Computational Complexity of Sufficiency in Decision Problems

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

February 20, 2026

Abstract

We characterize the computational complexity of coordinate sufficiency in decision problems within the formal model. Given action set A, state space $S = X_1 \times \cdots \times X_n$, and utility $U : A \times S \to \mathbb{Q}$, a coordinate set I is *sufficient* if $s_I = s'_I \implies \mathrm{Opt}(s) = \mathrm{Opt}(s')$.

The landscape in the formal model:

- **General case:** SUFFICIENCY-CHECK is coNP-complete; ANCHOR-SUFFICIENCY is Σ_2^{P}-complete.
- **Tractable cases:** Polynomial-time for bounded action sets under the explicit-state encoding; separable utilities ($u = f + g$) under any encoding; and tree-structured utility with explicit local factors.
- **Encoding-regime separation:** Polynomial-time under the explicit-state encoding (polynomial in $|S|$). Under ETH, there exist succinctly encoded worst-case instances witnessed by a strengthened gadget construction (mechanized in Lean; see Appendix A) with $k^* = n$ for which SUFFICIENCY-CHECK requires $2^{\Omega(n)}$ time.

The tractable cases are stated with explicit encoding assumptions (Section 2.4). Together, these results answer the question "when is decision-relevant information identifiable efficiently?" within the stated regimes. At the structural level, the apparent $\exists\forall$ form of MINIMUM-SUFFICIENT-SET collapses to a coNPcharacterization via the criterion sufficient$(I) \iff$ Relevant $\subseteq I$.

The primary contribution is theoretical: a complete characterization of the core decision-relevant problems in the formal model (coNP/Σ_2^P completeness and tractable cases under explicit encoding assumptions). The practical corollaries are diagnostic: the hardness results quantify the current cost of incomplete structural characterization rather than asserting permanent barriers.

The reduction constructions and key equivalence theorems are machine-checked in Lean 4 ($\sim$5,900 lines, 230+ theorem/lemma statements); complexity classifications follow by composition with standard results (see Appendix A).

Keywords: computational complexity, decision theory, polynomial hierarchy, tractability dichotomy, Lean 4

Zenodo DOI: 10.5281/zenodo.18140965

1 Introduction

Consider a decision problem with actions A and states $S = X_1 \times \cdots \times X_n$. A coordinate set $I \subseteq \{1, \ldots, n\}$ is *sufficient* if knowing only coordinates in I determines the optimal action:

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

This paper characterizes the efficient cases of coordinate sufficiency within the formal model: Section 2.4 fixes the computational model and input encodings used for all complexity claims.

Problem	**Complexity**	**When Tractable**
SUFFICIENCY-CHECK	coNP-complete	Bounded actions (explicit-state), separable utility, tree-structured utility
MINIMUM-SUFFICIENT-SET	coNP-complete	Same conditions
ANCHOR-SUFFICIENCY	Σ_2^{P}-complete	Open

The tractable cases are stated with explicit encoding assumptions (Section 2.4). Outside those regimes, the succinct model yields hardness.

1.1 Landscape Summary

When is sufficiency checking tractable? We identify three sufficient conditions:

1. **Bounded actions** ($|A| \leq k$) under explicit-state encoding: with constantly many actions, we enumerate action pairs over the explicit utility table.

2. **Separable utility** ($u(a, s) = f(a) + g(s)$): The optimal action depends only on f, making all coordinates irrelevant to the decision.

3. **Tree-structured utility**: With explicit local factors over a tree, dynamic programming yields polynomial algorithms in the input length.

Each condition is stated with its encoding assumption. Outside these regimes, the general problem remains coNP-hard (Theorem 3.6).

When is it intractable? The general problem is coNP-complete (Theorem 3.6), with a separation between explicit-state tractability and succinct worst-case hardness:

- In the explicit-state model: SUFFICIENCY-CHECK is solvable in polynomial time in $|S|$ by explicitly computing $\mathrm{Opt}(s)$ for all $s \in S$ and checking all pairs (s, s') with equal I-projection. In particular, instances with $k^* = O(\log |S|)$ are tractable in this model.

- In the succinct model: under ETH there exist worst-case instances produced by our strengthened gadget in which the minimal sufficient set has size $\Omega(n)$ (indeed n) and SUFFICIENCY-CHECK requires $2^{\Omega(n)}$ time.

The lower-bound statement does not address intermediate regimes.

1.2 Main Theorems

1. **Theorem 3.6:** SUFFICIENCY-CHECK is coNP-complete via reduction from TAUTOLOGY.
2. **Theorem 3.7:** MINIMUM-SUFFICIENT-SET is coNP-complete (the Σ_2^{P} structure collapses).
3. **Theorem 3.9:** ANCHOR-SUFFICIENCY is Σ_2^{P}-complete via reduction from ∃∀-SAT.
4. **Theorem 4.1:** Encoding-regime separation: explicit-state polynomial-time (polynomial in $|S|$), and under ETH a succinct worst-case lower bound witnessed by a hard family with $k^* = n$.
5. **Theorem 5.1:** Polynomial algorithms for bounded actions, separable utility, tree structure.

1.3 Machine-Checked Proofs

The reduction constructions and key equivalence theorems are machine-checked in Lean 4 [6] (∼5,900 lines, 230+ theorem/lemma statements). The formalization verifies that the TAUTOLOGY reduction correctly maps tautologies to sufficient coordinate sets. Complexity class membership (coNP-completeness, Σ_2^P-completeness) follows by composition with standard complexity-theoretic results.

What is new. We contribute (i) formal definitions of decision-theoretic sufficiency in Lean; (ii) machine-checked proofs of reduction correctness for the TAUTOLOGY and ∃∀-SAT reductions; and (iii) a complete complexity landscape for coordinate sufficiency with explicit encoding assumptions. Prior work establishes hardness informally; we formalize the constructions.

1.4 Paper Structure

The primary contribution is theoretical: a formalized reduction framework and a complete characterization of the core decision-relevant problems in the formal model (coNP/Σ_2^P completeness and tractable cases stated under explicit encoding assumptions). Sections 3–5 contain the core theorems.

Section 2: foundations. Section 3: hardness proofs. Section 4: dichotomy. Section 5: tractable cases. Section 6: implications for practice. Section 7: software architecture corollaries. Section 8: complexity conservation. Section 9: related work. Appendix A: Lean 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 $s = (s_1, \dots, s_n) \in S$ and coordinate set $I \subseteq \{1, \dots, n\}$:

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

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 $I \subseteq \{1, \dots, n\}$ 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: $X_1 = \{\text{rain}, \text{no rain}\}$, $X_2 = \{\text{hot}, \text{cold}\}$, $X_3 = \{\text{Monday}, \dots, \text{Sunday}\}$
- 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:

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

This measures the fraction of actions that are optimal for at least one state consistent with 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 $s \sim_I s' \implies \mathrm{Opt}(s) = \mathrm{Opt}(s')$, so the set of actions optimal for some $s' \sim_I s$ is exactly $\mathrm{Opt}(s)$.

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

2.4 Computational Model and Input Encoding

We fix the computational model used by the complexity claims.

Succinct encoding (primary for hardness). This succinct circuit encoding is the standard representation for decision problems in complexity theory; hardness is stated with respect to the input length of the circuit description [3]. An instance is encoded as:

- a finite action set A given explicitly,
- coordinate domains $X_1, \ldots, X_n$ given by their sizes in binary,
- a Boolean or arithmetic circuit C_U that on input (a, s) outputs $U(a, s)$.

The input length is $L = |A| + \sum_i \log |X_i| + |C_U|$. Polynomial time and all complexity classes (coNP, Σ_2^P, ETH) are measured in L. All hardness results in Section 3 use this encoding.

Explicit-state encoding (used for enumeration algorithms and experiments). The utility is given as a full table over $A \times S$. The input length is $L_{\text{exp}} = \Theta(|A||S|)$ (up to the bitlength of utilities). Polynomial time is measured in L_{exp}. Results stated in terms of $|S|$ use this encoding.

Unless explicitly stated otherwise, "polynomial time" refers to the succinct encoding.

2.5 Structural Complexity vs Representational Hardness

Definition 2.11 (Structural Complexity)**.** For a fixed formal decision relation (e.g., "I is sufficient for $\mathcal{D}$"), *structural complexity* means its placement in standard complexity classes within the formal model (coNP, Σ_2^P, etc.), as established by class-membership arguments and reductions.

Definition 2.12 (Representational Hardness)**.** For a fixed decision relation and an encoding regime E (Section 2.4), *representational hardness* is the worst-case computational cost incurred by solvers whose input access is restricted to E.

Remark 2.13 (Interpretation Contract). This paper keeps the decision relation fixed and varies the encoding regime explicitly. Thus, later separations are read as changes in representational hardness under fixed structural complexity, not as changes to the underlying sufficiency semantics.

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

Within the formal model (Section 2) and the succinct encoding used for hardness (Section 2.4), these results sit beyond NP-completeness and rigorously characterize over-modeling as a diagnostic response to unresolved boundary characterization: in the worst case, finding (or even certifying) a 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 $(A, X_1, \ldots, X_n, U)$ where:

- A is a finite set of alternatives
- $X_1, \ldots, X_n$ are the coordinate domains, with state space $S = X_1 \times \cdots \times X_n$
- $U : A \times S \to \mathbb{Q}$ is the utility function (in the succinct encoding, U is given as a Boolean circuit)

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

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

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

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

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

Problem 3.4 (SUFFICIENCY-CHECK)**. Input:** Decision problem $(A, X_1, \ldots, X_n, U)$ and coordinate set $I \subseteq \{1, \ldots, n\}$
Question: Is I sufficient?

Problem 3.5 (MINIMUM-SUFFICIENT-SET)**. Input:** Decision problem $(A, X_1, \ldots, X_n, U)$ 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 [5, 9].* (Machine-verified in Lean 4; see Appendix A.)

Source	Target	Key property preserved
TAUTOLOGY	SUFFICIENCY-CHECK	Tautology iff $\emptyset$ sufficient
∃∀-SAT	ANCHOR-SUFFICIENCY	Witness anchors iff formula true

Proof. **Membership in coNP:** The complementary problem INSUFFICIENCY is in NP. Given a decision problem $(A, X_1, \ldots, X_n, U)$ and coordinate set I, a witness for insufficiency is a pair (s, s') such that:

1. $s_I = s'_I$ (verifiable in polynomial time)
2. $\operatorname{Opt}(s) \neq \operatorname{Opt}(s')$ (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: $S = \{\text{reference}\} \cup \{0,1\}^n$ (equivalently, encode this as a product space with one extra coordinate $r \in \{0,1\}$ indicating whether the state is the reference state)

- 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 $\iff \varphi$ is a tautology.

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

- $\text{Opt}(a) = \text{accept}$ when $\varphi(a) = 1$

- $\text{Opt}(a) = \{\text{accept}, \text{reject}\}$ 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 $\text{Opt}(s) = \{\text{accept}\}$ for all states s, making $I = \emptyset$ sufficient. ■

Mechanized strengthening (all coordinates relevant). The reduction above establishes coNP-hardness using a single witness set $I = \emptyset$. For the ETH-based lower bound in Theorem 4.1, we additionally need worst-case instances where the minimal sufficient set has *linear* size.

We formalized a strengthened reduction in Lean 4: given a Boolean formula φ over n variables, construct a decision problem with n coordinates such that if φ is not a tautology then *every* coordinate is decision-relevant (so $k^* = n$). Intuitively, the construction places a copy of the base gadget at each coordinate and makes the global "accept" condition hold only when every coordinate's local test succeeds; a single falsifying assignment at one coordinate therefore changes the global optimal set, witnessing that coordinate's relevance. This strengthening is mechanized in Lean; see Appendix A.

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:

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

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):

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

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

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 $(A, X_1, \ldots, X_n, U)$ 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 $U(\mathrm{YES}) = 2$ and $U(\mathrm{NO}) \leq 1$, so $\mathrm{Opt}(x^\star, y) = \{\mathrm{YES}\}$ is constant. Conversely, fix x and suppose $\exists y_f$ with $\varphi(x, y_f) = 0$.

- If $\varphi(x, 0^m) = 1$, then $\mathrm{Opt}(x, 0^m) = \{\mathrm{YES}\}$. The falsifying assignment must satisfy $y_f \neq 0^m$, where $U(\mathrm{YES}) = U(\mathrm{NO}) = 0$, so $\mathrm{Opt}(x, y_f) = \{\mathrm{YES}, \mathrm{NO}\}$.
- If $\varphi(x, 0^m) = 0$, then $\mathrm{Opt}(x, 0^m) = \{\mathrm{NO}\}$. After padding we have $m > 0$, so choose any $y' \neq 0^m$: either $\varphi(x, y') = 1$ (then $\mathrm{Opt}(x, y') = \{\mathrm{YES}\}$) or $\varphi(x, y') = 0$ (then $\mathrm{Opt}(x, y') = \{\mathrm{YES}, \mathrm{NO}\}$). In both subcases the optimal set differs from $\{\mathrm{NO}\}$.

Hence the subcube for this x 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 $\mathrm{Opt}(s)$ 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*

The modeling budget for deciding what to model is itself a computationally constrained resource.

3.7 Quantifier Collapse for MINIMUM-SUFFICIENT-SET

Theorem 3.13 (Collapse of the Apparent $\exists\forall$ Structure). *The apparent second-level predicate*

$$\exists I\ (|I| \leq k)\ \forall s, s' \in S :\ s_I = s'_I \implies \mathit{Opt}(s) = \mathit{Opt}(s')$$

is equivalent to the coNP predicate $|\mathit{Relevant}| \leq k$, where

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

Consequently, MINIMUM-SUFFICIENT-SET is governed by coNP certificates rather than a genuine Σ_2^P alternation.

Proof. By the formal lemma `sufficient_contains_relevant`, a coordinate set I is sufficient iff $\mathrm{Relevant} \subseteq I$. Therefore:

$$\exists I\ (|I| \leq k \wedge \mathrm{sufficient}(I)) \iff \exists I\ (|I| \leq k \wedge \mathrm{Relevant} \subseteq I) \iff |\mathrm{Relevant}| \leq k.$$

So the existential-over-universal presentation collapses to counting the relevant coordinates.

A NO certificate for $|\mathrm{Relevant}| \leq k$ is a list of $k+1$ distinct relevant coordinates, each witnessed by two states that differ only on that coordinate and yield different optimal sets; this is polynomially verifiable. Hence the resulting predicate is in coNP, matching Theorem 3.7.

This also clarifies why ANCHOR-SUFFICIENCY remains Σ_2^P-complete: once an anchor assignment is existentially chosen, the universal quantifier over the residual subcube does not collapse to a coordinate-counting predicate. ■

4 Encoding-Regime Separation

The hardness results of Section 3 apply to worst-case instances under the succinct encoding. This section states an encoding-regime separation: an explicit-state upper bound versus a succinct-encoding worst-case lower bound.

Model note. Part 1 is an explicit-state upper bound (time polynomial in $|S|$). Part 2 is a succinct-encoding lower bound under ETH (time exponential in n). The encodings are defined in Section 2.4. These two parts are stated in different encodings and are not directly comparable as functions of a single input length.

Theorem 4.1 (Explicit–Succinct Regime Separation)**.** *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. ***Explicit-state upper bound:*** *Under the explicit-state encoding, SUFFICIENCY-CHECK is solvable in time polynomial in N (e.g. $O(N^2|A|)$).*
2. ***Succinct lower bound under ETH (worst case):*** *Assuming ETH, there exists a family of succinctly encoded instances with n coordinates and minimal sufficient set size $k^* = n$ such that SUFFICIENCY-CHECK requires time $2^{\Omega(n)}$.*

Proof. **Part 1 (Explicit-state upper bound):** Under the explicit-state encoding, SUFFICIENCY-CHECK is decidable in time polynomial in N by direct enumeration: compute $\mathrm{Opt}(s)$ for all $s \in S$ and then check all pairs (s, s') with $s_I = s'_I$.

Equivalently, for any fixed I, the projection map $s \mapsto s_I$ has image size $|S_I| \leq |S| = N$, so any algorithm that iterates over projection classes (or over all state pairs) runs in polynomial time in N. Thus, in particular, when $k^* = O(\log N)$, SUFFICIENCY-CHECK is solvable in polynomial time under the explicit-state encoding.

Remark (bounded coordinate domains). In the general model $S = \prod_i X_i$, for a fixed I one always has $|S_I| \leq \prod_{i \in I} |X_i|$ (and $|S_I| \leq N$). If the coordinate domains are uniformly bounded, $|X_i| \leq d$ for all i, then $|S_I| \leq d^{|I|}$.

Part 2 (Succinct ETH lower bound, worst case): A strengthened version of the TAUTOLOGY reduction used in Theorem 3.6 produces a family of instances in which the minimal sufficient set has size $k^* = n$: given a Boolean formula φ over n variables, we construct a decision problem with n coordinates such that if φ is not a tautology then *every* coordinate is decision-relevant (thus $k^* = n$). This strengthening is mechanized in Lean (see Appendix A). Under the Exponential Time Hypothesis (ETH) [8], TAUTOLOGY requires time $2^{\Omega(n)}$ in the succinct encoding, so SUFFICIENCY-CHECK inherits a $2^{\Omega(n)}$ worst-case lower bound via this reduction. ■

Corollary 4.2 (Regime Separation (by Encoding))**.** *There is a clean separation between explicit-state tractability and succinct worst-case hardness (with respect to the encodings in Section 2.4):*

- *Under the explicit-state encoding, SUFFICIENCY-CHECK is polynomial in $N = |S|$.*
- *Under ETH, there exist succinctly encoded instances with $k^* = \Omega(n)$ (indeed $k^* = n$) for which SUFFICIENCY-CHECK requires $2^{\Omega(n)}$ time.*

For Boolean coordinate spaces ($N = 2^n$), the explicit-state bound is polynomial in 2^n (exponential in n), while under ETH the succinct lower bound yields $2^{\Omega(n)}$ time for the hard family in which $k^* = \Omega(n)$.

Remark 4.3 (Instantiation of Definitions 2.11 and 2.12). Theorem 4.1 keeps the structural problem fixed (same sufficiency relation) and separates representational hardness by encoding regime: explicit-state access exposes the boundary $s \mapsto \mathrm{Opt}(s)$, while succinct access can hide it enough to force ETH-level worst-case cost on a hard family.

This encoding-regime separation explains why some domains admit tractable model selection under explicit-state assumptions, while other domains (or encodings) exhibit worst-case hardness that forces heuristic approaches.

5 Tractable Special Cases: When You Can Solve It

We distinguish the encodings of Section 2.4. The tractability results below state the model assumption explicitly. Structural insight: hardness dissolves exactly when the full decision boundary $s \mapsto \mathrm{Opt}(s)$ is recoverable in polynomial time from the input representation; the three cases below instantiate this single principle. Concretely, each tractable regime corresponds to a specific structural insight (explicit boundary exposure, additive separability, or tree factorization) that removes the hardness witness; this supports reading the general-case hardness as missing structural access in the current representation rather than as an intrinsic semantic barrier.

Theorem 5.1 (Tractable Subcases)**.** *SUFFICIENCY-CHECK is polynomial-time solvable in the following cases:*

1. ***Explicit-state encoding:*** *The input contains the full utility table over $A \times S$. SUFFICIENCY-CHECK runs in $O(|S|^2|A|)$ time; if $|A|$ is constant, $O(|S|^2)$.*
2. ***Separable utility (any encoding):*** *$U(a, s) = f(a) + g(s)$.*
3. ***Tree-structured utility with explicit local factors (succinct structured encoding):*** *There exists a rooted tree on coordinates and local functions u_i such that*

$$U(a, s) = \sum_i u_i\big(a, s_i, s_{\mathrm{parent}(i)}\big),$$

with the root term depending only on (a, s_{root}) and all u_i given explicitly as part of the input.

5.1 Explicit-State Encoding

Proof of Part 1. Given the full table of $U(a, s)$, compute $\mathrm{Opt}(s)$ for all $s \in S$ in $O(|S||A|)$ time. For SUFFICIENCY-CHECK on a given I, verify that for all pairs (s, s') with $s_I = s'_I$, we have $\mathrm{Opt}(s) = \mathrm{Opt}(s')$. This takes $O(|S|^2|A|)$ time by direct enumeration and is polynomial in the explicit input length. If $|A|$ is constant, the runtime is $O(|S|^2)$. ■

5.2 Separable Utility

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

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

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

5.3 Tree-Structured Utility

Proof of Part 3. Assume the tree decomposition and explicit local tables as stated. For each node i and each value of its parent coordinate, compute the set of actions that are optimal for some assignment of the subtree rooted at i. This is a bottom-up dynamic program that combines local tables with child summaries; each table lookup is explicit in the input. A coordinate is relevant if and only if varying its value changes the resulting optimal action set. The total runtime is polynomial in n, $|A|$, and the size of the local tables. ∎

5.4 Practical Implications

Condition	**Examples**
Explicit-state encoding	Small or fully enumerated state spaces
Separable utility	Additive costs, linear models
Tree-structured utility	Hierarchies, causal trees

6 Implications for Practice: Diagnostic Reading of Over-Modeling

This section states corollaries for engineering practice. Within the formal model, the complexity results of Sections 3 and 4 shift parts of this workflow from informal judgment toward explicit, checkable criteria. The observed behaviors—configuration over-specification, absence of automated minimization tools, heuristic model selection—are *diagnostic signals* that the decision boundary is not yet fully characterized in a tractable representation.

Some common prescriptions implicitly require exact minimization of sufficient parameter sets. In the worst case, that task is coNP-complete in our model, so critiques must distinguish between true irrelevance and unresolved relevance structure under the chosen encoding. The interpretive key for this section is Theorem 6.2: persistent failure to isolate a minimal sufficient set is a boundary-characterization signal in the current model, not a universal irreducibility claim.

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 $\mathrm{Opt}(s) = \{b \in B : b \text{ occurs under configuration } s\}$.

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. ■

Theorem 6.2 (Over-Modeling as Diagnostic Signal)**.** *By contraposition of Theorem 6.1, if the minimal sufficient parameter set is not immediately identifiable in the modeled system, then the decision boundary is not completely characterized by the current parameterization.*

Proof. Assume the decision boundary were completely characterized by the current parameterization. Then, via Theorem 6.1, the corresponding sufficiency instance would admit an exact statement of which parameter subsets preserve behavior, including the minimal sufficient set. Contraposition gives the claim: persistent failure to identify a minimal sufficient set signals incomplete characterization of decision relevance in the model. ■

6.2 Cost Asymmetry Under ETH

We now prove a cost asymmetry result under the stated cost model and complexity constraints.[1]

Theorem 6.3 (Cost-Asymmetry Consequence Under ETH)**.** *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*

Assume ETH in the succinct encoding model of Section 2.4. Then:

1. *Exact identification of minimal sufficient sets has worst-case finding cost $C_{find}(n) = 2^{\Omega(n)}$. (Under ETH, SUFFICIENCY-CHECK has a $2^{\Omega(n)}$ lower bound in the succinct model, and exact minimization subsumes this decision task.)*
2. *Maintenance cost is linear: $C_{over}(k) = O(k)$.*
3. *Under ETH, exponential finding cost dominates linear maintenance cost for sufficiently large n.*

Therefore, there exists n_0 such that for all $n > n_0$, the finding-vs-maintenance asymmetry satisfies:

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

Within this model, persistent over-specification is evidence of unresolved boundary characterization rather than evidence that all included parameters are intrinsically necessary.

[1] Naive subset enumeration still gives an intuitive baseline of $O(2^n)$ checks, but that is an algorithmic upper bound; the theorem below uses ETH for the lower-bound argument.

Proof. Under ETH, the TAUTOLOGY reduction used in Theorem 3.6 yields a $2^{\Omega(n)}$ worst-case lower bound for SUFFICIENCY-CHECK in the succinct encoding. Any exact algorithm that outputs a minimum sufficient set can decide whether the optimum size is 0 by checking whether the returned set is empty; this is exactly the SUFFICIENCY-CHECK query for $I = \emptyset$. Hence exact minimal-set finding inherits the same exponential worst-case lower bound.

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 $C_{\text{over}}(k) = O(k)$.

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} reinforces conservative configurations until additional structure is identified. ∎

Corollary 6.4 (Impossibility of Automated Configuration Minimization)**.** *Assuming $\mathsf{P} \neq \mathsf{coNP}$, 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. The correct interpretation is diagnostic: persistent inability to minimize indicates incomplete tractable characterization of decision relevance under the current representation.

6.3 Diagnostic Reading of Observed Practice

These theorems provide mathematical grounding for three widespread engineering behaviors:

1. Configuration files grow over time. Difficulty removing parameters indicates unresolved relevance structure in the current model/encoding.

2. Heuristic model selection dominates. ML practitioners use AIC, BIC, cross-validation instead of optimal feature selection because exact selection is intractable without additional structure (Theorem 6.3).

3. "Include everything" is a conservative upper-bound policy. When determining relevance/minimal sufficiency has a $2^{\Omega(n)}$ worst-case lower bound under ETH in the succinct encoding (Theorem 4.1), including all n parameters costs $O(n)$ while the exact boundary remains unresolved.

These behaviors are not ad hoc workarounds; under the stated computational model they are practical consequences of a boundary-identification bottleneck. The complexity results provide a diagnostic lens: persistent over-modeling marks the need for either stronger structural assumptions or explicit acceptance of approximation.

7 Applied Corollaries for Software Architecture

Proposition 7.1 (Model-Scope Firewall)**.** *The statements in this section are formal consequences of the model assumptions fixed in Sections 2–3; they are not empirical claims about systems outside those assumptions.*

Proof. Each result below is derived by explicit reference to prior theorems in this paper (Theorems 6.2, 5.1, 3.6, and 4.1) and therefore has exactly the scope of those premises. ∎

7.1 Over-Specification as Diagnostic Signal

Corollary 7.2 (Persistent Over-Specification)**.** *In the mechanized Boolean-coordinate model, if a coordinate is relevant and omitted from a candidate set I, then I is not sufficient. (Lean:* `DecisionQuotient.Sigma2PHardness.sufficient_iff_relevant_subset`*)*

Proof. This is the contrapositive of `DecisionQuotient.Sigma2PHardness.sufficient_iff_relevant_subset`. ∎

7.2 Architectural Decision Quotient

Definition 7.3 (Architectural Decision Quotient)**.** For a software system with configuration space S and behavior space B:

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

Proposition 7.4 (ADQ Ordering)**.** *For coordinate sets I, J in the same system, if $\mathrm{ADQ}(I) < \mathrm{ADQ}(J)$, then fixing I leaves a strictly smaller achievable-behavior set than fixing J.*

Proof. The denominator $|B|$ is shared. Thus $\mathrm{ADQ}(I) < \mathrm{ADQ}(J)$ is equivalent to a strict inequality between the corresponding achievable-behavior set cardinalities. ∎

7.3 Corollaries for Practice

Corollary 7.5 (Cardinality Criterion for Exact Minimization)**.** *In the mechanized Boolean-coordinate model, existence of a sufficient set of size at most k is equivalent to the relevance set having cardinality at most k. (Lean:* `DecisionQuotient.Sigma2PHardness.min_sufficient_set_iff_relevant_card`*)*

Proof. Directly from `DecisionQuotient.Sigma2PHardness.min_sufficient_set_iff_relevant_card`. ∎

Corollary 7.6 (Bounded-Regime Tractability)**.** *When the bounded-action or explicit-state conditions of Theorem 5.1 hold, minimal modeling can be solved in polynomial time in the stated input size. (Lean:* `DecisionQuotient.sufficiency_poly_bounded_actions`*)*

Proof. This is the mechanized theorem `DecisionQuotient.sufficiency_poly_bounded_actions`. ∎

Corollary 7.7 (Separable-Utility Tractability)**.** *When utility is separable with explicit factors, sufficiency checking is polynomial in the explicit-state regime. (Lean:* `DecisionQuotient.sufficiency_poly_separable`*)*

Proof. This is the mechanized theorem `DecisionQuotient.sufficiency_poly_separable`. ■

Corollary 7.8 (Tree-Structured Tractability)**.** *When utility factors form a tree structure with explicit local factors, sufficiency checking is polynomial in the explicit-state regime. (Lean:* `DecisionQuotient.sufficiency_poly_tree_structured`*)*

Proof. This is the mechanized theorem `DecisionQuotient.sufficiency_poly_tree_structured`. ■

Corollary 7.9 (Mechanized Hard Family)**.** *There is a machine-checked family of reduction instances where, for non-tautological source formulas, every coordinate is relevant ($k^* = n$), exhibiting worst-case boundary complexity. (Lean:* `DecisionQuotient.all_coords_relevant_of_not_tautology`*)*

Proof. Directly from `DecisionQuotient.all_coords_relevant_of_not_tautology`. ■

7.4 Hardness Distribution: Right Place vs Wrong Place

Definition 7.10 (Hardness Distribution)**.** Let P be a problem family under the succinct encoding of Section 2.4. In this section, baseline hardness $H(P;n)$ denotes worst-case computational step complexity on instances with n coordinates (equivalently, as a function of succinct input length L) in the fixed encoding regime. A *solution architecture* S partitions this baseline hardness into:

- $H_{\text{central}}(S)$: hardness paid once, at design time or in a shared component
- $H_{\text{distributed}}(S)$: hardness paid per use site

For n use sites, total realized hardness is:

$$H_{\text{total}}(S) = H_{\text{central}}(S) + n \cdot H_{\text{distributed}}(S)$$

Proposition 7.11 (Hardness Conservation Principle)**.** *For any problem family P measured by $H(P;n)$ above, any solution architecture S and any number of use sites $n \geq 1$, if $H_{total}(S)$ is measured in the same worst-case step units over the same input family, then:*

$$H_{total}(S) = H_{central}(S) + n \cdot H_{distributed}(S) \geq H(P;n).$$

For SUFFICIENCY-CHECK, Theorem 4.1 provides the baseline on the hard succinct family: $H(\textit{SUFFICIENCY-CHECK};n) = 2^{\Omega(n)}$ under ETH.

Proof. By definition, $H(P;n)$ is a worst-case lower bound for correct solutions in this encoding regime and cost metric. Any such solution architecture decomposes total realized work as $H_{\text{central}} + n \cdot H_{\text{distributed}}$, so that total cannot fall below the baseline. ■

Definition 7.12 (Hardness Efficiency)**.** The *hardness efficiency* of solution S with n use sites is:

$$\eta(S,n) = \frac{H_{\text{central}}(S)}{H_{\text{central}}(S) + n \cdot H_{\text{distributed}}(S)}$$

Proposition 7.13 (Efficiency Interpretation)**.** *For fixed n and positive total hardness, larger $\eta(S,n)$ is equivalent to a larger central share of realized hardness.*

Proof. From Definition 7.12, $\eta(S,n)$ is exactly the fraction of total realized hardness paid centrally. ■

Theorem 7.14 (Centralization Dominance)**.** *For $n > 1$ use sites, solutions with higher $H_{central}$ and lower $H_{distributed}$ yield:*

1. *Lower total realized hardness: $H_{total}(S_1) < H_{total}(S_2)$ when $H_{distributed}(S_1) < H_{distributed}(S_2)$*
2. *Fewer error sites: errors in centralized components affect 1 location; errors in distributed components affect n locations*
3. *Higher leverage: one unit of central effort affects n sites*

Proof. (1) follows from the total hardness formula. (2) follows from error site counting. (3) follows from the definition of leverage as $L = \Delta\text{Effect}/\Delta\text{Effort}$. ■

Corollary 7.15 (Right-Place vs Wrong-Place Hardness)**.** *For architectures $S_{\text{right}}, S_{\text{wrong}}$ over the same problem family, if S_{right} has right hardness, S_{wrong} has wrong hardness, and $n > H_{\text{central}}(S_{\text{right}})$, then*

$$H_{\text{central}}(S_{\text{right}}) + n\, H_{\text{distributed}}(S_{\text{right}}) < H_{\text{central}}(S_{\text{wrong}}) + n\, H_{\text{distributed}}(S_{\text{wrong}}).$$

(Lean: `DecisionQuotient.HardnessDistribution.right_dominates_wrong`)

Proof. This is the mechanized theorem `DecisionQuotient.HardnessDistribution.right_dominates_wrong`. ■

Example (Type System Instantiation). Consider a capability C (e.g., provenance tracking) with one-time central cost H_{central} and per-site manual cost $H_{\text{distributed}}$:

Approach	H_{central}	$H_{\text{distributed}}$
Native type system support	High (learning cost)	Low (type checker enforces)
Manual implementation	Low (no new concepts)	High (reimplement per site)

Corollary 7.16 (Type-System Threshold)**.** *For the formal native-vs-manual architecture instance, native support has lower total realized cost for all*

$$n > \text{intrinsicDOF}(P).$$

(Lean: `DecisionQuotient.HardnessDistribution.native_dominates_manual`)

Proof. Immediate from `DecisionQuotient.HardnessDistribution.native_dominates_manual`. ■

7.5 Extension: Non-Additive Site-Cost Models

Definition 7.17 (Generalized Site Accumulation)**.** Let $C_S : \mathbb{N} \to \mathbb{N}$ be a per-site accumulation function for architecture S. Define generalized total realized hardness by

$$H^{\text{gen}}_{\text{total}}(S, n) = H_{\text{central}}(S) + C_S(n).$$

Definition 7.18 (Eventual Saturation)**.** A cost function $f : \mathbb{N} \to \mathbb{N}$ is *eventually saturating* if there exists N such that for all $n \geq N$, $f(n) = f(N)$.

Theorem 7.19 (Linear Model: Saturation iff Zero Distributed Hardness)**.** *In the linear model of this section,*

$$H_{total}(S,n) = H_{central}(S) + n \cdot H_{distributed}(S),$$

the function $n \mapsto H_{total}(S,n)$ *is eventually saturating if and only if* $H_{distributed}(S) = 0$*. (Lean:* `DecisionQuotient.HardnessDistribution.totalDOF_eventually_constant_iff_zero_distributed`*)*

Proof. This is exactly the mechanized equivalence theorem above. ■

Theorem 7.20 (Generalized Model: Saturation is Possible)**.** *There exists a generalized site-cost model with eventual saturation. In particular, for*

$$C_K(n) = \begin{cases} n, & n \le K \\ K, & n > K, \end{cases}$$

both C_K *and* $n \mapsto H_{central} + C_K(n)$ *are eventually saturating. (Lean:* `DecisionQuotient.HardnessDistribution.saturatingSiteCost_eventually_constant`*,* `DecisionQuotient.HardnessDistribution.generalizedTotal_with_saturation_eventually_constant`*)*

Proof. This is the explicit construction mechanized in Lean. ■

Corollary 7.21 (Positive Linear Slope Cannot Represent Saturation)**.** *No positive-slope linear per-site model can represent the saturating family above for all* n*. (Lean:* `DecisionQuotient.HardnessDistribution.no_positive_slope_linear_represents_saturating`*)*

Proof. This follows from the mechanized theorem that any linear representation of the saturating family must have zero slope. ■

Mechanized strengthening reference. The strengthened all-coordinates-relevant reduction is presented in Section 3 ("Mechanized strengthening") and formalized in `Reduction_AllCoords.lean` via `all_coords_relevant_of_not_tautology`.

The next section develops the major practical consequence of this framework: the Simplicity Tax Theorem.

8 Corollary: Complexity Conservation

Definition 8.1. Let $R(P)$ be the required dimensions (those affecting Opt) and $A(M)$ the dimensions model M handles natively. The *expressive gap* is $\text{Gap}(M,P) = R(P) \setminus A(M)$.

Definition 8.2 (Simplicity Tax)**.** The *simplicity tax* is the size of the expressive gap:

$$\text{SimplicityTax}(M,P) := |\text{Gap}(M,P)|.$$

Theorem 8.3 (Conservation)**.** $|Gap(M,P)| + |R(P) \cap A(M)| = |R(P)|$*. The total cannot be reduced—only redistributed between "handled natively" and "handled externally." (Lean analogue:* `DecisionQuotient.HardnessDistribution.simplicityTax_conservation`*)*

Theorem 8.4 (Linear Growth)**.** *For* n *decision sites:*

$$\mathit{TotalExternalWork} = n \times \mathit{SimplicityTax}(M,P).$$

(Lean analogue: `DecisionQuotient.HardnessDistribution.simplicityTax_grows`*)*

Theorem 8.5 (Amortization)**.** *Let $H_{central}$ be the one-time cost of using a complete model. There exists*

$$n^* = \frac{H_{central}}{SimplicityTax(M, P)}$$

such that for $n > n^*$, the complete model has lower total cost. (Lean analogue: `DecisionQuotient.HardnessDistribution.amortization_threshold_native_manual`)

Corollary 8.6 (Gap Externalization)**.** *If $Gap(M, P) \neq \emptyset$, then external handling cost scales linearly with the number of decision sites. (Lean: `DecisionQuotient.HardnessDistribution.simplicityTax_grows`)*

Proof. Under the Section 7.4 identification of per-site external work with distributed DOF, this is exactly `DecisionQuotient.HardnessDistribution.simplicityTax_grows`. ■

Corollary 8.7 (Exact Minimization Criterion)**.** *For mechanized Boolean-coordinate instances, "there exists a sufficient set of size at most k" is equivalent to "the relevant-coordinate set has cardinality at most k." (Lean: `DecisionQuotient.Sigma2PHardness.min_sufficient_set_iff_relevant_card`)*

Proof. This is `DecisionQuotient.Sigma2PHardness.min_sufficient_set_iff_relevant_card`. ■

Appendix A provides theorem statements and module paths for the corresponding Lean formalization.

9 Related Work

9.1 Computational Decision Theory

The complexity of decision-making has been studied extensively. Papadimitriou [10] 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 [3].

Closest prior work and novelty. Closest to our contribution is the feature-selection/model-selection hardness literature, which proves NP-hardness and inapproximability for predictive subset selection [4, 2]. Our contribution is stronger on two axes those works do not provide: (i) machine-checked reductions (TAUTOLOGY and ∃∀-SAT mappings with explicit polynomial bounds), and (ii) a complete hardness/tractability landscape for decision relevance under explicit encoding assumptions. We study decision relevance rather than predictive compression, and we formalize the core reductions in Lean 4 rather than leaving them only on paper.

9.2 Feature Selection

In machine learning, feature selection asks which input features are relevant for prediction. This is known to be NP-hard in general [4]. Our results show the decision-theoretic analog is coNP-complete for both checking and minimization.

9.3 Value of Information

The value of information (VOI) framework [7] quantifies the maximum rational payment for information. Our work addresses a different question: not the *value* of information, but the *complexity* of identifying which information has value.

9.4 Model Selection

Statistical model selection (AIC [1], BIC [13], cross-validation [15]) provides practical heuristics for choosing among models. Our results clarify why these heuristics dominate current practice: without added structural assumptions, exact optimal model selection inherits worst-case intractability, so heuristic methods are approximation policies for unresolved structure until stronger structure is identified.

10 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 the target statements, 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.

Summary of Results

This paper establishes the computational complexity of coordinate sufficiency problems:

- **SUFFICIENCY-CHECK** is coNP-complete (Theorem 3.6)
- **MINIMUM-SUFFICIENT-SET** is coNP-complete (Theorem 3.7)
- **ANCHOR-SUFFICIENCY** is Σ_2^P-complete (Theorem 3.9)
- An encoding-regime separation contrasts explicit-state polynomial-time (polynomial in $|S|$) with a succinct worst-case ETH lower bound witnessed by a hard family with $k^* = n$ (Theorem 4.1)

- Tractable subcases exist for explicit-state encoding, separable utility, and tree-structured utility with explicit local factors (Theorem 5.1)

These results place the problem of identifying decision-relevant coordinates at the first and second levels of the polynomial hierarchy.

The reduction constructions and key equivalence theorems are machine-checked in Lean 4 (see Appendix A for proof listings). The formalization verifies that the TAUTOLOGY reduction correctly maps tautologies to sufficient coordinate sets; complexity classifications (coNP-completeness, Σ_2^{P}-completeness) follow by composition with standard complexity-theoretic results (TAUTOLOGY is coNP-complete, $\exists\forall$-SAT is Σ_2^{P}-complete). The strengthened gadget showing that non-tautologies yield instances with *all coordinates relevant* is also formalized.

Complexity Characterization

The results provide precise complexity characterizations within the formal model:

1. **Exact bounds.** SUFFICIENCY-CHECK is coNP-complete—both coNP-hard and in coNP.

2. **Constructive reductions.** The reductions from TAUTOLOGY and $\exists\forall$-SAT are explicit and machine-checked.

3. **Encoding-regime separation.** Under the explicit-state encoding, SUFFICIENCY-CHECK is polynomial in $|S|$. Under ETH, there exist succinctly encoded worst-case instances (witnessed by a strengthened gadget family with $k^* = n$) requiring $2^{\Omega(n)}$ time. Intermediate regimes are not ruled out by the lower-bound statement.

The Complexity Conservation Law

Section 8 develops a quantitative consequence: when a problem requires k dimensions and a model handles only $j < k$ natively, the remaining $k - j$ dimensions must be handled externally at each decision site. For n sites, total external work is $(k - j) \times n$.

This conservation law is formalized in Lean 4 (`HardnessDistribution.lean`), proving:

- Conservation: complexity cannot be eliminated, only redistributed

- Dominance: complete models have lower total work than incomplete models

- Amortization: there exists a threshold n^* beyond which higher-dimensional models have lower total cost

Open Questions

Several questions remain for future work:

- **Fixed-parameter tractability (primary):** Is SUFFICIENCY-CHECK FPT when parameterized by the minimal sufficient-set size k^*, or is it W[2]-hard under this parameterization?

- **Average-case complexity:** What is the complexity under natural distributions on decision problems?

- **Learning cost formalization:** Can central cost H_{central} be formalized as the rank of a concept matroid, making the amortization threshold precisely computable?

Practical interpretation

While this work is theoretical, the complexity bounds have practical meaning. Under ETH and the succinct encoding, generic polynomial-time worst-case solvers for arbitrary inputs are ruled out. This is a diagnostic statement, not a claim of permanent impossibility in all regimes: the barrier marks unresolved structural characterization in the current representation, and the response is to refine representation and assumptions until decision relevance becomes structurally tractable or to state approximation commitments explicitly. Theorem 6.2 states this interpretive point directly at the level of parameterization. Real datasets contain structure (sparsity, bounded dependency depth, separable utilities) that the tractable subcases capture. Engineering work therefore follows two tracks: (a) detecting and exploiting structural restrictions in inputs, and (b) designing heuristic or approximate methods for currently unstructured instances while those structural hypotheses are developed.

A Lean 4 Proof Listings

The complete Lean 4 formalization is available in the companion artifact (Zenodo DOI listed on the title page). The mechanization consists of approximately 5,900 lines across 39 files, with 230+ theorem/lemma statements.

A.1 What Is Machine-Checked

The Lean formalization establishes:

1. **Correctness of the TAUTOLOGY reduction:** The theorem `tautology_iff_sufficient` proves that the mapping from Boolean formulas to decision problems preserves the decision structure (accept iff the formula is a tautology).

2. **Decision problem definitions:** Formal definitions of sufficiency, optimality, and the decision quotient.

3. **Economic theorems:** The Simplicity Tax conservation laws and hardness distribution results.

Complexity classifications (coNP-completeness, Σ_2^P-completeness) follow from informal composition with standard results (TAUTOLOGY is coNP-complete, etc.). The Lean proofs verify the reduction constructions; the complexity class membership is derived by combining these with established theorems from complexity theory.

A.2 Polynomial-Time Reduction Definition

We use Mathlib's Turing machine framework to define polynomial-time computability:

```
/-- Polynomial-time computable function using Turing machines -/
def PolyTime {α β : Type} (ea : FinEncoding α) (eb : FinEncoding β)
    (f : α → β) : Prop :=
  Nonempty (Turing.TM2ComputableInPolyTime ea eb f)

/-- Polynomial-time many-one (Karp) reduction -/
def ManyOneReducesPoly {α β : Type} (ea : FinEncoding α) (eb : FinEncoding β)
```

```
    (A : Set α) (B : Set β) : Prop :=
  ∃ f : α → β, PolyTime ea eb f ∧ ∀ x, x ∈ A ↔ f x ∈ B
```

This uses the standard definition: a reduction is polynomial-time computable via Turing machines and preserves membership.

A.3 The Main Reduction Theorem

Theorem A.1 (TAUTOLOGY Reduction Correctness, Lean). *The reduction from TAUTOLOGY to SUFFICIENCY-CHECK is correct:*

```
theorem tautology_iff_sufficient (φ : Formula n) :
    φ.isTautology ↔ (reductionProblem φ).isSufficient Finset.empty
```

This theorem is proven by showing both directions:

- If φ is a tautology, then the empty coordinate set is sufficient
- If the empty coordinate set is sufficient, then φ is a tautology

The proof verifies that the utility construction in `reductionProblem` creates the appropriate decision structure where:

- At reference states, `accept` is optimal with utility 1
- At assignment states, `accept` is optimal iff $\varphi(a) = \text{true}$

A.4 Economic Results

The hardness distribution theorems (Section 8) are fully formalized:

```
theorem simplicityTax_conservation (P : SpecificationProblem)
    (S : SolutionArchitecture P) :
    S.centralDOF + simplicityTax P S ≥ P.intrinsicDOF

theorem simplicityTax_grows (P : SpecificationProblem)
    (S : SolutionArchitecture P) (n₁ n₂ : ℕ)
    (hn : n₁ < n₂) (htax : simplicityTax P S > 0) :
    totalDOF S n₁ < totalDOF S n₂

theorem native_dominates_manual (P : SpecificationProblem) (n : Nat)
    (hn : n > P.intrinsicDOF) :
    totalDOF (nativeTypeSystem P) n < totalDOF (manualApproach P) n

theorem totalDOF_eventually_constant_iff_zero_distributed
    (S : SolutionArchitecture P) :
    IsEventuallyConstant (fun n => totalDOF S n) ↔ S.distributedDOF = 0

theorem no_positive_slope_linear_represents_saturating
    (c d K : ℕ) (hd : d > 0) :
    ¬ (∀ n, c + n * d = generalizedTotalDOF c (saturatingSiteCost K) n)
```

A.5 Engineering Corollary Mapping

Paper handle	Lean theorem
cor:overmodel-diagnostic-implication	DecisionQuotient.Sigma2PHardness.sufficient_iff_relevant_subset
cor:practice-diagnostic	DecisionQuotient.Sigma2PHardness.min_sufficient_set_iff_relevant
cor:practice-bounded	DecisionQuotient.sufficiency_poly_bounded_actions
cor:practice-structured	DecisionQuotient.sufficiency_poly_separable
cor:practice-tree	DecisionQuotient.sufficiency_poly_tree_structured
cor:practice-unstructured	DecisionQuotient.all_coords_relevant_of_not_tautology
cor:right-wrong-hardness	DecisionQuotient.HardnessDistribution.right_dominates_wrong
cor:type-system-threshold	DecisionQuotient.HardnessDistribution.native_dominates_manual
cor:gap-externalization	DecisionQuotient.HardnessDistribution.simplicityTax_grows
cor:gap-minimization-hard	DecisionQuotient.Sigma2PHardness.min_sufficient_set_iff_relevant
thm:linear-saturation-iff-zero	DecisionQuotient.HardnessDistribution.totalDOF_eventually_consta
thm:generalized-saturation-possible	DecisionQuotient.HardnessDistribution.generalizedTotal_with_satu
cor:linear-positive-no-saturation	DecisionQuotient.HardnessDistribution.no_positive_slope_linear_r

A.6 Module Structure

- `Basic.lean` – Core definitions (DecisionProblem, sufficiency, optimality)
- `Sufficiency.lean` – Sufficiency checking algorithms and properties
- `Reduction.lean` – TAUTOLOGY reduction construction and correctness
- `Complexity.lean` – Polynomial-time reduction definitions using mathlib
- `HardnessDistribution.lean` – Simplicity Tax theorems
- `Tractability/` – Bounded actions, separable utilities, tree structure

A.7 Verification

The proofs compile with Lean 4 and contain no `sorry` placeholders. Run `lake build` in the proof directory to verify.

References

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

[2] Edoardo Amaldi and Viggo Kann. On the approximability of some maximum spanning tree problems. *Theoretical Computer Science*, 196(1-2):3–19, 1998.

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

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

[5] 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.

[6] Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *International Conference on Automated Deduction*, pages 625–635. Springer, 2021.

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

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

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

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

[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.