Computational Complexity of Sufficiency in Decision Problems

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

February 21, 2026

Zenodo DOI: 10.5281/zenodo.18140965

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.
- **Intermediate query-access lower bounds:** In mechanized Boolean query submodels, SUFFICIENCY-CHECK has worst-case query complexity $\Omega(2^n)$ for 'Opt'-oracle, value-entry, and state-batch interfaces via indistinguishable yes/no pairs for the $I = \emptyset$ subproblem; the obstruction is seedwise robust under finite-support randomization.

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 $\mathrm{sufficient}(I) \iff \mathrm{Relevant} \subseteq I$. Within this regime-typed framework, over-modeling is rational only under integrity-preserving attempted-competence-failure conditions; otherwise it is irrational.

The contribution has two levels: (i) a complete complexity landscape for the core decision-relevant problems in the formal model (coNP/Σ_2^P completeness and tractable regimes under explicit encoding assumptions), and (ii) a formal regime-typing framework that separates structural complexity from representational hardness and yields theorem-indexed engineering corollaries.

The reduction constructions and key equivalence theorems are machine-checked in Lean 4 (7741 lines, 350 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

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. Section 2.5 gives the model contract and regime tags used to type every strong claim.

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 Hardness–Recovery Reading Map

Theorems in this paper are intended to be read as hardness/recovery pairs for the same fixed decision relation:

1. **Sufficiency checking:** [S] coNP-complete in the general succinct regime (Theorem 4.6), paired with polynomial-time recovery under explicit-state and structured-access regimes (Theorem 6.1).

2. **Minimal sufficient sets:** [S] coNP-complete in general (Theorem 4.7), paired with the collapse criterion and relevance-computation recovery route (Theorems 4.13, 6.1).

3. **Anchor sufficiency:** Σ_2^{P}-complete in the general anchored formulation (Theorem 4.9); no general polynomial-time recovery condition is established in this model.

(Lean anchors for this map: `ClaimClosure.sufficiency_conp_complete_conditional`, `ClaimClosure.minsuff_conp_complete_conditional`, `ClaimClosure.minsuff_collapse_core`, `ClaimClosure.anchor_sigma2p_complete_conditional`, `ClaimClosure.tractable_subcases_conditional`, `Sigma2PHardness.min_sufficient_set_iff_relevant_card`.)

An operational criterion follows later from the same chain: once decision-site count exceeds the amortization threshold n^*, avoiding structural identification is more expensive than paying the one-time centralization cost (Theorem 9.6). *(Lean: `HardnessDistribution.complete_model_dominates_after_threshold`, `HardnessDistribution.totalExternalWork_eq_n_mul_gapCard`.)*

1.2 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 4.6).

When is it intractable? The general problem is coNP-complete (Theorem 4.6), 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 $\text{Opt}(s)$ 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 intermediate query-access model: the mechanized Boolean submodels yield full worst-case lower bounds $\Omega(2^n)$ for SUFFICIENCY-CHECK for 'Opt'-oracle, value-entry, and state-batch interfaces, with explicit subproblem-to-full transfer and seedwise randomized robustness (Propositions 5.3–5.7 and 5.5, 5.6).

- 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 explicit ETH lower bound is still a succinct worst-case statement; Propositions 5.3 and 5.4 strengthen the intermediate-regime treatment with theorem-level query lower bounds in mechanized Boolean oracle models.

1.3 Main Theorems

1. **Theorem 4.6:** SUFFICIENCY-CHECK is coNP-complete via reduction from TAUTOLOGY.

2. **Theorem 4.7:** MINIMUM-SUFFICIENT-SET is coNP-complete (the Σ_2^{P} structure collapses).

3. **Theorem 4.9:** ANCHOR-SUFFICIENCY is Σ_2^{P}-complete via reduction from $\exists\forall$-SAT.

4. **Theorem 5.1:** 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 6.1:** Polynomial algorithms for bounded actions, separable utility, tree structure.

1.4 Machine-Checked Proofs

The reduction constructions and key equivalence theorems are machine-checked in Lean 4 [7] (7741 lines, 350 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; (iii) a complete complexity landscape for coordinate sufficiency with explicit encoding assumptions; and (iv) a formal separation between structural complexity and representational hardness used to derive theorem-indexed engineering corollaries. Prior work establishes hardness informally; we formalize the constructions and their regime-typed consequences.

1.5 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). A second contribution is formal claim typing: Section 3 introduces the structural/representational and integrity/competence splits that type-check the applied corollaries.

Section 2: decision-problem foundations and encoding model. Section 3: structural vs representational hardness; integrity vs competence. Section 4: hardness proofs. Section 5: regime separation. Section 6: tractable cases. Section 7: engineering corollaries by regime. Section 8: software architecture corollaries. Section 9: complexity redistribution corollary. Section 10: 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 [15, 14].

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:

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

Equivalently, the optimal action depends only on coordinates in I. In this paper, this set-valued invariance of the full optimal-action correspondence is the primary decision-relevance target.

Definition 2.5 (Deterministic Selector)**.** A *deterministic selector* is a map $\sigma : 2^A \to A$ that returns one action from an optimal-action set.

Definition 2.6 (Selector-Level Sufficiency)**.** For fixed selector σ, a coordinate set I is *selector-sufficient* if

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

Proposition 2.7 (Set-Level Target Dominates Selector Targets)**.** *If I is sufficient in the set-valued sense (Definition 2.4), then I is selector-sufficient for every deterministic selector σ (Definition 2.6). (Lean: `DecisionProblem.sufficient_implies_selectorSufficient`)*

Proof. If I is sufficient, then $s_I = s'_I$ implies $\text{Opt}(s) = \text{Opt}(s')$. Applying any deterministic selector σ to equal optimal-action sets yields equal selected actions. ■

Definition 2.8 (ε-Optimal Set and ε-Sufficiency)**.** For $\varepsilon \geq 0$, define

$$\text{Opt}_\varepsilon(s) := \{a \in A : \forall a' \in A,\ U(a', s) \leq U(a, s) + \varepsilon\}.$$

Coordinate set I is *ε-sufficient* if

$$\forall s, s' \in S : \ s_I = s'_I \implies \text{Opt}_\varepsilon(s) = \text{Opt}_\varepsilon(s').$$

Proposition 2.9 (Zero-ε Reduction)**.** *Set-valued sufficiency is exactly the $\varepsilon = 0$ case:*

$$I \text{ sufficient} \iff I \text{ is } 0\text{-sufficient.}$$

(Lean: `DecisionProblem.epsOpt_zero_eq_opt`, `DecisionProblem.sufficient_iff_zeroEpsilonSufficient`)

Proposition 2.10 (Selector-Level Separation Witness)**.** *The converse of Proposition 2.7 fails in general: there exists a decision problem, selector, and coordinate set I such that I is selector-sufficient but not sufficient in the full set-valued sense. (Lean witness: `ClaimClosure.selectorSufficient_not_implies_setSufficient`.)*

Definition 2.11 (Minimal Sufficient Set)**.** A sufficient set I is *minimal* if no proper subset $I' \subsetneq I$ is sufficient.

Definition 2.12 (Relevant Coordinate)**.** Coordinate i is *relevant* if there exist states that differ only at coordinate i and induce different optimal action sets:

$$i \text{ is relevant} \iff \exists s, s' \in S : \Big(\forall j \neq i,\ s_j = s'_j\Big) \wedge \text{Opt}(s) \neq \text{Opt}(s').$$

Proposition 2.13 (Minimal-Set/Relevance Equivalence)**.** *For any minimal sufficient set I and any coordinate i:*

$$i \in I \iff i \text{ is relevant.}$$

Hence every minimal sufficient set is exactly the relevant-coordinate set. (Lean product-space handles: `DecisionProblem.minimalSufficient_iff_relevant`*,* `DecisionProblem.relevantSet_is_minimal`*.)*

Proof. The "only if" direction follows by minimality: if $i \in I$ were irrelevant, removing i would preserve sufficiency, contradicting minimality. The "if" direction follows from sufficiency: every sufficient set must contain each relevant coordinate. ■

Definition 2.14 (Exact Relevance Identifiability)**.** For a decision problem $\mathcal{D}$ and candidate coordinate set I, we say I is *exactly relevance-identifying* if

$$\forall i \in \{1, \ldots, n\} : \quad i \in I \iff i \text{ is relevant for } \mathcal{D}.$$

Equivalently, I is exactly relevance-identifying iff I equals the full relevant-coordinate set.

Example 2.15 (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 2.16 (Decision Equivalence)**.** For coordinate set I, states s, s' are *I-equivalent* (written $s \sim_I s'$) if $s_I = s'_I$.

Definition 2.17 (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 are optimal for at least one state consistent with I.

Proposition 2.18 (Sufficiency Characterization)**.** *Coordinate set I is sufficient if and only if $DQ_I(s) = |Opt(s)|/|A|$ for all $s \in S$. (Lean finite-model form:* `ClaimClosure.sufficiency_iff_dq_ratio`*,* `ClaimClosure.sufficiency_iff_projectedOptCover_eq_opt`*)*

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). ■

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 [4]. 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 4 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.

Query-access regime (intermediate black-box access). The solver is given oracle access to decision information at queried states (e.g., $\text{Opt}(s)$, or $U(a, s)$ with $\text{Opt}(s)$ reconstructed from finitely many value queries). Complexity is measured by oracle-query count (optionally paired with per-query evaluation cost). This separates representational access from full-table availability and from succinct-circuit input length.

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

2.5 Model Contract and Regime Tags

All theorem statements in this paper are typed by the following model contract:

- **C1 (finite actions):** A is finite.
- **C2 (finite coordinate domains):** each X_i is finite, so $S = \prod_i X_i$ is finite.
- **C3 (evaluable utility):** $U(a, s)$ is computable from the declared input encoding.
- **C4 (fixed decision semantics):** optimality is defined by $\text{Opt}(s) = \arg\max_a U(a, s)$.

We use short regime tags for applied corollaries:

- **[E]** explicit-state encoding,
- **[Q]** query-access (oracle) regime,
- **[S]** succinct encoding,
- **[S+ETH]** succinct encoding with ETH,
- **[Q_bool]** mechanized Boolean query submodel,
- **[S_bool]** mechanized Boolean-coordinate submodel.

This tagging is a claim-typing convention: each strong statement is attached to the regime where it is proven.

2.6 Adjacent Objective Regimes and Bridge

Definition 2.19 (Adjacent Sequential Objective Regime)**.** An adjacent sequential objective instance consists of:

- finite action set A,
- finite coordinate state space $S = X_1 \times \cdots \times X_n$,
- horizon $T \in \mathbb{N}_{\geq 1}$ and history-dependent policy class,
- reward process r_t and objective functional $J(\pi)$ (e.g., cumulative reward or regret).

Proposition 2.20 (One-Step Deterministic Bridge)**.** *Consider an instance of Definition 2.19 satisfying:*

1. *$T = 1$,*
2. *deterministic rewards $r_1(a, s) = U(a, s)$ for some evaluable U,*
3. *objective $J(\pi) = U(\pi(s), s)$ (single-step maximization),*
4. *no post-decision state update relevant to the objective.*

Then the induced optimization problem is exactly the static decision problem of Definition 2.1, and coordinate sufficiency in the sequential formulation is equivalent to Definition 2.4. (Lean bridge core: `ClaimClosure.one_step_bridge`)

Proof. Under (1)–(3), optimizing J at state s is identical to choosing an action in $\arg\max_{a \in A} U(a, s) = \text{Opt}(s)$. Condition (4) removes any dependence on future transition effects. Therefore the optimal-policy relation in the adjacent formulation coincides pointwise with Opt from Definition 2.3, and the invariance condition "same projection implies same optimal choice set" is exactly Definition 2.4. ■

Proposition 2.21 (Bridge Transfer Rule)**.** *Under conditions (1)–(4), any sufficiency statement formulated over Definition 2.4 is equivalent between the adjacent sequential formulation and the static model. (Lean: `ClaimClosure.one_step_bridge`)*

Proof. By Proposition 2.20, the adjacent sequential objective induces exactly the same sufficiency predicate as Definition 2.4 when (1)–(4) hold. Equivalence of any sufficiency statement then follows by substitution. ■

If any bridge condition fails, direct transfer from this paper's static complexity theorems is not licensed by this rule.

Remark 2.22 (Extension Boundary). Beyond Proposition 2.20 (multi-step horizon, stochastic transitions/rewards, or regret objectives), the governing complexity objects change. Those regimes are natural extensions, but they are distinct formal classes from the static sufficiency class analyzed in this paper.

3 Interpretive Foundations: Hardness and Solver Claims

The claims in later applied sections are theorem-indexed consequences of this section and Sections 4–6.

3.1 Structural Complexity vs Representational Hardness

Definition 3.1 (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 3.2 (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 3.3 (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.

3.2 Solver Integrity and Regime Competence

To keep practical corollaries type-safe, we separate *integrity* (what a solver is allowed to assert) from *competence* (what it can cover under a declared regime), following the certifying-algorithms schema [11].

Definition 3.4 (Certifying Solver)**.** Fix a decision relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ and an encoding regime E over $\mathcal{X}$. A *certifying solver* is a pair (Q, V) where:

- Q maps each input $x \in \mathcal{X}$ to either $\mathsf{ABSTAIN}$ or a candidate pair (y, w),
- V is a polynomial-time checker (in $|\mathrm{enc}_E(x)|$) with output in $\{0, 1\}$.

Definition 3.5 (Solver Integrity)**.** A certifying solver (Q, V) has *integrity* for relation $\mathcal{R}$ if:

- (assertion soundness) $Q(x) = (y, w) \implies V(x, y, w) = 1$,
- (checker soundness) $V(x, y, w) = 1 \implies (x, y) \in \mathcal{R}$.

The output $\mathsf{ABSTAIN}$ (equivalently, $\mathsf{UNKNOWN}$) is first-class and carries no assertion about membership in $\mathcal{R}$.

Definition 3.6 (Competence Under a Regime)**.** Fix a regime $\Gamma = (\mathcal{X}_\Gamma, E_\Gamma, \mathcal{C}_\Gamma)$ with instance family $\mathcal{X}_\Gamma \subseteq \mathcal{X}$, encoding assumptions E_Γ, and resource bound $\mathcal{C}_\Gamma$. A certifying solver (Q, V) is *competent* on Γ for relation $\mathcal{R}$ if:

- it has integrity for $\mathcal{R}$ (Definition 3.5),
- (coverage) $\forall x \in \mathcal{X}_\Gamma,\ Q(x) \neq \mathsf{ABSTAIN}$,
- (resource bound) $\mathrm{runtime}_Q(x) \leq \mathcal{C}_\Gamma(|\mathrm{enc}_{E_\Gamma}(x)|)$ for all $x \in \mathcal{X}_\Gamma$.

Proposition 3.7 (Integrity–Competence Separation)**.** *Integrity and competence are distinct predicates: integrity constrains asserted outputs, while competence adds non-abstaining coverage under resource bounds.*

Proof. Take the always-abstain solver $Q_\perp(x) = \mathsf{ABSTAIN}$ with any polynomial-time checker V. Definition 3.5 holds vacuously, so $(Q_\perp, V)$ is integrity-preserving, but it fails Definition 3.6 whenever $\mathcal{X}_\Gamma \neq \emptyset$ because coverage fails. Hence integrity does not imply competence. The converse is immediate because competence includes integrity as a conjunct. ■

Definition 3.8 (Attempted Competence Failure)**.** Fix an exact objective under regime Γ. A solver state is an *attempted competence failure* if:

- integrity holds (Definition 3.5),
- exact competence was actually attempted for the active scope/objective,
- competence on Γ fails for that exact objective (Definition 3.6).

Proposition 3.9 (Attempted-Competence Rationality Matrix)**.** *Let $I, A, C \in \{0, 1\}$ denote integrity, attempted exact competence, and competence available in the active regime. Policy verdict for persistent over-specification is:*

- *if $I = 0$: inadmissible,*
- *if $I = 1$ and $(A, C) = (1, 0)$: conditionally rational,*
- *if $I = 1$ and $(A, C) \in \{(0, 0), (0, 1), (1, 1)\}$: irrational for the same verified-cost objective.*

Hence, in the integrity-preserving plane ($I = 1$), exactly one cell is rational and three are irrational. (Lean: `IntegrityCompetence.overModelVerdict_rational_iff`, `IntegrityCompetence.admissible_matrix_counts`, `IntegrityCompetence.admissible_irrational_strictly_more_than_rational`)

This separation plus Proposition 3.9 is load-bearing for the regime-conditional trilemma used later: if exact competence is blocked by hardness in a declared regime after an attempted exact procedure, integrity forces one of three responses—abstain, weaken guarantees, or change regime assumptions.

Mechanized status. This separation is machine-checked in `DecisionQuotient/IntegrityCompetence.lean` via: `competence_implies_integrity` and `integrity_not_competent_of_nonempty_scope`; the attempted-competence matrix is mechanized via `overModelVerdict_rational_iff`, `admissible_matrix_counts`, and `admissible_irrational_strictly_more_than_rational`.

4 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

Under the model contract of Section 2.5 and the succinct encoding of Section 2.4, these results place exact relevance identification beyond NP-completeness: in the worst case, finding (or certifying) minimal decision-relevant sets is computationally intractable.

Reading convention for this section. Each hardness theorem below is paired with a recovery boundary for the same decision relation: when structural-access assumptions in Theorem 6.1 hold, polynomial-time exact procedures are recovered; when they fail in [S], the stated hardness applies. *(Lean map handles: `ClaimClosure.sufficiency_conp_complete_conditional`, `ClaimClosure.minsuff_conp_complete_conditional`, `ClaimClosure.anchor_sigma2p_complete_conditional`, `ClaimClosure.tractable_subcases_conditional`.)*

4.1 Problem Definitions

Definition 4.1 (Decision Problem Encoding)**.** A *decision problem instance* is a tuple $(A, X_1, \dots, X_n, U)$ where:

- A is a finite set of alternatives
- $X_1, \dots, X_n$ 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 4.2 (Optimizer Map)**.** For state $s \in S$, define:

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

Definition 4.3 (Sufficient Coordinate Set)**.** A coordinate set $I \subseteq \{1, \dots, n\}$ 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 4.4 (SUFFICIENCY-CHECK)**. Input:** Decision problem $(A, X_1, \dots, X_n, U)$ and coordinate set $I \subseteq \{1, \dots, n\}$
Question: Is I sufficient?

Problem 4.5 (MINIMUM-SUFFICIENT-SET)**. Input:** Decision problem $(A, X_1, \dots, X_n, U)$ and integer k
Question: Does there exist a sufficient set I with $|I| \leq k$?

4.2 Hardness of SUFFICIENCY-CHECK

Theorem 4.6 (coNP-completeness of SUFFICIENCY-CHECK)**.** *SUFFICIENCY-CHECK is coNP-complete [6, 10]. (Lean handles: `ClaimClosure.sufficiency_conp_reduction_core`, `ClaimClosure.sufficiency_conp_complete_conditional`, `tautology_iff_sufficient`.)*

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

Proof. **Membership in coNP:** The complementary problem INSUFFICIENCY is in NP. Given a decision problem $(A, X_1, \dots, X_n, U)$ 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. $\mathrm{Opt}(s) \neq \mathrm{Opt}(s')$ (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: $S = \{\text{reference}\} \cup \{0,1\}^n$ (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 $\iff$ 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. ■

Immediate recovery boundary (SUFFICIENCY-CHECK). Theorem 4.6 is the [S] general-case hardness statement. For the same sufficiency relation, Theorem 6.1 gives polynomial-time recovery under explicit-state, separable, and tree-structured regimes. *(Lean bridge handles: `ClaimClosure.sufficiency_conp_complete_conditional`, `ClaimClosure.tractable_subcases_conditional`, `ClaimClosure.tractable_bounded_core`, `ClaimClosure.tractable_separable_core`, `ClaimClosure.tractable_tree_core`.)*

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 5.1, 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.

4.3 Complexity of MINIMUM-SUFFICIENT-SET

Theorem 4.7 (MINIMUM-SUFFICIENT-SET is coNP-complete)**.** *MINIMUM-SUFFICIENT-SET is coNP-complete. (Lean handles: `ClaimClosure.minsuff_collapse_core`, `ClaimClosure.minsuff_conp_complete_conditional`.)*

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

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

However, this collapses because sufficiency has a simple characterization.

Key lemma: In the Boolean-coordinate collapse model, a coordinate set I is sufficient if and only if I contains all relevant coordinates (proven formally as `sufficient_iff_relevant_subset` / `sufficient_iff_relevantFinset_subset` in Lean):

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

where $\mathrm{Relevant} = \{i : \exists s, s'.\ s \text{ differs from } s' \text{ only at } i \text{ and } \mathrm{Opt}(s) \neq \mathrm{Opt}(s')\}$. This is the same relevance object as Definition 2.12; Proposition 2.13 gives the minimal-set equivalence in the product-space semantics used by the collapse.

Consequence: The minimum sufficient set is exactly the relevant-coordinate set. 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 4.6. ■

Immediate recovery boundary (MINIMUM-SUFFICIENT-SET). Theorem 4.7 pairs with Theorem 4.13: exact minimization complexity is governed by relevance-cardinality once the collapse is applied. Recovery then depends on computing relevance efficiently, with structured-access assumptions from Theorem 6.1 giving the corresponding route for the underlying sufficiency computations. *(Lean bridge handles: `ClaimClosure.minsuff_conp_complete_conditional`, `ClaimClosure.minsuff_collapse_core`, `ClaimClosure.minsuff_collapse_to_conp_conditional`, `Sigma2PHardness.min_sufficient_set_iff_relevant_card`, `ClaimClosure.tractable_subcases_conditional`.)*

4.4 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 4.8 (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 4.9 (ANCHOR-SUFFICIENCY is Σ_2^P-complete)**.** *ANCHOR-SUFFICIENCY is Σ_2^P-complete [17] (already for Boolean coordinate spaces). (Lean handles: `ClaimClosure.anchor_sigma2p_reduction_core`, `ClaimClosure.anchor_sigma2p_complete_conditional`, `anchor_sufficiency_sigma2p`.)*

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. ■

Immediate boundary statement (ANCHOR-SUFFICIENCY). Theorem 4.9 remains a second-level hardness statement in the anchored formulation; unlike MINIMUM-SUFFICIENT-SET, no general collapse to relevance counting is established here, so the corresponding tractability status remains open in this model. *(Lean anchor handles: `ClaimClosure.anchor_sigma2p_reduction_core`, `ClaimClosure.anchor_sigma2p_complete_conditional`, `anchor_sufficiency_sigma2p`.)*

4.5 Tractable Subcases

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

Proposition 4.10 (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 4.11 (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.*

4.6 Implications

Corollary 4.12 (Computational Bottleneck for Exact Minimization)**.** *Under the succinct encoding, exact minimization of sufficient coordinate sets is coNP-hard via the $k = 0$ case, and fixed-anchor minimization is Σ_2^P-complete. (Lean handles:* `ClaimClosure.minsuff_conp_complete_conditional`*,* `ClaimClosure.anchor_sigma2p_complete_conditional`*)*

Proof. The $k = 0$ case of MINIMUM-SUFFICIENT-SET is SUFFICIENCY-CHECK (Theorem 4.6), giving coNP-hardness for exact minimization. The fixed-anchor variant is exactly Theorem 4.9. ■

The modeling budget for deciding what to model is therefore a computationally constrained resource under this encoding.

4.7 Quantifier Collapse for MINIMUM-SUFFICIENT-SET

Theorem 4.13 (Collapse of the Apparent ∃∀ Structure)**.** *The apparent second-level predicate*

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

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

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

Consequently, MINIMUM-SUFFICIENT-SET is governed by coNP certificates rather than a genuine Σ_2^P alternation. (Lean handles: `ClaimClosure.minsuff_collapse_core`*,* `ClaimClosure.minsuff_collapse_to_conp_conditional`*,* `Sigma2PHardness.min_sufficient_set_iff_relevant_card`*.)*

Proof. By the formal equivalence `sufficient_iff_relevant_subset` (finite-set form `sufficient_iff_relevantFinset_subset`), a coordinate set I is sufficient iff Relevant $\subseteq I$. Therefore:

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

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

A NO certificate for $|\text{Relevant}| \leq k$ 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 4.7.

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. ■

5 Encoding-Regime Separation

The hardness results of Section 4 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, and an intermediate query-access obstruction family.

Model note. Theorem 5.1 has Part 1 in [E] (time polynomial in $|S|$) and Part 2 in [S+ETH] (time exponential in n). Propositions 5.3 and 5.4 are [Q_bool] (query-count lower bounds for two oracle interfaces). These regimes are defined in Section 2.4 and are not directly comparable as functions of one numeric input-length parameter.

Theorem 5.1 (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. ***[E] 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. ***[S+ETH] Succinct lower bound (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 $2^{\Omega(n)}$.*

(Lean handles: `ClaimClosure.dichotomy_conditional`, `ClaimClosure.explicit_state_upper_core`, `ClaimClosure.hard_family_all_coords_core`.)

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 $\text{Opt}(s)$ 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 $|S_I| \le |S| = N$, 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 $|S_I| \le \prod_{i \in I} |X_i|$ (and $|S_I| \le N$). If the coordinate domains are uniformly bounded, $|X_i| \le d$ for all i, then $|S_I| \le d^{|I|}$.

Part 2 (Succinct ETH lower bound, worst case): A strengthened version of the TAUTOLOGY reduction used in Theorem 4.6 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) [9], 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 5.2 (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 $k^* = \Omega(n)$.

Proposition 5.3 (Full Query-Access Lower Bound via Empty-Set Subproblem)**.** *In the mechanized Boolean query regime [Q_bool], fix $n > 0$ and let Q be any queried-state set with $|Q| < 2^n$. Then there exist two decision problems $\mathcal{D}_{\text{yes}}, \mathcal{D}_{\text{no}}$ over the same state space such that:*

- *their oracle views on all states in Q are identical,*
- *$\emptyset$ is sufficient for $\mathcal{D}_{\text{yes}}$,*
- *$\emptyset$ is not sufficient for $\mathcal{D}_{\text{no}}$.*

Consequently, no deterministic query procedure using fewer than 2^n state queries can solve SUFFICIENCY-CHECK on all such instances; the worst-case query complexity is $\Omega(2^n)$. (Lean: `emptySufficiency_query_indistinguishable_pair`, `constTrue_empty_sufficient`, `spike_empty_not_sufficient`; obstruction-scale companion: `queryComplexityLowerBound`, `exponential_query_complexity`.)

Proof. The theorem `emptySufficiency_query_indistinguishable_pair` provides, for any $|Q| < 2^n$, two oracle-indistinguishable instances with opposite truth values for the $\emptyset$-sufficiency predicate. Since SUFFICIENCY-CHECK contains the $I = \emptyset$ case, any deterministic solver that queries fewer than 2^n states cannot be correct on both instances, yielding the $\Omega(2^n)$ worst-case query lower bound in this regime. The scale identities `queryComplexityLowerBound` and `exponential_query_complexity` remain as companion obstruction statements. ■

Proposition 5.4 (Value-Entry Query Lower Bound). *In the mechanized Boolean value-entry query submodel [Q_ bool], for any deterministic procedure using fewer than 2^n value-entry queries $(a, s) \mapsto U(a, s)$, there exist two queried-value-indistinguishable instances with opposite truth values for SUFFICIENCY-CHECK on the $I = \emptyset$ subproblem. Therefore worst-case value-entry query complexity is also $\Omega(2^n)$. (Lean: `emptySufficiency_valueEntry_indistinguishable_pair`, `touchedStates_card_le_query_card`.)*

Proof. The theorem `emptySufficiency_valueEntry_indistinguishable_pair` constructs, for any query set of cardinality $< 2^n$, an unqueried hidden state s_0 and a yes/no instance pair that agree on all queried values but disagree on $\emptyset$-sufficiency. The auxiliary bound `touchedStates_card_le_query_card` ensures that fewer than 2^n value-entry queries cannot cover all states, so the indistinguishability argument applies in the worst case. ■

Proposition 5.5 (Subproblem-to-Full Transfer Rule). *If every full-problem solver induces a solver for a fixed subproblem, then any lower bound for that subproblem lifts to the full problem. (Lean closure: `ClaimClosure.subproblem_hardness_lifts_to_full`.)*

Proposition 5.6 (Randomized Robustness (Seedwise)). *In [Q_ bool], for any query set with cardinality $< 2^n$, the indistinguishable yes/no pair from Proposition 5.3 forces one decoding error* per random seed *for any seed-indexed decoder from oracle transcripts. Consequently, finite-support randomization does not remove the obstruction: averaging preserves a constant error floor on the hard pair. (Lean: `indistinguishable_pair_forces_one_error`, `indistinguishable_pair_forces_one_error_per_seed`, `decode_error_sum_two_labels`.)*

Proposition 5.7 (State-Batch Query Lower Bound). *In [Q_ bool], the same $\Omega(2^n)$ lower bound holds for a state-batch oracle that returns the full Boolean-action utility tuple at each queried state. (Lean: `emptySufficiency_stateBatch_indistinguishable_pair`, `stateBatchView_eq_if_hidden_untouched`.)*

Proposition 5.8 (Finite-State Empty-Subproblem Generalization). *The empty-subproblem indistinguishability lower-bound core extends from Boolean-vector state spaces to any finite state type with at least two states. (Lean: `emptySufficiency_query_indistinguishable_pair_finite`, `spikeFinite_empty_not_sufficient`.)*

Proposition 5.9 (Adversary-Family Tightness by Full Scan)**.** *For the const/spike adversary family used in the query lower bounds, querying all states distinguishes the pair; thus the lower-bound family is tight up to constant factors under full-state scan. (Lean: `full_query_distinguishes_const_spike_finite`.)*

Proposition 5.10 (Weighted Query-Cost Transfer)**.** *Let $w(q)$ be per-query cost and $w_{\min}$ a lower bound on all queried costs. Any cardinality lower bound $|Q| \geq L$ lifts to weighted cost:*

$$\sum_{q \in Q} w(q) \;\geq\; w_{\min} \cdot L.$$

(Lean: `weightedQueryCost_ge_min_mul_card`, `weightedQueryCost_ge_min_mul_of_card_lb`.)

Remark 5.11 (Instantiation of Definitions 3.1 and 3.2). Theorem 5.1 and Propositions 5.3–5.4 keep the structural problem fixed (same sufficiency relation) and separate representational hardness by access regime: explicit-state access exposes the boundary $s \mapsto \mathrm{Opt}(s)$, query access yields $\Omega(2^n)$ lower bounds in mechanized Boolean submodels (for both 'Opt' and value-entry interfaces), and succinct access can hide structure enough to force ETH-level worst-case cost on a hard family.

This regime-typed landscape identifies tractability under [E], intermediate [Q_bool] lower bounds for two oracle interfaces, and worst-case intractability under [S+ETH] for the same underlying decision relation.

6 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 6.1 (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.

(Lean handles: `ClaimClosure.tractable_bounded_core`, `ClaimClosure.tractable_separable_core`, `ClaimClosure.tractable_tree_core`, `ClaimClosure.tractable_subcases_conditional`.)

6.1 Explicit-State Encoding

Proof of Part 1. Given the full table of $U(a, s)$, compute $\text{Opt}(s)$ 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 $\text{Opt}(s) = \text{Opt}(s')$. 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 $O(|S|^2)$. ■

6.2 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. ■

6.3 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. ■

6.4 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

7 Engineering Corollaries by Regime

This section derives regime-typed engineering corollaries from the core complexity theorems. Theorem 7.2 maps configuration simplification to SUFFICIENCY-CHECK; Theorems 4.6, 4.7, and 5.1 then yield exact minimization consequences under [S] and [S+ETH].

Regime tags used below follow Section 2.5: [S], [S+ETH], [E], [S_bool]. Any prescription that requires exact minimization is constrained by these theorem-level bounds. Theorem 7.4 implies that persistent failure to isolate a minimal sufficient set is a boundary-characterization signal in the current model, not a universal irreducibility claim.

Conditional rationality criterion. For the objective "minimize verified total cost while preserving integrity," over-specification is rational only under *attempted competence failure* in the active regime (Definition 3.8): if exact irrelevance cannot be certified efficiently after an attempted exact procedure, integrity forbids uncertified exclusion. When exact competence is available in the active regime (e.g., Theorem 6.1 and the exact-identifiability criterion), persistent over-specification is irrational relative to that objective because proven-irrelevant coordinates can be removed with certified correctness. Proposition 3.9 makes this explicit: in the integrity-preserving matrix, one cell is rational and three are irrational, so irrationality is the default verdict. *(Lean anchors: `IntegrityCompetence.competence_implies_integrity`, `IntegrityCompetence.integrity_not_competent_of_nonempty_scope`, `IntegrityCompetence.admissible_matrix_counts`, `ClaimClosure.tractable_subcases_conditional`, `Sigma2PHardness.exactlyIdentifiesRelevant_iff_sufficient_and_subset_relevantFinset`.)*

Remark 7.1 (Regime Contract for Engineering Corollaries)*.* All claims in this section are formal corollaries under the declared model assumptions.

- Competence claims are indexed by the regime tuple of Definition 3.6; prescriptions are meaningful only relative to feasible resources under that regime (bounded-rationality feasibility discipline) [1].
- Integrity (Definition 3.5) forbids overclaiming beyond certifiable outputs; ABSTAIN/UNKNOWN is first-class when certification is unavailable.
- Therefore, hardness results imply a regime-conditional trilemma: abstain, weaken guarantees (heuristics/approximation), or change encoding/structural assumptions to recover competence.

7.1 Configuration Simplification is SUFFICIENCY-CHECK

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

Theorem 7.2 (C1–C4: 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. (Lean: `ConfigReduction.config_sufficiency_iff_behavior_preserving`)*

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

- Actions $A = B \cup \{\bot\}$, where $\bot$ is a sentinel "no-observed-behavior" action
- Coordinates X_i = domain of parameter p_i
- State space $S = X_1 \times \cdots \times X_n$
- For $b \in B$, utility $U(b, s) = 1$ if behavior b occurs under configuration s, else 0
- Sentinel utility $U(\bot, s) = 1$ iff no behavior in B occurs under configuration s, else 0

Then

$$\mathrm{Opt}(s) = \begin{cases} \{b \in B : b \text{ occurs under configuration } s\}, & \text{if this set is nonempty,} \\ \{\bot\}, & \text{otherwise.} \end{cases}$$

So the optimizer map exactly encodes observed-behavior equivalence classes, including the empty-behavior case.

Coordinate set I is sufficient iff:

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

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 7.3 (Reduction Scope). The reduction above requires only:

1. a finite behavior set,
2. parameters with finite domains, and
3. an evaluable behavior map from configurations to achieved behaviors.

These are exactly the model-contract premises C1–C3 instantiated for configuration systems.

Theorem 7.4 (S: Over-Modeling as Boundary-Signal Corollary)**.** *By contraposition of Theorem 7.2, if no coordinate set can be certified as exactly relevance-identifying (Definition 2.14) for the modeled system, then the decision boundary is not completely characterized by the current parameterization. (Lean model-level contrapositive:* `ClaimClosure.no_exact_identifier_implies_not_boundary_characterized`*,* `ClaimClosure.boundaryCharacterized_iff_exists_sufficient_subset`*)*

Proof. Assume the decision boundary were completely characterized by the current parameterization. Then, via Theorem 7.2, the corresponding sufficiency instance admits exact relevance membership, hence a coordinate set that satisfies Definition 2.14. Contraposition gives the claim: persistent failure of exact relevance identification signals incomplete characterization of decision relevance in the model. ■

7.2 Cost Asymmetry Under ETH

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

Theorem 7.5 (S+ETH: Cost-Asymmetry Consequence)**.** *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:

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

[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.

Within [S+ETH], persistent over-specification is consistent with unresolved boundary characterization rather than a proof that all included parameters are intrinsically necessary. Conversely, when exact competence is available in the active regime, persistent over-specification is irrational for the same cost-minimization objective. (Lean handles: `ClaimClosure.cost_asymmetry_eth_conditional`, `HardnessDistribution.linear_lt_exponential_plus_constant_eventually`.)

Proof. Under ETH, the TAUTOLOGY reduction used in Theorem 4.6 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 $C_{\text{over}}(k) = O(k)$.

The eventual dominance step is mechanized in `HardnessDistribution.linear_lt_exponential_plus_constant_eventually`: for fixed linear-overhead parameter k and additive constant C_{under} there is n_0 such that $k < 2^n + C_{\text{under}}$ for all $n \geq n_0$. Therefore:

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

For any fixed nonnegative C_{under}, the asymptotic dominance inequality remains and only shifts the finite threshold n_0. ■

Corollary 7.6 (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)*

(Lean conditional closure: `ClaimClosure.no_auto_minimize_of_p_neq_conp`)

Proof. Such an algorithm would solve MINIMUM-SUFFICIENT-SET in polynomial time, contradicting Theorem 4.7 (assuming $\mathsf{P} \neq \mathsf{coNP}$). ■

Remark 7.7. Corollary 7.6 is a formal boundary statement: an always-correct polynomial-time minimizer for arbitrary succinct inputs would collapse P and coNP.

Remark 7.8 (FPT Scope Caveat). The practical force of worst-case hardness depends on instance structure, especially k^*. If SUFFICIENCY-CHECK is FPT in parameter k^*, then small-k^* families can remain tractable even under succinct encodings. The strengthened mechanized gadget (`all_coords_relevant_of_not_tautology`) still proves existence of hard families with $k^* = n$; what is typical in deployed systems is an empirical question outside this formal model.

7.3 Regime-Conditional Operational Corollaries

Theorems 7.4 and 7.5 yield the following conditional operational consequences:

1. Conservative retention under unresolved relevance. If irrelevance cannot be certified efficiently under the active regime, retaining a superset of parameters is a sound conservative policy.

2. Heuristic selection as weakened-guarantee mode. Under [S+ETH], exact global minimization can be exponentially costly in the worst case (Theorem 7.5); methods such as AIC/BIC/cross-validation therefore fit the "weaken guarantees" branch of Definition 3.6.

3. Full-parameter inclusion as an $O(n)$ upper-bound strategy. Under [S+ETH], if exact minimization is unresolved, including all n parameters incurs linear maintenance overhead while avoiding false irrelevance claims.

4. Irrationality outside attempted-competence-failure conditions. If the active regime admits exact competence (tractable structural-access conditions or exact relevance identifiability), or if exact competence was never actually attempted, continued over-specification is not justified by hardness and is irrational relative to the stated objective. *(Lean anchors: `ClaimClosure.tractable_subcases_conditional`, `IntegrityCompetence.admissible_irrational_strictly_more_than_rational`, `Sigma2PHardness.exactlyIdentifiesRelevant_iff_sufficient_and_subset_relevantFinset`.)*

These corollaries are direct consequences of the hardness/tractability landscape: over-specification is an attempted-competence-failure policy, not a default optimum. To move beyond it, one must either shift to tractable regimes from Theorem 6.1 or adopt explicit approximation commitments.

8 Applied Corollaries for Software Architecture

Regime for this section: the mechanized Boolean-coordinate model [S_bool] plus the architecture cost model defined below.

8.1 Over-Specification as Diagnostic Signal

Corollary 8.1 (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: `Sigma2PHardness.sufficient_iff_relevant_subset`)*

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

Corollary 8.2 (Exact Relevance Identifiability Criterion). *In the mechanized Boolean-coordinate model, for any candidate set I:*

$$I \textit{ is exactly relevance-identifying} \iff \big(I \textit{ is sufficient and } I \subseteq R_{\text{rel}}\big),$$

where R_{rel} is the full relevant-coordinate set. (Lean: `Sigma2PHardness.exactlyIdentifiesRelevant_iff_sufficient_and_subset_relevantFinset`)

Proof. This is exactly `Sigma2PHardness.exactlyIdentifiesRelevant_iff_sufficient_and_subset_relevantFinset`, with $R_{\text{rel}} =$ `relevantFinset`. ∎

Remark 8.3 (Over-Specification Is Regime-Conditional, Not Default). In this paper's formal typing, over-specification is rational only under attempted competence failure: exact relevance competence is unavailable in the active regime *after* an attempted exact procedure, and integrity forbids uncertified exclusion (Section 3; Section 7). Once exact competence is available in the active regime (Corollaries 8.7–8.9 together with Corollary 8.2), persistent over-specification is irrational for the same objective (verified total-cost minimization), because excluded coordinates can be certified irrelevant. *(Lean anchors:* `IntegrityCompetence.competence_implies_integrity`*,* `IntegrityCompetence.integrity_not_competent_of_nonempty_scope`*,* `IntegrityCompetence.admissible_matrix_counts`*,* `sufficiency_poly_bounded_actions`*,* `sufficiency_poly_separable`*,* `sufficiency_poly_tree_structured`*,* `Sigma2PHardness.exactlyIdentifiesRelevant_iff_sufficient_and_subset_relevantFinset`*.)*

8.2 Architectural Decision Quotient

Definition 8.4 (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 8.5 (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. (Lean finite-cardinality form:* `ClaimClosure.adq_ordering`*)*

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. ■

8.3 Corollaries for Practice

Corollary 8.6 (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:* `Sigma2PHardness.min_sufficient_set_iff_relevant_card`*)*

Proof. By `Sigma2PHardness.min_sufficient_set_iff_relevant_card`, sufficiency of size $\leq k$ is equivalent to a relevance-cardinality bound $\leq k$ in the Boolean-coordinate model. ■

Corollary 8.7 (Bounded-Regime Tractability)**.** *When the bounded-action or explicit-state conditions of Theorem 6.1 hold, minimal modeling can be solved in polynomial time in the stated input size. (Lean:* `sufficiency_poly_bounded_actions`*)*

Proof. This is the bounded-regime branch of Theorem 6.1, mechanized as `sufficiency_poly_bounded_actions`. ■

Corollary 8.8 (Separable-Utility Tractability)**.** *When utility is separable with explicit factors, sufficiency checking is polynomial in the explicit-state regime. (Lean:* `sufficiency_poly_separable`*)*

Proof. This is the separable-utility branch of Theorem 6.1, mechanized as `sufficiency_poly_separable`. ■

Corollary 8.9 (Tree-Structured Tractability)**.** *When utility factors form a tree structure with explicit local factors, sufficiency checking is polynomial in the explicit-state regime. (Lean:* `sufficiency_poly_tree_structured`*)*

Proof. This is the tree-factor branch of Theorem 6.1, mechanized as `sufficiency_poly_tree_structured`. ■

Corollary 8.10 (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:* `all_coords_relevant_of_not_tautology`*)*

Proof. The strengthened reduction proves that non-tautological source formulas induce instances where every coordinate is relevant; this is mechanized as `all_coords_relevant_of_not_tautology`. ■

8.4 Hardness Distribution: Right Place vs Wrong Place

Definition 8.11 (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 8.12 (Baseline Lower-Bound 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 5.1 provides the baseline on the hard succinct family: $H(\textit{SUFFICIENCY-CHECK}; n) = 2^{\Omega(n)}$ under ETH. (Lean structural core: `HardnessDistribution.totalDOF_ge_intrinsic`*)*

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

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

(Lean ratio identity when denominator is positive: `HardnessDistribution.hardnessEfficiency_eq_central_share`*)*

Proposition 8.14 (Efficiency Equivalence)**.** *For fixed n and positive total hardness, larger $\eta(S, n)$ is equivalent to a larger central share of realized hardness. (Lean definitional step:* `HardnessDistribution.hardnessEfficiency_eq_central_share`*)*

Proof. From Definition 8.13, $\eta(S, n)$ is exactly the fraction of total realized hardness paid centrally. ■

Definition 8.15 (Right vs Wrong Hardness Placement)**.** For a solution architecture S in this linear model:

$$\text{right hardness} \iff H_{\text{distributed}}(S) = 0, \qquad \text{wrong hardness} \iff H_{\text{distributed}}(S) > 0.$$

(Lean: `HardnessDistribution.isRightHardness`, `HardnessDistribution.isWrongHardness`)

Theorem 8.16 (Centralization Dominance)**.** *Let $S_{\text{right}}, S_{\text{wrong}}$ be architectures over the same problem family with*

$$H_{\text{distributed}}(S_{\text{right}}) = 0, \quad H_{\text{central}}(S_{\text{right}}) > 0, \quad H_{\text{distributed}}(S_{\text{wrong}}) > 0,$$

and let $n > \max\big(1, H_{\text{central}}(S_{\text{right}})\big)$. Then:

1. *Lower total realized hardness:*

$$H_{total}(S_{\text{right}}) < H_{total}(S_{\text{wrong}})$$

2. *Fewer error sites: errors in centralized components affect 1 location; errors in distributed components affect n locations*

3. *Quantified leverage: moving one unit of work from distributed to central saves exactly $n - 1$ units of total realized hardness*

(Lean: `HardnessDistribution.centralization_dominance_bundle`, `HardnessDistribution.centralization_step_saves_n_minus_one`)

Proof. (1) and (2) are exactly the bundled theorem `HardnessDistribution.centralization_dominance_bundle`. (3) is exactly `HardnessDistribution.centralization_step_saves_n_minus_one`. ■

Corollary 8.17 (Right-Place vs Wrong-Place Hardness)**.** *In the linear model, a right-hardness architecture strictly dominates a wrong-hardness architecture once use-site count exceeds central one-time hardness. Formally, 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: `HardnessDistribution.right_dominates_wrong`)

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

Proposition 8.18 (Dominance Modes)**.** *This section uses two linear-model dominance modes plus generalized nonlinear dominance and boundary modes:*

1. ***Strict threshold dominance:*** *Corollary 8.17 gives strict inequality once $n > H_{\text{central}}(S_{\text{right}})$.*

2. ***Global weak dominance:*** *under the decomposition identity used in `HardnessDistribution.centralized_higher_leverage`, centralized hardness placement is never worse for all $n \geq 1$.*

3. ***Generalized nonlinear dominance:*** *under bounded-vs-growing site-cost assumptions (Theorem 8.22), right placement strictly dominates beyond a finite threshold without assuming linear per-site cost.*

4. ***Generalized boundary mode:*** *without those growth-separation assumptions, strict dominance is not guaranteed (Proposition 8.24).*

Proof. Part (1) is Corollary 8.17. Part (2) is exactly `HardnessDistribution.centralized_higher_leverage`. Part (3) is Theorem 8.22. Part (4) is Proposition 8.24. ∎

Illustrative Instantiation (Type Systems). 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)

The table is schematic; the formal statement is Corollary 8.19.

Corollary 8.19 (Type-System Threshold)**.** *For the formal native-vs-manual architecture instance, native support has lower total realized cost for all*

$$n > H_{\text{baseline}}(P),$$

where $H_{\text{baseline}}(P)$ corresponds to the Lean identifier `intrinsicDOF(P)` in module `HardnessDistribution`. (Lean: `HardnessDistribution.native_dominates_manual`)

Proof. Immediate from `HardnessDistribution.native_dominates_manual`. ∎

8.5 Extension: Non-Additive Site-Cost Models

Definition 8.20 (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 8.21 (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 8.22 (Generalized Dominance by Growth Separation)**.** *Let*

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

For two architectures $S_{\text{right}}, S_{\text{wrong}}$, suppose there exists $B \in \mathbb{N}$ such that:

1. *$C_{S_{\text{right}}}(m) \leq B$ for all m (bounded right-side per-site accumulation),*
2. *$m \leq C_{S_{\text{wrong}}}(m)$ for all m (identity-lower-bounded wrong-side growth).*

Then for every

$$n > H_{central}(S_{\text{right}}) + B,$$

one has

$$H^{\text{gen}}_{total}(S_{\text{right}}, n) < H^{\text{gen}}_{total}(S_{\text{wrong}}, n).$$

(Lean: `HardnessDistribution.generalized_right_dominates_wrong_of_bounded_vs_identity_lower`)

Proof. This is exactly the mechanized theorem `HardnessDistribution.generalized_right_dominates_wrong_of_bounded_vs_identity_lower`. ■

Corollary 8.23 (Eventual Generalized Dominance)**.** *If condition (1) above holds and there exists N such that condition (2) holds for all $m \geq N$, then there exists N_0 such that for all $n \geq N_0$,*

$$H^{\text{gen}}_{total}(S_{\text{right}}, n) < H^{\text{gen}}_{total}(S_{\text{wrong}}, n).$$

(Lean: `HardnessDistribution.generalized_right_eventually_dominates_wrong`)

Proof. Immediate from `HardnessDistribution.generalized_right_eventually_dominates_wrong`. ■

Proposition 8.24 (Generalized Assumption Boundary)**.** *In the generalized model, strict right-vs-wrong dominance is not unconditional. There are explicit counterexamples:*

1. *If wrong-side growth lower bounds are dropped, right-side strict dominance can fail for all n.*
2. *If right-side boundedness is dropped, strict dominance can fail for all n even when wrong-side growth is linear.*

(Lean: `HardnessDistribution.generalized_dominance_can_fail_without_wrong_growth`, `HardnessDistribution.generalized_dominance_can_fail_without_right_boundedness`)

Proof. This is exactly the pair of mechanized boundary theorems listed above. ■

Theorem 8.25 (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: `HardnessDistribution.totalDOF_eventually_constant_iff_zero_distributed`)

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

Theorem 8.26 (Generalized Model: Saturation is Possible)**.** *There exists a generalized site-cost model with eventual saturation. In particular, for*

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

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

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

Corollary 8.27 (Positive Linear Slope Cannot Represent Saturation)**.** *No positive-slope linear per-site model can represent the saturating family above for all n. (Lean: `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 4 ("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.

9 Corollary: Complexity Redistribution Under Incomplete Models

The load-bearing fact in this section is not the set identity itself; it is the difficulty of shrinking the required set $R(P)$ in the first place. By Theorem 4.6 (and Theorem 4.7 for minimization), exact relevance identification is intractable in the worst case under succinct encoding. The identities below therefore quantify how unresolved relevance is redistributed between central and per-site work.

Definition 9.1. Let $R(P)$ be the required dimensions (those affecting Opt) and $A(M)$ the dimensions model M handles natively. The *expressive gap* is $\mathrm{Gap}(M,P) = R(P) \setminus A(M)$.

Definition 9.2 (Simplicity Tax)**.** The *simplicity tax* is the size of the expressive gap:

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

Theorem 9.3 (Redistribution Identity)**.** $|Gap(M,P)| + |R(P) \cap A(M)| = |R(P)|$. *The total cannot be reduced—only redistributed between "handled natively" and "handled externally." (Lean: `HardnessDistribution.gap_conservation_card`)*

Proof. In the finite-coordinate model this is the exact set-cardinality identity

$$|R \setminus A| + |R \cap A| = |R|,$$

formalized as `HardnessDistribution.gap_conservation_card`. ■

Remark 9.4 (Why this is nontrivial in context). The algebraic identity in Theorem 9.3 is elementary. Its force comes from upstream hardness: reducing $|R(P)|$ by exact relevance minimization is worst-case intractable under the succinct encoding, so redistribution is often the only tractable lever available.

Theorem 9.5 (Linear Growth)**.** *For n decision sites:*

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

(Lean: `HardnessDistribution.totalExternalWork_eq_n_mul_gapCard`)

Proof. This is by definition of per-site externalization and is mechanized as `HardnessDistribution.totalExternalWork_eq_n_mul_gapCard`. ■

Theorem 9.6 (Amortization)**.** *Let $H_{central}$ be the one-time cost of using a complete model. There exists*

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

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

Proof. For positive per-site tax, the threshold inequality

$$n > \left\lfloor \frac{H_{\text{central}}}{\text{SimplicityTax}} \right\rfloor \implies H_{\text{central}} < n \cdot \text{SimplicityTax}$$

is mechanized as `HardnessDistribution.complete_model_dominates_after_threshold`. ■

Corollary 9.7 (Gap Externalization)**.** *If $Gap(M, P) \neq \emptyset$, then external handling cost scales linearly with the number of decision sites. (Lean:* `HardnessDistribution.totalExternalWork_eq_n_mul_gapCard`*,* `HardnessDistribution.simplicityTax_grows`*)*

Proof. The exact linear form is `HardnessDistribution.totalExternalWork_eq_n_mul_gapCard`. When the gap is nonempty (positive tax), monotone growth with n is `HardnessDistribution.simplicityTax_grows`. ■

Corollary 9.8 (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:* `Sigma2PHardness.min_sufficient_set_iff_relevant_card`*)*

Proof. This is `Sigma2PHardness.min_sufficient_set_iff_relevant_card`. ■

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

10 Related Work

10.1 Computational Decision Theory

The complexity of decision-making has been studied extensively. Papadimitriou [13] 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 [4].

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 [5, 3]. Our contribution is stronger on two axes not provided by those works: (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.

10.2 Feature Selection

In machine learning, feature selection asks which input features are relevant for prediction. This is known to be NP-hard in general [5]. Our results show the decision-theoretic analog is coNP-complete for both checking and minimization.

10.3 Value of Information

The value of information (VOI) framework [8] 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.

10.4 Model Selection

Statistical model selection (AIC [2], BIC [16], cross-validation [18]) provides practical heuristics for choosing among models. Our results formalize the regime-level reason heuristic selection appears: without added structural assumptions, exact optimal model selection inherits worst-case intractability, so heuristic methods implement explicit weakened-guarantee policies for unresolved structure.

10.5 Certifying Outputs and Proof-Carrying Claims

Our integrity layer matches the certifying-algorithms pattern: algorithms emit candidate outputs together with certificates that can be checked quickly, separating *producing* claims from *verifying* claims [11]. In this paper, Definition 3.5 is exactly that soundness discipline.

At the systems level, this is the same architecture as proof-carrying code: a producer ships evidence and a consumer runs a small checker before accepting the claim [12]. Our competence definition adds the regime-specific coverage/resource requirement that certifying soundness alone does not provide.

The feasibility qualifier in Definition 3.6 also aligns with bounded-rationality normativity: what agents *should* do is constrained by what is computationally feasible under the declared resource model [1].

11 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 4.6)

- **MINIMUM-SUFFICIENT-SET** is coNP-complete (Theorem 4.7)
- **ANCHOR-SUFFICIENCY** is Σ_2^{P}-complete (Theorem 4.9)
- 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 5.1)
- Full intermediate query-access lower bounds are formalized in Boolean query submodels: $\Omega(2^n)$ worst-case queries for SUFFICIENCY-CHECK for 'Opt'-oracle, value-entry, and state-batch interfaces via indistinguishable yes/no pairs on the $I = \emptyset$ subproblem, with explicit subproblem-to-full transfer and seedwise randomized robustness (Propositions 5.3–5.7, 5.5, 5.6)
- Tractable subcases exist for explicit-state encoding, separable utility, and tree-structured utility with explicit local factors (Theorem 6.1)

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

Beyond classification, the paper contributes a formal claim-typing framework (Section 3): structural complexity is a property of the fixed decision relation, while representational hardness is regime-conditional access cost. This is why encoding-regime changes can move practical hardness without changing the underlying semantics.

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, ∃∀-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 ∃∀-SAT are explicit and machine-checked.
3. **Encoding-regime separation.** Under [E], SUFFICIENCY-CHECK is polynomial in $|S|$. Under [S+ETH], there exist succinct worst-case instances (with $k^* = n$) requiring $2^{\Omega(n)}$ time. Under [Q_bool], mechanized lower bounds of $\Omega(2^n)$ hold for 'Opt'-oracle, value-entry, and state-batch interfaces (via the $I = \emptyset$ subproblem), with finite-state and weighted-cost transfer closures.

The Complexity Redistribution Corollary

Section 9 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 $(k - j) \times n$.

The set identity is elementary; its operational content comes from composition with the hardness results on exact relevance minimization. This redistribution corollary is formalized in Lean 4 (`HardnessDistribution.lean`), proving:

- Redistribution identity: complexity burden cannot be eliminated by omission, only moved between native handling and external handling
- 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?
- **Sequential/stochastic bridge extension:** Characterize the exact frontier where adjacent sequential objectives reduce to the static class via Proposition 2.20, and where genuinely new complexity objects (e.g., horizon/sample/regret complexity) must replace the present coNP/Σ_2^P analysis.
- **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 Corollaries

The practical corollaries are regime-indexed and theorem-indexed:

- **[E] and structured regimes:** polynomial-time exact procedures exist (Theorem 6.1).
- **[Q_bool] query-access lower bounds:** worst-case full-problem query complexity is $\Omega(2^n)$ for ‘Opt‘-oracle, value-entry, and state-batch interfaces (Propositions 5.3–5.7), with subproblem-transfer and randomized-robustness closures.
- **[S+ETH] hard families:** exact minimization inherits exponential worst-case cost (Theorem 5.1 together with Theorem 4.6).
- **[S_bool] mechanized criterion:** minimization reduces to relevance-cardinality constraints (Corollary 8.6).
- **Redistribution consequences:** omitted native coverage externalizes work with explicit growth/amortization laws (Theorems 9.3–9.6).

Hence the design choice is typed: enforce a tractable regime, or adopt weakened guarantees with explicit verification boundaries. Equivalently, over-specification is a conditional attempted-competence-failure policy in this framework; once exact competence is available in the active regime (or no attempted exact competence was made), persistent over-specification is irrational for the same verified-cost objective. By Proposition 3.9, this is not a close call: in the integrity-preserving matrix, irrational cells outnumber rational cells (3 vs 1).

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 7741 lines across 42 files, with 350 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:** Simplicity Tax redistribution identities and hardness distribution results.
4. **Query-access lower-bound core:** Formalized Boolean query-model indistinguishability theorem for the full problem via the $I = \emptyset$ subproblem (`emptySufficiency_query_indistinguishable_pair`), plus obstruction-scale identities (`queryComplexityLowerBound`, `exponential_query_complexity`).

Complexity classifications (coNP-completeness, Σ_2^P-completeness) follow by conditional composition with standard results (e.g., TAUTOLOGY coNP-completeness and ∃∀-SAT Σ_2^P-completeness), represented explicitly as hypotheses in the conditional transfer theorems listed below. The Lean proofs verify the reduction constructions and the transfer closures under those hypotheses. The assumptions themselves are unpacked by an explicit ledger projection theorem (`ClaimClosure.standard_assumption_ledger_unpack`) so dependency tracking is machine-auditable.

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 9) 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)
```

Identifier note. Lean identifiers retain internal naming (`intrinsicDOF`, `simplicityTax_conservation`); in paper terminology these correspond to *baseline hardness* and the *redistribution lower-bound identity*, respectively.

A.5 Complete Claim Coverage Matrix

Paper handle	Status	Lean support	Notes
cor:exact-identifiability	Full	Sigma2PHardness.exactlyIdentifiesRelevant_iff_sufficient_and_subset_relevantFinset	Direct theorem mapping.
cor:gap-externalization	Full	HardnessDistribution.totalExternalWork_eq_n_mul_gapCard, HardnessDistribution.simplicityTax_grows	Composite from two mechanized theorems.
cor:gap-minimization-hard	Full	Sigma2PHardness.min_sufficient_set_iff_relevant_card	Direct theorem mapping.
cor:generalized-eventual-dominance	Full	HardnessDistribution.generalized_right_eventually_dominates_wrong	Direct theorem mapping.
cor:linear-positive-no-saturation	Full	HardnessDistribution.no_positive_slope_linear_represents_saturating	Direct theorem mapping.
cor:no-auto-minimize	Full (conditional)	ClaimClosure.no_auto_minimize_of_p_neq_conp	Mechanized conditional closure: non-collapse plus collapse-implication yields impossibility.
cor:overmodel-diagnostic-implication	Full	Sigma2PHardness.sufficient_iff_relevant_subset	Contrapositive of mechanized theorem.
cor:practice-bounded	Full	sufficiency_poly_bounded_actions	Direct theorem mapping.
cor:practice-diagnostic	Full	Sigma2PHardness.min_sufficient_set_iff_relevant_card	Direct theorem mapping.
cor:practice-structured	Full	sufficiency_poly_separable	Direct theorem mapping.
cor:practice-tree	Full	sufficiency_poly_tree_structured	Direct theorem mapping.
cor:practice-unstructured	Full	all_coords_relevant_of_not_tautology	Direct theorem mapping.
cor:right-wrong-hardness	Full	HardnessDistribution.right_dominates_wrong	Direct theorem mapping.
cor:type-system-threshold	Full	HardnessDistribution.native_dominates_manual	Direct theorem mapping.
prop:adq-ordering	Full	ClaimClosure.adq_ordering	Strict ordering theorem for shared nonempty behavior denominator.
prop:dominance-modes	Full	HardnessDistribution.right_dominates_wrong, HardnessDistribution.centralized_higher_leverage, HardnessDistribution.generalized_right_dominates_wrong_of_bounded_vs_identity_lower, HardnessDistribution.generalized_dominance_can_fail_without_wrong_growth, HardnessDistribution.generalized_dominance_can_fail_without_right_boundedness	Compositional index proposition.
prop:generalized-assumption-boundary	Full	HardnessDistribution.generalized_dominance_can_fail_without_wrong_growth, HardnessDistribution.generalized_dominance_can_fail_without_right_boundedness	Direct pair mapping.
prop:hardness-conservation	Full	HardnessDistribution.totalDOF_ge_intrinsic	Direct structural core theorem.
prop:hardness-efficiency-interpretation	Full	HardnessDistribution.hardnessEfficiency_eq_central_share	Direct definitional equivalence.
prop:integrity-competence-separation	Full	IntegrityCompetence.competence_implies_integrity, IntegrityCompetence.integrity_not_competent_of_nonempty_scope	Direct pair mapping.
prop:minimal-relevant-equiv	Full	DecisionProblem.minimalSufficient_iff_relevant, DecisionProblem.relevantSet_is_minimal	Product-space bridge: minimal sufficient sets coincide with the witness-defined relevance set.
prop:query-regime-obstruction	Full	emptySufficiency_query_indistinguishable_pair, constTrue_empty_sufficient, spike_empty_not_sufficient, queryComplexityLowerBound, exponential_query_complexity	Mechanized Boolean query-regime full lower-bound core (via $I = \emptyset$ indistinguishability) plus obstruction-scale identities.

prop:query-value-entry-lb	Full	emptySufficiency_valueEntry_indistinguishable_pair, valueEntryView_eq_if_hidden_untouched, touchedStates_card_le_query_card	Mechanized Boolean value-entry query lower bound (full problem via $I = \emptyset$ indistinguishability).
prop:query-subproblem-transfer	Full	ClaimClosure.subproblem_hardness_lifts_to_full	Generic mechanized closure from subproblem hardness to full-problem hardness under restriction maps.
prop:query-randomized-robustness	Full	indistinguishable_pair_forces_one_error, indistinguishable_pair_forces_one_error_per_seed, decode_error_sum_two_labels	Seedwise mechanized randomized-robustness closure for indistinguishable yes/no pairs.
prop:query-state-batch-lb	Full	emptySufficiency_stateBatch_indistinguishable_pair, stateBatchView_eq_if_hidden_untouched	Mechanized state-batch query lower bound via hidden-state indistinguishability.
prop:query-finite-state-generalization	Full	emptySufficiency_query_indistinguishable_pair_finite, spikeFinite_empty_not_sufficient	Finite-state generalization of the empty-subproblem obstruction core (cardinality ≥ 2).
prop:query-tightness-full-scan	Full	full_query_distinguishes_const_spike_finite	Mechanized adversary-family tightness witness under full-state scan.
prop:query-weighted-transfer	Full	weightedQueryCost_ge_min_mul_card, weightedQueryCost_ge_min_mul_of_card_lb	Weighted-query transfer: cardinality lower bounds lift to weighted-cost lower bounds.
prop:set-to-selector	Full	DecisionProblem.sufficient_implies_selectorSufficient	Set-valued sufficiency implies selector-level sufficiency for any deterministic selector.
prop:zero-epsilon-reduction	Full	DecisionProblem.epsOpt_zero_eq_opt, DecisionProblem.sufficient_iff_zeroEpsilonSufficient	Mechanized zero-ε reduction: exact sufficiency is exactly the $\varepsilon = 0$ case.
prop:selector-separation	Full	ClaimClosure.selectorSufficient_not_implies_setSufficient	Mechanized counterexample: selector-sufficiency does not imply set-valued sufficiency.
prop:attempted-competence-matrix	Full	IntegrityCompetence.overModelVerdict_rational_iff, IntegrityCompetence.admissible_matrix_counts, IntegrityCompetence.admissible_irrational_strictly_more_than_rational	Integrity-preserving matrix is mechanized: one rational admissible cell and three irrational admissible cells.
prop:bridge-transfer-scope	Full	ClaimClosure.one_step_bridge	Transfer rule follows from the one-step deterministic bridge equivalence.
prop:one-step-bridge	Full	ClaimClosure.one_step_bridge	One-step deterministic bridge is mechanized as an equivalence of sufficiency predicates.
prop:sufficiency-char	Full	ClaimClosure.sufficiency_iff_dq_ratio, ClaimClosure.sufficiency_iff_projectedOptCover_eq_opt	Finite-model quotient characterization mechanized in both ratio form and projected-cover form.
thm:amortization	Full	HardnessDistribution.complete_model_dominates_after_threshold	Direct theorem mapping.
thm:anchor-sigma2p	Full (conditional)	ClaimClosure.anchor_sigma2p_reduction_core, ClaimClosure.anchor_sigma2p_complete_conditional, anchor_sufficiency_sigma2p	Reduction core is mechanized; completeness lift is mechanized as a conditional transfer on standard source-class facts.
thm:centralization-dominance	Full	HardnessDistribution.centralization_dominance_bundle, HardnessDistribution.centralization_step_saves_n_minus_one	Direct pair mapping.
thm:config-reduction	Full	ConfigReduction.config_sufficiency_iff_behavior_preserving	Direct theorem mapping.
thm:cost-asymmetry-eth	Full (conditional)	ClaimClosure.cost_asymmetry_eth_conditional, HardnessDistribution.linear_lt_exponential_plus_constant_eventually	Asymptotic dominance and ETH-conditioned lift are both mechanized.

`thm:dichotomy`	Full (conditional)	`ClaimClosure.dichotomy_conditional`, `ClaimClosure.hard_family_all_coords_core`, `ClaimClosure.explicit_state_upper_core`	Upper-branch core and hard-family core are mechanized; ETH lower branch is mechanized as conditional transfer.
`thm:generalized-dominance`	Full	`HardnessDistribution.generalized_right_dominates_wrong_of_bounded_vs_identity_lower`	Direct theorem mapping.
`thm:generalized-saturation-possible`	Full	`HardnessDistribution.saturatingSiteCost_eventually_constant`, `HardnessDistribution.generalizedTotal_with_saturation_eventually_constant`	Direct construction mapping.
`thm:linear-saturation-iff-zero`	Full	`HardnessDistribution.totalDOF_eventually_constant_iff_zero_distributed`	Direct theorem mapping.
`thm:minsuff-collapse`	Full (conditional)	`ClaimClosure.minsuff_collapse_core`, `ClaimClosure.minsuff_collapse_to_conp_conditional`, `Sigma2PHardness.min_sufficient_set_iff_relevant_card`	Quantifier collapse core and coNP-reading lift are mechanized.
`thm:minsuff-conp`	Full (conditional)	`ClaimClosure.minsuff_collapse_core`, `ClaimClosure.minsuff_conp_complete_conditional`	coNP-completeness transfer is mechanized conditionally from the collapse core.
`thm:overmodel-diagnostic`	Full	`ClaimClosure.no_exact_identifier_implies_not_boundary_characterized`, `ClaimClosure.boundaryCharacterized_iff_exists_sufficient_subset`, `ConfigReduction.config_sufficiency_iff_behavior_preserving`	Contrapositive diagnostic step mechanized under explicit boundary-characterization definition.
`thm:sufficiency-conp`	Full (conditional)	`ClaimClosure.sufficiency_conp_reduction_core`, `ClaimClosure.sufficiency_conp_complete_conditional`, `tautology_iff_sufficient`	Reduction core and coNP-completeness transfer are mechanized conditionally.
`thm:tax-conservation`	Full	`HardnessDistribution.gap_conservation_card`	Direct theorem mapping.
`thm:tax-grows`	Full	`HardnessDistribution.totalExternalWork_eq_n_mul_gapCard`	Direct theorem mapping.
`thm:tractable`	Full (conditional)	`ClaimClosure.tractable_bounded_core`, `ClaimClosure.tractable_separable_core`, `ClaimClosure.tractable_tree_core`, `ClaimClosure.tractable_subcases_conditional`	All three tractable branch cores and assembly closure are mechanized.

A.6 Claims Not Fully Mechanized

Status: all theorem/proposition/corollary handles in this paper now have Lean backing. Entries marked **Full (conditional)** are explicitly mechanized transfer theorems that depend on standard external complexity facts (e.g., source-class completeness or ETH assumptions), with those dependencies represented as hypotheses in Lean.

A.7 Module Structure

- `Basic.lean` – Core definitions (DecisionProblem, sufficiency, optimality)
- `Sufficiency.lean` – Sufficiency checking algorithms and properties
- `Reduction.lean` – TAUTOLOGY reduction construction and correctness
- `Hardness/ConfigReduction.lean` – Sentinel-action configuration reduction theorem
- `Complexity.lean` – Polynomial-time reduction definitions using mathlib
- `HardnessDistribution.lean` – Simplicity Tax redistribution and amortization theorems
- `IntegrityCompetence.lean` – Solver integrity vs regime competence separation
- `ClaimClosure.lean` – Mechanized closure of paper-level bridge and diagnostic claims

- `Tractability/` – Bounded actions, separable utilities, tree structure

A.8 Verification

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

References

[1] Bounded rationality. Stanford Encyclopedia of Philosophy, 2024. https://plato.stanford.edu/entries/bounded-rationality/, accessed 2026-02-20.

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

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

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

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

[6] 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.

[7] 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.

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

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

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

[11] Ross M. McConnell, Kurt Mehlhorn, Stefan Näher, and Pascal Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011.

[12] George C. Necula. Proof-carrying code. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 106–119. ACM, 1997.

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

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

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

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

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

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