufficient_iff_relevant`) shows the minimum sufficient set is *uniquely determined*: it equals the set of relevant coordinates. Therefore MINIMUM-SUFFICIENT-SET asks "Does Relevant have size $\leq k$?" which is in coNP.

ANCHOR-SUFFICIENCY retains its Σ_2^{P}-completeness because the existential quantifier ranges over *assignments*, not coordinate sets. The "for all suffixes" quantifier cannot be eliminated when the anchor assignment is part of the existential choice.

1.5 Connection to Prior Papers

This paper continues a series on the complexity of architectural decisions. Paper 1 introduced the language of configuration spaces. Paper 2 formalized behavioral equivalence. Paper 3 developed leverage theory. Here we provide the complexity-theoretic foundations that explain why minimal modeling is computationally hard.

1.6 Machine-Checked Proofs

All theorems are formalized in Lean 4 with complete machine-checked proofs:

- Location: `docs/papers/paper4_decision_quotient/proofs/`
- Total lines: 3,500+ across 28 files
- Theorems proven: ∼65 (13 core results, 52 supporting lemmas)
- Sorry placeholders: 0

The formalization includes:

1. General Boolean Formula AST for TAUTOLOGY reduction (not CNF, which is in P)
2. Quantified Boolean Formula encoding for $\exists\forall$-SAT reduction
3. Product space structure with hybrid state construction
4. Polynomial-time reduction composition (200-line proof of closure under composition)
5. Decidable sufficiency checking algorithm with correctness proof

The Lean compiler verifies all proof steps, all definitions are consistent, and no axioms are added beyond Lean's foundations (extended with Mathlib for basic combinatorics)

Section 2 establishes formal foundations: decision problems, coordinate spaces, sufficiency. Section 3 proves hardness results with complete reductions. Section 4 develops the complexity dichotomy. Section 5 presents tractable special cases. Section 7 discusses implications for software architecture and modeling. Section 8 surveys related work. Appendix A contains Lean proof listings.

2 Formal Foundations

We formalize decision problems with coordinate structure, sufficiency of coordinate sets, and the decision quotient, drawing on classical decision theory [12, 11].

2.1 Decision Problems with Coordinate Structure

Definition 2.1 (Decision Problem)**.** A *decision problem with coordinate structure* is a tuple $\mathcal{D} = (A, X_1, \ldots, X_n, U)$ where:

- A is a finite set of *actions* (alternatives)
- $X_1, \ldots, X_n$ are finite *coordinate spaces*
- $S = X_1 \times \cdots \times X_n$ is the *state space*
- $U : A \times S \to \mathbb{Q}$ is the *utility function*

Definition 2.2 (Projection)**.** For state $s = (s_1, \ldots, s_n) \in S$ and coordinate set $I \subseteq \{1, \ldots, n\}$:

$$s_I := (s_i)_{i \in I}$$

is the *projection* of s onto coordinates in I.

Definition 2.3 (Optimizer Map)**.** For state $s \in S$, the *optimal action set* is:

$$\mathrm{Opt}(s) := \arg\max_{a \in A} U(a, s) = \{a \in A : U(a, s) = \max_{a' \in A} U(a', s)\}$$

2.2 Sufficiency and Relevance

Definition 2.4 (Sufficient Coordinate Set)**.** A coordinate set $I \subseteq \{1, \ldots, n\}$ is *sufficient* for decision problem $\mathcal{D}$ if:

$$\forall s, s' \in S : \quad s_I = s'_I \implies \mathrm{Opt}(s) = \mathrm{Opt}(s')$$

Equivalently, the optimal action depends only on coordinates in I.

Definition 2.5 (Minimal Sufficient Set)**.** A sufficient set I is *minimal* if no proper subset $I' \subsetneq I$ is sufficient.

Definition 2.6 (Relevant Coordinate)**.** Coordinate i is *relevant* if it belongs to some minimal sufficient set.

Example 2.7 (Weather Decision)**.** Consider deciding whether to carry an umbrella:

- Actions: $A = \{\text{carry}, \text{don't carry}\}$

- Coordinates: $X_1 = \{\text{rain}, \text{no rain}\}$, $X_2 = \{\text{hot}, \text{cold}\}$, $X_3 = \{\text{Monday}, \ldots, \text{Sunday}\}$
- Utility: $U(\text{carry}, s) = -1 + 3 \cdot \mathbf{1}[s_1 = \text{rain}]$, $U(\text{don't carry}, s) = -2 \cdot \mathbf{1}[s_1 = \text{rain}]$

The minimal sufficient set is $I = \{1\}$ (only rain forecast matters). Coordinates 2 and 3 (temperature, day of week) are irrelevant.

2.3 The Decision Quotient

Definition 2.8 (Decision Equivalence)**.** For coordinate set I, states s, s' are *I-equivalent* (written $s \sim_I s'$) if $s_I = s'_I$.

Definition 2.9 (Decision Quotient)**.** The *decision quotient* for state s under coordinate set I is:

$$\text{DQ}_I(s) = \frac{|\{a \in A : a \in \text{Opt}(s') \text{ for some } s' \sim_I s\}|}{|A|}$$

This measures the fraction of actions that *could* be optimal given only the information in I.

Proposition 2.10 (Sufficiency Characterization)**.** *Coordinate set I is sufficient if and only if $DQ_I(s) = |Opt(s)|/|A|$ for all $s \in S$.*

Proof. If I is sufficient, then $s \sim_I s' \implies \text{Opt}(s) = \text{Opt}(s')$, so the set of actions optimal for some $s' \sim_I s$ is exactly $\text{Opt}(s)$.

Conversely, if the condition holds, then for any $s \sim_I s'$, the optimal actions form the same set (since $\text{DQ}_I(s) = \text{DQ}_I(s')$ and both equal the relative size of the common optimal set).

3 Computational Complexity of Decision-Relevant Uncertainty

This section establishes the computational complexity of determining which state coordinates are decision-relevant. We prove three main results:

1. **SUFFICIENCY-CHECK** is coNP-complete
2. **MINIMUM-SUFFICIENT-SET** is coNP-complete (the Σ_2^P structure collapses)
3. **ANCHOR-SUFFICIENCY** (fixed coordinates) is Σ_2^P-complete

These results sit beyond NP-completeness and formally explain why engineers default to over-modeling: finding the minimal set of decision-relevant factors is computationally intractable.

3.1 Problem Definitions

Definition 3.1 (Decision Problem Encoding)**.** A *decision problem instance* is a tuple (A, n, U) where:

- A is a finite set of alternatives
- n is the number of state coordinates, with state space $S = \{0, 1\}^n$
- $U : A \times S \to \mathbb{Q}$ is the utility function, given as a Boolean circuit

Definition 3.2 (Optimizer Map)**.** For state $s \in S$, define:

$$\mathrm{Opt}(s) := \arg\max_{a \in A} U(a, s)$$

Definition 3.3 (Sufficient Coordinate Set)**.** A coordinate set $I \subseteq \{1, \ldots, n\}$ is *sufficient* if:

$$\forall s, s' \in S : \quad s_I = s'_I \implies \mathrm{Opt}(s) = \mathrm{Opt}(s')$$

where s_I denotes the projection of s onto coordinates in I.

Problem 3.4 (SUFFICIENCY-CHECK)**. Input:** Decision problem (A, n, U) and coordinate set $I \subseteq \{1, \ldots, n\}$
Question: Is I sufficient?

Problem 3.5 (MINIMUM-SUFFICIENT-SET)**. Input:** Decision problem (A, n, U) and integer k
Question: Does there exist a sufficient set I with $|I| \leq k$?

3.2 Hardness of SUFFICIENCY-CHECK

Theorem 3.6 (coNP-completeness of SUFFICIENCY-CHECK)**.** *SUFFICIENCY-CHECK is coNP-complete [4, 7].*

Proof. **Membership in coNP:** The complementary problem INSUFFICIENCY is in NP. Given (A, n, U, I), a witness for insufficiency is a pair (s, s') such that:

1. $s_I = s'_I$ (verifiable in polynomial time)
2. $\mathrm{Opt}(s) \neq \mathrm{Opt}(s')$ (verifiable by evaluating U on all alternatives)

coNP-hardness: We reduce from TAUTOLOGY.
Given Boolean formula $\varphi(x_1, \ldots, x_n)$, construct a decision problem with:

- Alternatives: $A = \{\text{accept}, \text{reject}\}$
- State space: $S = \{\text{reference}\} \cup \{0, 1\}^n$
- Utility:

$$\begin{aligned} U(\text{accept}, \text{reference}) &= 1 \\ U(\text{reject}, \text{reference}) &= 0 \\ U(\text{accept}, a) &= \varphi(a) \\ U(\text{reject}, a) &= 0 \quad \text{for assignments } a \in \{0, 1\}^n \end{aligned}$$

- Query set: $I = \emptyset$

Claim: $I = \emptyset$ is sufficient $\iff$ φ is a tautology.
($\Rightarrow$) Suppose I is sufficient. Then $\mathrm{Opt}(s)$ is constant over all states. Since $U(\text{accept}, a) = \varphi(a)$ and $U(\text{reject}, a) = 0$:

- $\mathrm{Opt}(a) = \text{accept}$ when $\varphi(a) = 1$
- $\mathrm{Opt}(a) = \{\text{accept}, \text{reject}\}$ when $\varphi(a) = 0$

For Opt to be constant, $\varphi(a)$ must be true for all assignments a; hence φ is a tautology.
($\Leftarrow$) If φ is a tautology, then $U(\text{accept}, a) = 1 > 0 = U(\text{reject}, a)$ for all assignments a. Thus $\mathrm{Opt}(s) = \{\text{accept}\}$ for all states s, making $I = \emptyset$ sufficient.

3.3 Complexity of MINIMUM-SUFFICIENT-SET

Theorem 3.7 (MINIMUM-SUFFICIENT-SET is coNP-complete)**.** *MINIMUM-SUFFICIENT-SET is coNP-complete.*

Proof. **Structural observation:** The $\exists\forall$ quantifier pattern suggests Σ_2^P:

$$\exists I\,(|I| \le k)\;\forall s, s' \in S: \quad s_I = s'_I \implies \mathrm{Opt}(s) = \mathrm{Opt}(s')$$

However, this collapses because sufficiency has a simple characterization.

Key lemma: A coordinate set I is sufficient if and only if I contains all relevant coordinates (proven formally as `sufficient_contains_relevant` in Lean):

$$\mathrm{sufficient}(I) \iff \mathrm{Relevant} \subseteq I$$

where $\mathrm{Relevant} = \{i : \exists s, s'.\ s \text{ differs from } s' \text{ only at } i \text{ and } \mathrm{Opt}(s) \neq \mathrm{Opt}(s')\}$.

Consequence: The minimum sufficient set is exactly the set of relevant coordinates. Thus MINIMUM-SUFFICIENT-SET asks: "Is the number of relevant coordinates at most k?"

coNP membership: A witness that the answer is NO is a set of $k+1$ coordinates, each proven relevant (by exhibiting s, s' pairs). Verification is polynomial.

coNP-hardness: The $k = 0$ case asks whether no coordinates are relevant, i.e., whether $\emptyset$ is sufficient. This is exactly SUFFICIENCY-CHECK, which is coNP-complete by Theorem 3.6.

3.4 Anchor Sufficiency (Fixed Coordinates)

We also formalize a strengthened variant that fixes the coordinate set and asks whether there exists an *assignment* to those coordinates that makes the optimal action constant on the induced subcube.

Problem 3.8 (ANCHOR-SUFFICIENCY)**. Input:** Decision problem (A, n, U) and fixed coordinate set $I \subseteq \{1, \ldots, n\}$
Question: Does there exist an assignment α to I such that $\mathrm{Opt}(s)$ is constant for all states s with $s_I = \alpha$?

Theorem 3.9 (ANCHOR-SUFFICIENCY is Σ_2^P-complete)**.** *ANCHOR-SUFFICIENCY is Σ_2^P-complete [14] (already for Boolean coordinate spaces).*

Proof. **Membership in Σ_2^P:** The problem has the form

$$\exists\alpha\;\forall s \in S:\ (s_I = \alpha) \implies \mathrm{Opt}(s) = \mathrm{Opt}(s_\alpha),$$

which is an $\exists\forall$ pattern.

Σ_2^P-hardness: Reduce from $\exists\forall$-SAT. Given $\exists x\,\forall y\,\varphi(x, y)$ with $x \in \{0,1\}^k$ and $y \in \{0,1\}^m$, if $m = 0$ we first pad with a dummy universal variable (satisfiability is preserved), construct a decision problem with:

- Actions $A = \{\mathrm{YES}, \mathrm{NO}\}$
- State space $S = \{0,1\}^{k+m}$ representing (x, y)
- Utility

$$U(\mathrm{YES}, (x,y)) = \begin{cases} 2 & \text{if } \varphi(x,y) = 1 \\ 0 & \text{otherwise} \end{cases} \qquad U(\mathrm{NO}, (x,y)) = \begin{cases} 1 & \text{if } y = 0^m \\ 0 & \text{otherwise} \end{cases}$$

- Fixed coordinate set I = the x-coordinates.

If $\exists x^\star \, \forall y \, \varphi(x^\star, y) = 1$, then for any y we have $U(\text{YES}) = 2$ and $U(\text{NO}) \leq 1$, so $\text{Opt}(x^\star, y) = \{\text{YES}\}$ is constant. Conversely, if $\varphi(x, y)$ is false for some y, then either $y = 0^m$ (where NO is optimal) or $y \neq 0^m$ (where YES and NO tie), so the optimal set varies across y and the subcube is not constant. Thus an anchor assignment exists iff the $\exists\forall$-SAT instance is true.

3.5 Tractable Subcases

Despite the general hardness, several natural subcases admit efficient algorithms:

Proposition 3.10 (Small State Space)**.** *When $|S|$ is polynomial in the input size (i.e., explicitly enumerable), MINIMUM-SUFFICIENT-SET is solvable in polynomial time.*

Proof. Compute $\text{Opt}(s)$ for all $s \in S$. The minimum sufficient set is exactly the set of coordinates that "matter" for the resulting function, computable by standard techniques.

Proposition 3.11 (Linear Utility)**.** *When $U(a, s) = w_a \cdot s$ for weight vectors $w_a \in \mathbb{Q}^n$, MINIMUM-SUFFICIENT-SET reduces to identifying coordinates where weight vectors differ.*

3.6 Implications

Corollary 3.12 (Why Over-Modeling Is Rational)**.** *Finding the minimal set of decision-relevant factors is coNP-complete. Even* verifying *that a proposed set is sufficient is coNP-complete.*

This formally explains the engineering phenomenon:

1. *It's computationally easier to model everything than to find the minimum*
2. *"Which unknowns matter?" is a hard question, not a lazy one to avoid*
3. *Bounded scenario analysis (small $\hat{S}$) makes the problem tractable*

This connects to the pentalogy's leverage framework: the "epistemic budget" for deciding what to model is itself a computationally constrained resource.

3.7 Remark: The Collapse to coNP

Early analysis of MINIMUM-SUFFICIENT-SET focused on the apparent $\exists\forall$ quantifier structure, which suggested a Σ_2^P-complete result. We initially explored certificate-size lower bounds for the complement, attempting to show MINIMUM-SUFFICIENT-SET was unlikely to be in coNP.

However, the key insight—formalized as `sufficient_contains_relevant`—is that sufficiency has a simple characterization: a set is sufficient iff it contains all relevant coordinates. This collapses the $\exists\forall$ structure because:

- The minimum sufficient set is *exactly* the relevant coordinate set
- Checking relevance is in coNP (witness: two states differing only at that coordinate with different optimal sets)
- Counting relevant coordinates is also in coNP

This collapse explains why ANCHOR-SUFFICIENCY retains its Σ_2^P-completeness: fixing coordinates and asking for an assignment that works is a genuinely different question. The "for all suffixes" quantifier cannot be collapsed when the anchor assignment is part of the existential choice.

4 Complexity Dichotomy

The hardness results of Section 3 apply to worst-case instances. This section develops a more nuanced picture: a *dichotomy theorem* showing that problem difficulty depends on the size of the minimal sufficient set.

Theorem 4.1 (Complexity Dichotomy)**.** *Let $\mathcal{D} = (A, X_1, \ldots, X_n, U)$ be a decision problem with $|S| = N$ states. Let k^* be the size of the minimal sufficient set.*

1. ***Logarithmic case:*** *If $k^* = O(\log N)$, then SUFFICIENCY-CHECK is solvable in polynomial time.*
2. ***Linear case:*** *If $k^* = \Omega(n)$, then SUFFICIENCY-CHECK requires time $\Omega(2^{n/c})$ for some constant $c > 0$ (assuming ETH).*

Proof. **Part 1 (Logarithmic case):** If $k^* = O(\log N)$, then the number of distinct projections $|S_{I^*}|$ is at most $2^{k^*} = O(N^c)$ for some constant c. We can enumerate all projections and verify sufficiency in polynomial time.

Part 2 (Linear case): The reduction from TAUTOLOGY in Theorem 3.6 produces instances where the minimal sufficient set has size $\Omega(n)$ (all coordinates are relevant when the formula is not a tautology). Under the Exponential Time Hypothesis (ETH) [6], TAUTOLOGY requires time $2^{\Omega(n)}$, so SUFFICIENCY-CHECK inherits this lower bound.

Corollary 4.2 (Phase Transition)**.** *There exists a threshold $\tau \in (0, 1)$ such that:*

- *If $k^*/n < \tau$, SUFFICIENCY-CHECK is "easy" (polynomial in N)*
- *If $k^*/n > \tau$, SUFFICIENCY-CHECK is "hard" (exponential in n)*

This dichotomy explains why some domains admit tractable model selection (few relevant variables) while others require heuristics (many relevant variables).

5 Tractable Special Cases

Despite the general hardness, several natural problem classes admit polynomial-time algorithms.

Theorem 5.1 (Tractable Subcases)**.** *SUFFICIENCY-CHECK is polynomial-time solvable for:*

1. ***Bounded actions:*** *$|A| \leq k$ for constant k*
2. ***Separable utility:*** *$U(a, s) = f(a) + g(s)$*
3. ***Tree-structured dependencies:*** *Coordinates form a tree where each coordinate depends only on its ancestors*

5.1 Bounded Actions

Proof of Part 1. With $|A| = k$ constant, the optimizer map $\text{Opt} : S \to 2^A$ has at most 2^k distinct values. For each pair of distinct optimizer values, we can identify the coordinates that distinguish them. The union of these distinguishing coordinates forms a sufficient set.

The algorithm:

1. Sample states to identify distinct optimizer values (polynomial samples suffice with high probability)
2. For each pair of optimizer values, find distinguishing coordinates
3. Return the union of distinguishing coordinates

This runs in time $O(|S| \cdot k^2)$ which is polynomial when k is constant.

5.2 Separable Utility

Proof of Part 2. If $U(a, s) = f(a) + g(s)$, then:

$$\text{Opt}(s) = \arg\max_{a \in A}[f(a) + g(s)] = \arg\max_{a \in A} f(a)$$

The optimal action is independent of the state! Thus $I = \emptyset$ is always sufficient.

5.3 Tree-Structured Dependencies

Proof of Part 3. When coordinates form a tree, we can use dynamic programming. For each node i, compute the set of optimizer values achievable in the subtree rooted at i. A coordinate is relevant if and only if different values at that coordinate lead to different optimizer values in its subtree. This approach is analogous to inference in probabilistic graphical models [10, 8].

The algorithm runs in time $O(n \cdot |A|^2)$ by processing the tree bottom-up.

5.4 Practical Implications

These tractable cases correspond to common modeling scenarios:

- **Bounded actions:** Most real decisions have few alternatives (buy/sell/hold, approve/reject, etc.)
- **Separable utility:** Additive cost models, linear utility functions
- **Tree structure:** Hierarchical decision processes, causal models with tree structure

When a problem falls outside these cases, the hardness results apply, justifying heuristic approaches.

6 Mathematical Justification of Engineering Practice

The complexity results of Sections 3 and 4 provide mathematical grounding for widespread engineering practices. We prove that observed behaviors—configuration over-specification, absence of automated minimization tools, heuristic model selection—are not failures of engineering discipline but rational adaptations to computational constraints.

6.1 Configuration Simplification is SUFFICIENCY-CHECK

Real engineering problems reduce directly to the decision problems studied in this paper.

Theorem 6.1 (Configuration Simplification Reduces to SUFFICIENCY-CHECK)**.** *Given a software system with configuration parameters $P = \{p_1, \ldots, p_n\}$ and observed behaviors $B = \{b_1, \ldots, b_m\}$, the problem of determining whether parameter subset $I \subseteq P$ preserves all behaviors is equivalent to SUFFICIENCY-CHECK.*

Proof. Construct decision problem $\mathcal{D} = (A, X_1, \ldots, X_n, U)$ where:

- Actions $A = B$ (each behavior is an action)
- Coordinates X_i = domain of parameter p_i
- State space $S = X_1 \times \cdots \times X_n$
- Utility $U(b, s) = 1$ if behavior b occurs under configuration s, else $U(b, s) = 0$

Then $\text{Opt}(s) = \{b \in B : b \text{ occurs under configuration } s\}$.
Coordinate set I is sufficient iff:

$$s_I = s'_I \implies \text{Opt}(s) = \text{Opt}(s')$$

This holds iff configurations agreeing on parameters in I exhibit identical behaviors.

Therefore, "does parameter subset I preserve all behaviors?" is exactly SUFFICIENCY-CHECK for the constructed decision problem.

Remark 6.2. This reduction is *parsimonious*: every instance of configuration simplification corresponds bijectively to an instance of SUFFICIENCY-CHECK. The problems are not merely related—they are identical up to encoding.

6.2 Computational Rationality of Over-Modeling

We now prove that over-specification is the optimal engineering strategy given complexity constraints.

Theorem 6.3 (Rational Over-Modeling)**.** *Consider an engineer specifying a system configuration with n parameters. Let:*

- $C_{over}(k)$ = *cost of maintaining k extra parameters beyond minimal*
- $C_{find}(n)$ = *cost of finding minimal sufficient parameter set*
- C_{under} = *expected cost of production failures from underspecification*

When SUFFICIENCY-CHECK is coNP-complete (Theorem 3.6):

1. *Worst-case finding cost is exponential: $C_{find}(n) = \Omega(2^n)$*
2. *Maintenance cost is linear: $C_{over}(k) = O(k)$*
3. *For sufficiently large n, exponential cost dominates linear cost*

Therefore, when n exceeds a threshold, over-modeling minimizes total expected cost:

$$C_{over}(k) < C_{find}(n) + C_{under}$$

Over-modeling is the economically optimal strategy under computational constraints.

Proof. By Theorem 3.6, SUFFICIENCY-CHECK is coNP-complete. Under standard complexity assumptions ($\mathsf{P} \neq \mathsf{coNP}$), no polynomial-time algorithm exists for checking sufficiency.

Finding the minimal sufficient set requires checking sufficiency of multiple candidate sets. Exhaustive search examines:

$$\sum_{i=0}^{n} \binom{n}{i} = 2^n \text{ candidate subsets}$$

Each check requires $\Omega(1)$ time (at minimum, reading the input). Therefore:

$$C_{\text{find}}(n) = \Omega(2^n)$$

Maintaining k extra parameters incurs:

- Documentation cost: $O(k)$ entries
- Testing cost: $O(k)$ test cases
- Migration cost: $O(k)$ parameters to update

Total maintenance cost is $C_{\text{over}}(k) = O(k)$.

For concrete threshold: when $n = 20$ parameters, exhaustive search requires $2^{20} \approx 10^6$ checks. Including $k = 5$ extra parameters costs $O(5)$ maintenance overhead but avoids 10^6 computational work.

Since 2^n grows faster than any polynomial in k or n, there exists n_0 such that for all $n > n_0$:

$$C_{\text{over}}(k) \ll C_{\text{find}}(n)$$

Adding underspecification risk C_{under} (production failures from missing parameters), which can be arbitrarily large, makes over-specification strictly dominant.

Corollary 6.4 (Impossibility of Automated Configuration Minimization)**.** *There exists no polynomial-time algorithm that:*

1. *Takes an arbitrary configuration file with n parameters*
2. *Identifies the minimal sufficient parameter subset*
3. *Guarantees correctness (no false negatives)*

Proof. Such an algorithm would solve MINIMUM-SUFFICIENT-SET in polynomial time, contradicting Theorem 3.7 (assuming $\mathsf{P} \neq \mathsf{coNP}$).

Remark 6.5. Corollary 6.4 explains the observed absence of "config cleanup" tools in software engineering practice. Engineers who include extra parameters are not exhibiting poor discipline—they are adapting optimally to computational impossibility. The problem is not lack of tooling effort; it is mathematical intractability.

6.3 Connection to Observed Practice

These theorems provide mathematical grounding for three widespread engineering behaviors:

1. Configuration files grow over time. Removing parameters requires solving coNP-complete problems. Engineers rationally choose linear maintenance cost over exponential minimization cost.

2. Heuristic model selection dominates. ML practitioners use AIC, BIC, cross-validation instead of optimal feature selection because optimal selection is intractable (Theorem 6.3).

3. "Include everything" is a legitimate strategy. When determining relevance costs $\Omega(2^n)$, including all n parameters costs $O(n)$. For large n, this is the rational choice.

These are not workarounds or approximations. They are *optimal responses* to computational constraints. The complexity results transform engineering practice from art to mathematics: over-modeling is not a failure—it is the provably correct strategy.

7 Implications for Software Architecture

The complexity results have direct implications for software engineering practice.

7.1 Why Over-Specification Is Rational

Software architects routinely specify more configuration parameters than strictly necessary. Our results show this is computationally rational:

Corollary 7.1 (Rational Over-Specification)**.** *Given a software system with n configuration parameters, checking whether a proposed subset suffices is coNP-complete. Finding the minimum such set is also coNP-complete.*

This explains why configuration files grow over time: removing "unnecessary" parameters requires solving a hard problem.

7.2 Connection to Leverage Theory

Paper 3 introduced leverage as the ratio of impact to effort. The decision quotient provides a complementary measure:

Definition 7.2 (Architectural Decision Quotient)**.** For a software system with configuration space S and behavior space B:

$$\text{ADQ}(I) = \frac{|\{b \in B : b \text{ achievable with some } s \text{ where } s_I \text{ fixed}\}|}{|B|}$$

High ADQ means the configuration subset I leaves many behaviors achievable—it doesn't constrain the system much. Low ADQ means I strongly constrains behavior.

Proposition 7.3 (Leverage-ADQ Duality)**.** *High-leverage architectural decisions correspond to low-ADQ configuration subsets: they strongly constrain system behavior with minimal specification.*

7.3 Practical Recommendations

Based on our theoretical results:

1. **Accept over-modeling:** Don't penalize engineers for including "extra" parameters. The alternative (minimal modeling) is computationally hard.
2. **Use bounded scenarios:** When the scenario space is small (Proposition 2.10), minimal modeling becomes tractable.
3. **Exploit structure:** Tree-structured dependencies, bounded alternatives, and separable utilities admit efficient algorithms.
4. **Invest in heuristics:** For general problems, develop domain-specific heuristics rather than seeking optimal solutions.

8 Related Work

8.1 Computational Decision Theory

The complexity of decision-making has been studied extensively. Papadimitriou [9] established foundational results on the complexity of game-theoretic solution concepts. Our work extends this to the meta-question of identifying relevant information. For a modern treatment of complexity classes, see Arora and Barak [2].

8.2 Feature Selection

In machine learning, feature selection asks which input features are relevant for prediction. This is known to be NP-hard in general [3]. Our results show the decision-theoretic analog is coNP-complete for both checking and minimization.

8.3 Value of Information

The value of information (VOI) framework [5] quantifies how much a decision-maker should pay for information. Our work addresses a different question: not the *value* of information, but the *complexity* of identifying which information has value.

8.4 Model Selection

Statistical model selection (AIC [1], BIC [13], cross-validation [15]) provides practical heuristics for choosing among models. Our results provide theoretical justification: optimal model selection is intractable, so heuristics are necessary.

9 Conclusion

Methodology and Disclosure

Role of LLMs in this work. This paper was developed through human-AI collaboration. The author provided the core intuitions—the connection between decision-relevance and computational complexity, the conjecture that SUFFICIENCY-CHECK is coNP-complete, and the insight that the

Σ_2^P structure collapses for MINIMUM-SUFFICIENT-SET. Large language models (Claude, GPT-4) served as implementation partners for proof drafting, Lean formalization, and LaTeX generation.

The Lean 4 proofs were iteratively refined: the author specified what should be proved, the LLM proposed proof strategies, and the Lean compiler served as the arbiter of correctness. The complexity-theoretic reductions required careful human oversight to ensure the polynomial bounds were correctly established.

What the author contributed: The problem formulations (SUFFICIENCY-CHECK, MINIMUM-SUFFICIENT-SET, ANCHOR-SUFFICIENCY), the hardness conjectures, the tractability conditions, and the connection to over-modeling in engineering practice.

What LLMs contributed: LaTeX drafting, Lean tactic exploration, reduction construction assistance, and prose refinement.

The proofs are machine-checked; their validity is independent of generation method. We disclose this methodology in the interest of academic transparency.

We have established that identifying decision-relevant information is computationally hard:

- Checking whether a coordinate set is sufficient is coNP-complete
- Finding the minimum sufficient set is coNP-complete (the Σ_2^P structure collapses)
- Anchor sufficiency (fixed-coordinate subcube) is Σ_2^P-complete
- A complexity dichotomy separates easy (logarithmic) from hard (linear) cases
- Tractable subcases exist for bounded actions, separable utilities, and tree structures

These results formalize a fundamental insight: **determining what you need to know is harder than knowing everything**. This explains the ubiquity of over-modeling in engineering practice and provides theoretical grounding for heuristic approaches to model selection.

All proofs are machine-checked in Lean 4, ensuring correctness of the core mathematical claims.

A Lean 4 Proof Listings

The complete Lean 4 formalization is available at:

https://github.com/[repository]/openhcs/docs/papers/paper4_decision_quotient/proofs

A.1 Module Structure

The formalization consists of 25 files organized as follows:

- `Basic.lean` – Core definitions (DecisionProblem, CoordinateSet, Projection)
- `AlgorithmComplexity.lean` – Complexity definitions (polynomial time, reductions)
- `PolynomialReduction.lean` – Polynomial reduction composition (Theorem A.1)
- `Reduction.lean` – TAUTOLOGY reduction for sufficiency checking
- `Hardness/` – Counting complexity and approximation barriers

- `Tractability/` – Bounded actions, separable utilities, tree structure, FPT
- `Economics/` – Value of information and elicitation connections
- `Dichotomy.lean` and `ComplexityMain.lean` – Summary results

A.2 Key Theorems

Theorem A.1 (Polynomial Composition, Lean)**.** *Polynomial-time reductions compose to polynomial-time reductions.*

```
theorem PolyReduction.comp_exists
    (f : PolyReduction A B) (g : PolyReduction B C) :
    exists h : PolyReduction A C,
      forall a, h.reduce a = g.reduce (f.reduce a)
```

A.3 Verification Status

- Total lines: 3,500+
- Theorems: ∼65
- Files: 28
- Status: All proofs in this directory compile with no `sorry`. The formalization has been upgraded from CNF to general Boolean Formula ASTs to correctly model the coNP-completeness of TAUTOLOGY.

References

[1] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[2] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[3] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[4] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM, 1971.

[5] Ronald A. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, 2(1):22–26, 1966.

[6] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[7] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

[8] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[9] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[10] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[11] Howard Raiffa and Robert Schlaifer. *Applied Statistical Decision Theory*. Harvard University Press, 1961.

[12] Leonard J. Savage. *The Foundations of Statistics*. John Wiley & Sons, 1954.

[13] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[14] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[15] Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.