Zero-Incoherence Capacity of Interactive Encoding Systems: Achievability, Converse, and Side Information Bounds

Tristan Simas

***Abstract*—We introduce the zero-incoherence capacity for interactive multi-location encoding systems: the maximum encoding rate that guarantees exactly zero probability of disagreement among replicated encodings. Our main information-theoretic results are compact and self-contained: an exact capacity theorem ($C_0 = 1$), a tight side-information lower bound for resolution ($\geq \log_2 k$ bits for k-way incoherence), and a rate–complexity separation (modification cost $O(1)$ at capacity vs $\Omega(n)$ above).**

The paper frames encoding locations as terminals in a multi-terminal source-coding model. Derivation (automatic deterministic dependence) is interpreted as perfect correlation that reduces effective rate; only complete derivation (one independent source) achieves zero incoherence. We give concise achievability and converse proofs in IT style, formalize the confusability/incoherence graph connection, and present an explicit mutual-information argument for the side-information bound.

Theoretical contributions are supplemented by constructive instantiations (programming-language patterns and a software case study). For TIT submission we move detailed language evaluation, extended code examples, and the full Lean proof corpus to supplementary material; the main text contains brief instantiations only. Core theorems (capacity, realizability, bounds) are machine-checked in Lean 4; entropy arguments apply standard Fano-inequality techniques.

Index Terms—zero-error capacity, multi-terminal source coding, side information, mutual information, confusability graph

I. Introduction

A. Zero-Incoherence Capacity

We address this information-theoretic question: what encoding rate guarantees exactly zero probability of disagreement among replicated encodings in an interactive setting? An *encoding system* stores a fact F (value in $\mathcal{V}_F$) at locations $\{L_1, \ldots, L_n\}$. The system is *coherent* when all locations agree; otherwise it is *incoherent*. The central quantity is the **zero-incoherence capacity** C_0, the supremal encoding rate that forces incoherence probability to be exactly zero. Our main theorem is compact:

$$\boxed{C_0 = 1}$$

T. Simas is with McGill University, Montreal, QC, Canada (e-mail: tristan.simas@mail.mcgill.ca).

Manuscript received February 20, 2026.

This is a zero-error style result in the Shannon tradition [1]–[3], adapted to storage/encoding systems that are modified over time rather than one-shot channels.

Main results. Let DOF (Degrees of Freedom) denote the encoding rate: the number of independent locations that can hold distinct values simultaneously.

- **Achievability (Theorem II.37):** DOF $= 1$ achieves zero incoherence.
- **Converse (Theorem II.38):** DOF > 1 does not achieve zero incoherence.
- **Capacity (Theorem II.36):** $C_0 = 1$ exactly. Tight.
- **Side Information (Theorem II.47):** Resolution of k-way incoherence requires $\geq \log_2 k$ bits.

Theorem I.1 (Resolution Impossibility, informal)**.** *For any incoherent encoding system and any resolution procedure, there exists a value present in the system that disagrees with the resolution. Without* $\log_2 k$ *bits of side information (where* k *= DOF), no resolution is information-theoretically justified.*

This parallels zero-error decoding constraints [2], [3]: without sufficient side information, error-free reconstruction is impossible.

B. The Capacity Theorem

The zero-incoherence capacity follows the achievability/converse structure of Shannon's channel capacity theorem:

Encoding Rate	Zero Incoherence?	Interpretation
DOF $= 0$	N/A	Fact not encoded
DOF $= 1$	**Yes**	Capacity-achieving
DOF > 1	No	Above capacity

Comparison to Shannon capacity. Shannon's channel capacity C is the supremum of rates R achieving vanishing error probability: $\lim_{n\to\infty} P_e^{(n)} = 0$. Our zero-incoherence capacity is the supremum of rates achieving *exactly zero* incoherence probability—paralleling zero-error capacity [1], not ordinary capacity.

Connection to MDL. The capacity theorem generalizes Rissanen's Minimum Description Length principle [4], [5] to interactive systems. MDL optimizes description length for static data. We optimize encoding rate for modifiable data subject to coherence constraints. The result: exactly one rate

($R = 1$) achieves zero incoherence, making this a **forcing theorem**.

C. Applications Across Domains

The abstract encoding model applies wherever facts are stored redundantly:

- **Distributed databases:** Replica consistency under partition constraints [6]
- **Version control:** Merge resolution when branches diverge [7]
- **Configuration systems:** Multi-file settings with coherence requirements [8]
- **Software systems:** Class registries, type definitions, interface contracts [9]

In each domain, the question is identical: given multiple encoding locations, which is authoritative? Our theorems characterize when this question has a unique answer (DOF = 1) versus when it requires arbitrary external resolution (DOF > 1).

D. Connection to Classical Information Theory

Our results extend classical source coding theory to interactive multi-terminal systems.

1. Multi-terminal source coding. Slepian-Wolf [10] characterizes distributed encoding of correlated sources. We model encoding locations as terminals: derivation introduces *perfect correlation* (deterministic dependence), reducing effective rate. The capacity result shows that only complete correlation (all terminals derived from one source) guarantees coherence—partial correlation permits divergence. Section II-L develops this connection.

2. Zero-error capacity. Shannon [1], Körner [2], and Lovász [3] characterize zero-error communication. We characterize **zero-incoherence encoding**—a storage analog where "errors" are disagreements among locations. The achievability/converse structure (Theorems II.37, II.38) parallels zero-error capacity proofs.

3. Interactive information theory. The BIRS workshop [11] identified interactive IT as encoding/decoding with feedback and multi-round protocols. Our model is interactive: encodings are modified over time, and causal propagation (a realizability requirement) is analogous to channel feedback. Ma-Ishwar [12] showed interaction can reduce rate; we show derivation (a form of interaction) can reduce effective DOF.

4. Rate-complexity tradeoffs. Rate-distortion theory [13] trades rate R against distortion D. We trade encoding rate (DOF) against modification complexity M: DOF $= 1$ achieves $M = O(1)$; DOF > 1 requires $M = \Omega(n)$. The gap is unbounded (Theorem VI.7).

E. Encoder Realizability

A key question: what encoder properties are necessary and sufficient to achieve capacity ($C_0 = 1$)? We prove realizability requires two information-theoretic properties:

1) **Causal update propagation (feedback coupling):** Changes to the source must automatically trigger updates to derived locations. This is analogous to *channel coding with feedback* [13]—the encoder (source) and decoder (derived locations) are coupled causally. Without feedback, a temporal window exists where source and derived locations diverge (temporary incoherence).
2) **Provenance observability (decoder side information):** The system must support queries about derivation structure. This is the encoding-system analog of *Slepian-Wolf side information* [10]—the decoder has access to structural information enabling verification that all terminals are derived from the source.

Theorem I.2 (Encoder Realizability, informal)**.** *An encoding system achieves $C_0 = 1$ iff it provides both causal propagation and provenance observability. Neither alone suffices (Theorem IV.8).*

Connection to multi-version coding. Rashmi et al. [14] prove an "inevitable storage cost for consistency" in distributed storage. Our realizability theorem is analogous: systems lacking either encoder property *cannot* achieve capacity—the constraint is information-theoretic, not implementation-specific.

Instantiations. The encoder properties instantiate across domains: programming languages (definition-time hooks, introspection), distributed databases (triggers, system catalogs), configuration systems (dependency graphs, state queries). Section V provides a programming-language instantiation as a corollary; the core theorems are domain-independent.

F. Paper Organization

Core theorems (capacity, realizability, complexity bounds) are machine-checked in Lean 4 [15] (9,369 lines, 441 theorems, 0 `sorry` placeholders). The entropy and mutual-information arguments in Section VI-D apply standard information theory (Fano's inequality [13]); we state these results without separate formalization, as they follow directly from the classical source.

Section II—Encoding Model and Capacity. We define multi-location encoding systems, encoding rate (DOF), and coherence/incoherence. We introduce information-theoretic quantities (value entropy, redundancy, incoherence entropy). We prove the **zero-incoherence capacity theorem** ($C_0 = 1$) with explicit achievability/converse structure, and the **side information bound** ($\geq \log_2 k$ bits for k-way resolution). We formalize encoding-theoretic CAP/FLP.

Section III—Derivation and Optimal Rate. We characterize derivation as the mechanism achieving capacity: derived locations are perfectly correlated with their source, contributing zero effective rate.

Section IV—Encoder Realizability. We prove that achieving capacity requires causal propagation (feedback) and provenance observability (decoder side information). Both necessary; together sufficient. This is an iff characterization.

Section VI—Rate-Complexity Tradeoffs. We prove modification complexity is $O(1)$ at capacity vs. $\Omega(n)$ above capacity. The gap is unbounded.

Sections V, VII—Instantiations (Corollaries). Programming-language instantiation and worked example. These illustrate the abstract theory; core results are domain-independent.

G. Core Theorems

We establish five *core* theorems:

1. **Theorem II.36 (Zero-Incoherence Capacity):** $C_0 = 1$. The maximum encoding rate guaranteeing zero incoherence is exactly 1.
   Structure: Achievability (Theorem II.37) + Converse (Theorem II.38).
2. **Theorem II.47 (Side Information Bound):** Resolution of k-way incoherence requires $\geq \log_2 k$ bits of side information. At DOF $= 1$, zero bits suffice.
   Proof: The k alternatives have entropy $H(S) = \log_2 k$. Resolution requires mutual information $I(S;Y) \geq H(S)$.
3. **Theorem II.4 (Resolution Impossibility):** Without side information, no resolution procedure is information-theoretically justified.
   Proof: By incoherence, $k \geq 2$ values exist. Any selection leaves $k-1$ values disagreeing. No internal information distinguishes them.
4. **Theorem IV.9 (Encoder Realizability):** Achieving capacity requires encoder properties: (a) causal propagation (feedback), and (b) provenance observability (side information). Both necessary; together sufficient.
   Proof: Necessity by constructing above-capacity configurations when either is missing. Sufficiency by exhibiting capacity-achieving encoders.
5. **Theorem VI.7 (Rate-Complexity Tradeoff):** Modification complexity scales as $O(1)$ at capacity vs. $\Omega(n)$ above capacity. The gap is unbounded.
   Proof: At capacity, one source update suffices. Above capacity, n independent locations require n updates.

Uniqueness. $C_0 = 1$ is the **unique** capacity: DOF $= 0$ fails to encode; DOF > 1 exceeds capacity. Given zero-incoherence as a constraint, the rate is mathematically forced.

H. Scope

This work characterizes SSOT for *structural facts* (class existence, method signatures, type relationships) within *single-language* systems. The complexity analysis is asymptotic, applying to systems where n grows. External tooling can approximate SSOT behavior but operates outside language semantics.

Multi-language systems. When a system spans multiple languages (e.g., Python backend + TypeScript frontend + protobuf schemas), cross-language SSOT requires external code generation tools. The analysis in this paper characterizes single-language SSOT; multi-language SSOT is noted as future work (Section IX).

I. Contributions

This paper makes five information-theoretic contributions:

1. Zero-incoherence capacity (Section II-K):

- Definition of encoding rate (DOF) and incoherence
- **Theorem II.36:** $C_0 = 1$ (tight: achievability + converse)
- **Theorem II.33:** Redundancy $\rho > 0$ iff incoherence reachable

2. Side information bounds (Section II-L):

- **Theorem II.47:** k-way resolution requires $\geq \log_2 k$ bits
- **Corollary II.48:** DOF $= 1$ requires zero side information
- Multi-terminal interpretation: derivation as perfect correlation

3. Encoder realizability (Section IV):

- **Theorem IV.9:** Capacity achieved iff causal propagation AND provenance observability
- **Theorem IV.8:** Requirements are independent
- Connection to feedback channels and Slepian-Wolf side information

4. Rate-complexity tradeoffs (Section VI):

- **Theorem VI.2:** $O(1)$ at capacity
- **Theorem VI.3:** $\Omega(n)$ above capacity
- **Theorem VI.7:** Gap unbounded

5. Encoding-theoretic CAP/FLP (Section II-I):

- **Theorem II.27:** CAP as encoding impossibility
- **Theorem II.29:** FLP as resolution impossibility

Instantiations (corollaries). Sections V and VII instantiate the realizability theorem for programming languages and provide a worked example. These are illustrative corollaries; the core information-theoretic results are self-contained in Sections II–VI.

II. Encoding Systems and Coherence

We formalize encoding systems with modification constraints and prove fundamental limits on coherence. The core results apply universally to any domain where facts are encoded at multiple locations and modifications must preserve correctness. Software systems are one instantiation; distributed databases, configuration management, and version control are others.

A. Model assumptions and notation

For rigour we state the modeling assumptions used throughout the paper. These are intentionally minimal and are made explicit so reviewers can judge applicability.

1) **Fact value model (A1):** Each fact F takes values in a finite set $\mathcal{V}_F$; $H(F) = \log_2 |\mathcal{V}_F|$ denotes its entropy under a chosen prior. When we write "zero probability of incoherence" we mean $\Pr(\text{incoherent}) = 0$ under the model for edits and randomness specified below.
2) **Update model (A2):** Modifications occur as discrete events (rounds). An edit δ_F changes the source value for F; derived locations update deterministically when causal propagation is present. We do not require a blocklength parameter; our asymptotic statements are in the number of encoding locations n or in ensemble scaling of the codebase.
3) **Adversary / randomness (A3):** When randomness is needed we assume a benign stochastic model for edits (e.g., uniformly sampled new values for illustration). For impossibility/converse statements we make no cooperative assumptions of the editor: adversarial sequences of edits that produce incoherence are allowed when DOF > 1.

4) **Side-information channel (A4):** Derived locations and runtime queries collectively form side information S available to any verifier. We model S as a random variable correlated with F and use mutual information $I(F;S)$ to quantify its information content.

References to these assumptions use labels (A1)–(A4). Full formal versions of the operational model are given in the mechanized supplement (Supplement A).

B. The Encoding Model

We begin with the abstract encoding model: locations, values, and coherence constraints.

Definition II.1 (Encoding System)**.** An *encoding system* for a fact F is a collection of locations $\{L_1, \ldots, L_n\}$, each capable of holding a value for F.

Definition II.2 (Coherence)**.** An encoding system is *coherent* iff all locations hold the same value:

$$\forall i, j : \text{value}(L_i) = \text{value}(L_j)$$

Definition II.3 (Incoherence)**.** An encoding system is *incoherent* iff some locations disagree:

$$\exists i, j : \text{value}(L_i) \neq \text{value}(L_j)$$

The Resolution Problem. When an encoding system is incoherent, no resolution procedure is information-theoretically justified. Any oracle selecting a value leaves another value disagreeing, creating an unresolvable ambiguity.

Theorem II.4 (Oracle Arbitrariness)**.** *For any incoherent encoding system S and any oracle O that resolves S to a value $v \in S$, there exists a value $v' \in S$ such that $v' \neq v$.*

Proof. By incoherence, $\exists v_1, v_2 \in S : v_1 \neq v_2$. Either O picks v_1 (then v_2 disagrees) or O doesn't pick v_1 (then v_1 disagrees). ■

Interpretation. This theorem parallels zero-error capacity constraints in communication theory. Just as insufficient side information makes error-free decoding impossible, incoherence makes truth-preserving resolution impossible. The encoding system does not contain sufficient information to determine which value is correct. Any resolution requires external information not present in the encodings themselves.

Definition II.5 (Degrees of Freedom)**.** The *degrees of freedom* (DOF) of an encoding system is the number of locations that can be modified independently.

Theorem II.6 (DOF = 1 Guarantees Coherence)**.** *If DOF = 1, then the encoding system is coherent in all reachable states.*

Proof. With DOF = 1, exactly one location is independent. All other locations are derived (automatically updated when the source changes). Derived locations cannot diverge from their source. Therefore, all locations hold the value determined by the single independent source. Disagreement is impossible. ■

Theorem II.7 (DOF $>$ 1 Permits Incoherence)**.** *If DOF $>$ 1, then incoherent states are reachable.*

Proof. With DOF $>$ 1, at least two locations are independent. Independent locations can be modified separately. A sequence of edits can set $L_1 = v_1$ and $L_2 = v_2$ where $v_1 \neq v_2$. This is an incoherent state. ■

Corollary II.8 (Coherence Forces DOF = 1)**.** *If coherence must be guaranteed (no incoherent states reachable), then DOF = 1 is necessary and sufficient.*

This is the information-theoretic foundation of optimal encoding under coherence constraints.

Connection to Minimum Description Length. The DOF = 1 optimum directly generalizes Rissanen's MDL principle [4]. MDL states that the optimal representation minimizes total description length: |model| + |data given model|. In encoding systems:

- **DOF = 1:** The single source is the minimal model. All derived locations are "data given model" with zero additional description length (fully determined by the source). Total encoding rate is minimized.
- **DOF $>$ 1:** Redundant independent locations require explicit synchronization. Each additional independent location adds description length with no reduction in uncertainty—pure overhead serving no encoding purpose.

Grünwald [5] proves that MDL-optimal representations are unique under mild conditions. Theorem II.24 establishes the analogous uniqueness for encoding systems under modification constraints: DOF = 1 is the unique coherence-guaranteeing rate.

Generative Complexity. Heering [16] formalized this for computational systems: the *generative complexity* of a program family is the length of the shortest generator. DOF = 1 systems achieve minimal generative complexity—the single source is the generator, derived locations are generated instances. This connects our framework to Kolmogorov complexity while remaining constructive (we provide the generator, not just prove existence).

The following sections show how computational systems instantiate this encoding model.

C. Computational Realizations

The abstract encoding model (Definitions II.1–II.5) applies to any system where:

1) Facts are encoded at multiple locations
2) Locations can be modified
3) Correctness requires coherence across modifications

Domains satisfying these constraints:

- **Software codebases:** Type definitions, registries, configurations
- **Distributed databases:** Replica consistency under updates
- **Configuration systems:** Multi-file settings (e.g., infrastructure-as-code)
- **Version control:** Merge resolution under concurrent modifications

We focus on *computational realizations*—systems where locations are syntactic constructs manipulated by tools or

humans. Software codebases are the primary example, but the encoding model is not software-specific. This subsection is illustrative; the core information-theoretic results do not depend on any particular computational domain.

Definition II.9 (Codebase (Software Realization))**.** A *codebase* C is a finite collection of source files, each containing syntactic constructs (classes, functions, statements, expressions). This is the canonical computational encoding system.

Definition II.10 (Location)**.** A *location* $L \in C$ is a syntactically identifiable region: a class definition, function body, configuration value, type annotation, database field, or configuration entry.

Definition II.11 (Modification Space)**.** For encoding system C, the *modification space* $E(C)$ is the set of all valid modifications. Each edit $\delta \in E(C)$ transforms C into $C' = \delta(C)$.

The modification space is large (exponential in system size). But we focus on modifications that *change a specific fact*.

D. Facts: Atomic Units of Specification

Definition II.12 (Fact)**.** A *fact* F is an atomic unit of program specification: a single piece of knowledge that can be independently modified. Facts are the indivisible units of meaning in a specification.

The granularity of facts is determined by the specification, not the implementation. If two pieces of information must always change together, they constitute a single fact. If they can change independently, they are separate facts.

Examples of facts:

Fact	Description
F_1: "threshold = 0.5"	A configuration value
F_2: "`PNGLoader` handles `.png`"	A type-to-handler mapping
F_3: "`validate()` returns `bool`"	A method signature
F_4: "`Detector` is a subclass of `Processor`"	An inheritance relationship
F_5: "`Config` has field `name: str`"	A dataclass field

Definition II.13 (Structural Fact)**.** A fact F is *structural* with respect to encoding system C iff the locations encoding F are fixed at definition time:

$$\text{structural}(F, C) \iff \forall L : \text{encodes}(L, F) \to L \in \text{DefinitionSyntax}(C)$$

where $\text{DefinitionSyntax}(C)$ comprises declarative constructs that cannot change post-definition without recreation.

Examples across domains:

- **Software:** Class declarations, method signatures, inheritance clauses, attribute definitions
- **Databases:** Schema definitions, table structures, foreign key constraints
- **Configuration:** Infrastructure topology, service dependencies
- **Version control:** Branch structure, merge policies

Key property: Structural facts are fixed at *definition time*. Once defined, their structure cannot change without recreation. This is why structural coherence requires definition-time computation: the encoding locations are only mutable during creation.

Non-structural facts (runtime values, mutable state) have encoding locations modifiable post-definition. Achieving DOF = 1 for non-structural facts requires different mechanisms (reactive bindings, event systems) and is outside this paper's scope. We focus on structural facts because they demonstrate the impossibility results most clearly.

E. Encoding: The Correctness Relationship

Definition II.14 (Encodes)**.** Location L *encodes* fact F, written $\text{encodes}(L, F)$, iff correctness requires updating L when F changes.

Formally:

$$\text{encodes}(L, F) \iff \forall \delta_F : \neg\text{updated}(L, \delta_F) \to \text{incorrect}(\delta_F(C))$$

where δ_F is an edit targeting fact F.

Key insight: This definition is **forced** by correctness, not chosen. We do not decide what encodes what. Correctness requirements determine it. If failing to update location L when fact F changes produces an incorrect program, then L encodes F. This is an objective, observable property.

Example II.15 (Encoding in Practice)**.** Consider a type registry:

```
# Location L1: Class definition
class PNGLoader(ImageLoader):
    format = "png"

# Location L2: Registry entry
LOADERS = {"png": PNGLoader, "jpg": JPGLoader}

# Location L3: Documentation
# Supported formats: png, jpg
```

The fact F = "`PNGLoader` handles `png`" is encoded at:

- L_1: The class definition (primary encoding)
- L_2: The registry dictionary (secondary encoding)
- L_3: The documentation comment (tertiary encoding)

If F changes (e.g., to "`PNGLoader` handles `png` and `apng`"), all three locations must be updated for correctness. The program is incorrect if L_2 still says `{"png": PNGLoader}` when the class now handles both formats.

F. Modification Complexity

Definition II.16 (Modification Complexity)**.**

$$M(C, \delta_F) = |\{L \in C : \text{encodes}(L, F)\}|$$

The number of locations that must be updated when fact F changes.

Modification complexity is the central metric of this paper. It measures the *cost* of changing a fact. A codebase with $M(C, \delta_F) = 47$ requires 47 edits to correctly implement a change to fact F. A codebase with $M(C, \delta_F) = 1$ requires only 1 edit.

Theorem II.17 (Correctness Forcing)**.** *$M(C, \delta_F)$ is the **minimum** number of edits required for correctness. Fewer edits imply an incorrect program.*

Proof. Suppose $M(C, \delta_F) = k$, meaning k locations encode F. By Definition II.14, each encoding location must be updated when F changes. If only $j < k$ locations are updated, then $k - j$ locations still reflect the old value of F. These locations create inconsistencies:

1) The specification says F has value v' (new)
2) Locations $L_1, \ldots, L_j$ reflect v'
3) Locations $L_{j+1}, \ldots, L_k$ reflect v (old)

By Definition II.14, the program is incorrect. Therefore, all k locations must be updated, and k is the minimum. ■

G. Independence and Degrees of Freedom

Not all encoding locations are created equal. Some are *derived* from others.

Definition II.18 (Independent Locations)**.** Locations L_1, L_2 are *independent* for fact F iff they can diverge. Updating L_1 does not automatically update L_2, and vice versa.

Formally: L_1 and L_2 are independent iff there exists a sequence of edits that makes L_1 and L_2 encode different values for F.

Definition II.19 (Derived Location)**.** Location L_{derived} is *derived from* L_{source} iff updating L_{source} automatically updates L_{derived}. Derived locations are not independent of their sources.

Example II.20 (Independent vs. Derived)**.** Consider two architectures for the type registry:

Architecture A (independent locations):

```
# L1: Class definition
class PNGLoader(ImageLoader): ...

# L2: Manual registry (independent of L1)
LOADERS = {"png": PNGLoader}
```

Here L_1 and L_2 are independent. A developer can change L_1 without updating L_2, causing inconsistency.

Architecture B (derived location):

```
# L1: Class definition with registration
class PNGLoader(ImageLoader):
    format = "png"

# L2: Derived registry (computed from L1)
LOADERS = {cls.format: cls for cls in
    ImageLoader.__subclasses__()}
```

Here L_2 is derived from L_1. Updating the class definition automatically updates the registry. They cannot diverge.

Definition II.21 (Degrees of Freedom)**.**

$$\text{DOF}(C, F) = |\{L \in C : \text{encodes}(L, F) \wedge \text{independent}(L)\}|$$

The number of *independent* locations encoding fact F.

DOF is the key metric. Modification complexity M counts all encoding locations. DOF counts only the independent ones. If all but one encoding location is derived, DOF = 1 even though M can be large.

Theorem II.22 (DOF = Incoherence Potential)**.** *$DOF(C, F) = k$ implies k different values for F can coexist in C simultaneously. With $k > 1$, incoherent states are reachable.*

Proof. Each independent location can hold a different value. By Definition II.18, no constraint forces agreement between independent locations. Therefore, k independent locations can hold k distinct values. This is an instance of Theorem II.7 applied to software. ■

Corollary II.23 (DOF $>$ 1 Implies Incoherence Risk)**.** *$DOF(C, F) > 1$ implies incoherent states are reachable. The codebase can enter a state where different locations encode different values for the same fact.*

H. The DOF Lattice

DOF values form a lattice with distinct information-theoretic meanings:

DOF	Encoding Status
0	Fact F is not encoded (no representation)
1	Coherence guaranteed (optimal rate under coherence constraint)
$k > 1$	Incoherence possible (redundant independent encodings)

Theorem II.24 (DOF = 1 is Uniquely Coherent)**.** *For any fact F that must be encoded, $DOF(C, F) = 1$ is the unique value guaranteeing coherence:*

1) *DOF = 0: Fact is not represented*
2) *DOF = 1: Coherence guaranteed (by Theorem II.6)*
3) *DOF $>$ 1: Incoherence reachable (by Theorem II.7)*

This is formalized as the Zero-Incoherence Capacity theorem (Theorem II.36) in Section II-K.

I. Encoding-Theoretic CAP and FLP

We now formalize CAP and FLP inside the encoding model.

Definition II.25 (Local Availability)**.** An encoding system for fact F is *locally available* iff for every encoding location L

of F and every value v, there exists a valid edit $\delta \in E(C)$ such that updated(L, δ) and for every other encoding location L', $\neg$updated(L', δ). Informally: each encoding location can be updated without coordinating with others.

Definition II.26 (Partition Tolerance)**.** An encoding system for fact F is *partition-tolerant* iff F is encoded at two or more locations:

$$|\{L \in C : \text{encodes}(L, F)\}| \geq 2.$$

This is the minimal formal notion of "replication" in our model; without it, partitions are vacuous.

Theorem II.27 (CAP in the Encoding Model)**.** *No encoding system can simultaneously guarantee coherence (Definition II.2), local availability (Definition II.25), and partition tolerance (Definition II.26) for the same fact F.*

Proof. Partition tolerance gives at least two encoding locations. Local availability allows each to be updated without updating any other encoding location, so by Definition II.18 there exist two independent locations and thus DOF$(C, F) > 1$. By Theorem II.7, incoherent states are reachable, contradicting coherence. ■

Definition II.28 (Resolution Procedure)**.** A *resolution procedure* is a deterministic function R that maps an encoding system state to a value present in that state.

Theorem II.29 (Static FLP in the Encoding Model)**.** *For any incoherent encoding system state and any resolution procedure R, the returned value is arbitrary relative to the other values present; no deterministic R can be justified by internal information alone.*

Proof. Immediate from Theorem II.4: in an incoherent state, at least two distinct values are present, and any choice leaves another value disagreeing. ■

These theorems are the encoding-theoretic counterparts of CAP [6], [17] and FLP [18]: CAP corresponds to the impossibility of coherence when replicated encodings remain independently updatable; FLP corresponds to the impossibility of truth-preserving resolution in an incoherent state without side information.

J. Information-Theoretic Quantities

Before establishing the capacity theorem, we define the information-theoretic quantities that characterize encoding systems.

Definition II.30 (Encoding Rate)**.** The *encoding rate* of system C for fact F is $R(C, F) = \text{DOF}(C, F)$, the number of independent encoding locations. This counts locations that can hold distinct values simultaneously.

Definition II.31 (Value Entropy)**.** For a fact F with value space $\mathcal{V}_F$, the *value entropy* is:

$$H(F) = \log_2 |\mathcal{V}_F|$$

This is the information content of specifying F's value.

Definition II.32 (Encoding Redundancy)**.** The *encoding redundancy* of system C for fact F is:

$$\rho(C, F) = \text{DOF}(C, F) - 1$$

Redundancy measures excess independent encodings beyond the minimum needed to represent F.

Theorem II.33 (Redundancy-Incoherence Equivalence)**.** *Encoding redundancy $\rho > 0$ is necessary and sufficient for incoherence reachability:*

$$\rho(C, F) > 0 \iff \textit{incoherent states reachable}$$

Proof. ($\Rightarrow$) If $\rho > 0$, then DOF > 1. By Theorem II.7, incoherent states are reachable.

($\Leftarrow$) If incoherent states are reachable, then by contrapositive of Theorem II.6, DOF $\neq 1$. Since the fact is encoded, DOF ≥ 1, so DOF > 1, hence $\rho > 0$. ■

Definition II.34 (Incoherence Entropy)**.** For an incoherent state with k independent locations holding distinct values, the *incoherence entropy* is:

$$H_{\text{inc}} = \log_2 k$$

This quantifies the uncertainty about which value is "correct."

K. Zero-Incoherence Capacity Theorem

We now establish the central capacity result. This extends zero-error capacity theory [2], [3] to interactive encoding systems with modification constraints.

Background: Zero-Error Capacity. Shannon [1] introduced zero-error capacity: the maximum rate at which information can be transmitted with *zero* probability of error (not vanishing, but exactly zero). Körner [2] and Lovász [3] characterized this via graph entropy and confusability graphs. Our zero-incoherence capacity is analogous: the maximum encoding rate guaranteeing *zero* probability of incoherence.

Definition II.35 (Zero-Incoherence Capacity)**.** The *zero-incoherence capacity* of an encoding system is:

$$C_0 = \sup\{R : \text{encoding rate } R \Rightarrow \text{incoherence probability} = 0\}$$

where "incoherence probability = 0" means no incoherent state is reachable under any modification sequence.

Theorem II.36 (Zero-Incoherence Capacity)**.** *The zero-incoherence capacity of any encoding system under independent modification is exactly 1:*

$$C_0 = 1$$

We prove this via the standard achievability/converse structure. (These formalize Theorems II.6 and II.7 in IT capacity-proof style.)

Theorem II.37 (Achievability)**.** *Encoding rate DOF $= 1$ achieves zero incoherence. Therefore $C_0 \geq 1$.*

Proof. Let DOF$(C, F) = 1$. By Definition II.21, exactly one location L_s is independent; all other locations $\{L_i\}$ are derived from L_s.

By Definition II.19, derived locations satisfy: update$(L_s) \Rightarrow$ automatically_updated(L_i). Therefore, for any reachable state:

$$\forall i : \text{value}(L_i) = f_i(\text{value}(L_s))$$

where f_i is the derivation function. All values are determined by L_s. Disagreement requires two locations with different determining sources, but only L_s exists. Therefore, no incoherent state is reachable. ■

Theorem II.38 (Converse)**.** *Encoding rate DOF > 1 does not achieve zero incoherence. Therefore $C_0 < R$ for all $R > 1$.*

Proof. Let DOF$(C, F) = k > 1$. By Definition II.18, there exist locations L_1, L_2 that can be modified independently.

Construction of incoherent state: Consider the modification sequence:

1) δ_1: Set $L_1 = v_1$ for some $v_1 \in \mathcal{V}_F$
2) δ_2: Set $L_2 = v_2$ for some $v_2 \neq v_1$

Both modifications are valid (each location accepts values from $\mathcal{V}_F$). By independence, δ_2 does not affect L_1. The resulting state has:

$$\text{value}(L_1) = v_1 \neq v_2 = \text{value}(L_2)$$

By Definition II.3, this state is incoherent. Since it is reachable, zero incoherence is not achieved.

Fano-style argument. We can also prove the converse via an entropy argument analogous to Fano's inequality. Let X_i denote the value at location L_i. With DOF $= k > 1$, the k independent locations have joint entropy $H(X_1, \ldots, X_k) = k \cdot H(F)$ (each independently samples from $\mathcal{V}_F$). For coherence, we require $X_1 = X_2 = \cdots = X_k$, which constrains the joint distribution to the diagonal $\{(v, v, \ldots, v) : v \in \mathcal{V}_F\}$—a set of measure zero in the product space when $k > 1$. Thus the probability of coherence under independent modifications is zero, and the probability of incoherence is one. ■

Proof of Theorem II.36. By Theorem II.37, $C_0 \geq 1$.

By Theorem II.38, $C_0 < R$ for all $R > 1$.

Therefore $C_0 = 1$ exactly. The bound is tight: achieved at DOF $= 1$, not achieved at any DOF > 1. ■

Comparison to Shannon capacity. The structure parallels Shannon's noisy channel coding theorem [19]:

	Shannon (Channel)	**This Work (Encoding)**
Rate	Bits per channel use	Independent encoding locations
Constraint	Error probability $\to 0$	Incoherence probability $= 0$
Achievability	$R < C$ achieves	DOF $= 1$ achieves
Converse	$R > C$ fails	DOF > 1 fails
Capacity	$C = \max I(X; Y)$	$C_0 = 1$

The key difference: Shannon capacity allows vanishing error; zero-incoherence capacity requires *exactly* zero, paralleling zero-error capacity.

Corollary II.39 (Capacity-Achieving Rate is Unique)**.** *DOF $= 1$ is the unique capacity-achieving encoding rate. No alternative achieves zero incoherence at higher rate.*

Corollary II.40 (Redundancy Above Capacity)**.** *Any encoding with $\rho > 0$ (redundancy) operates above capacity and cannot guarantee coherence.*

1) Rate-Incoherence Function: Analogous to the rate-distortion function $R(D)$ in lossy source coding, we define the *rate-incoherence function*:

Definition II.41 (Rate-Incoherence Function)**.** The rate-incoherence function is:

$$R(\epsilon) = \inf\{R : \exists \text{ encoding achieving } P(\text{incoherence}) \leq \epsilon\}$$

where $\epsilon \in [0, 1]$ is the tolerable incoherence probability.

Theorem II.42 (Rate-Incoherence is a Step Function)**.** *The rate-incoherence function is discrete (all-or-nothing at $\epsilon = 0$):*

$$R(\epsilon) = \begin{cases} 1 & \text{if } \epsilon = 0 \\ \infty & \text{if } \epsilon > 0 \end{cases}$$

Proof. For $\epsilon = 0$ (zero incoherence tolerated): By Theorem II.36, the minimum rate achieving $P(\text{incoherence}) = 0$ is exactly DOF $= 1$. Thus $R(0) = 1$.

For $\epsilon > 0$ (some incoherence tolerated): Any DOF ≥ 1 is achievable—there is no upper bound on rate when incoherence is permitted. Thus $R(\epsilon) = \infty$ (no finite constraint). ■

Interpretation. Unlike rate-distortion, which exhibits smooth tradeoffs, the rate-incoherence function is discontinuous. There is no "graceful degradation": either you operate at capacity (DOF $= 1$, zero incoherence) or you have no coherence guarantee whatsoever. This discontinuity reflects the qualitative nature of correctness—a system is either coherent or it is not; there is no intermediate state.

2) Design Necessity: The capacity theorem yields a prescriptive result for system design:

Theorem II.43 (Design Necessity)**.** *If an encoding system requires zero incoherence for correctness, then the encoding rate must satisfy DOF ≤ 1. This is a necessary condition; no design choice can circumvent it.*

Proof. Suppose a system requires $P(\text{incoherence}) = 0$ and has DOF > 1. By Theorem II.38, incoherent states are reachable. This contradicts the requirement. Therefore DOF ≤ 1 is necessary.

The bound is not an implementation artifact but an information-theoretic limit. No clever engineering, additional tooling, or process discipline changes the fundamental constraint: with DOF > 1, independent locations can diverge. ■

Corollary II.44 (Architectural Forcing)**.** *For any fact F requiring coherence guarantees, system architecture must either:*

1) *Achieve DOF$(F) = 1$ through derivation mechanisms, or*
2) *Accept that coherence cannot be guaranteed and implement resolution protocols with $\geq \log_2(\text{DOF})$ bits of side information.*

There is no third option.

3) Coding Interpretation of Derivation: The capacity theorem admits a coding-theoretic interpretation: *derivation is the capacity-achieving code*.

Codebook analogy. In channel coding, a codebook maps messages to codewords. The capacity-achieving code is the one that achieves the maximum rate with vanishing error. In encoding systems:

- The "message" is the fact F's value
- The "codeword" is the encoding across all locations
- The "code" is the derivation structure—how locations depend on each other

A DOF $= 1$ derivation structure is a *capacity-achieving code*: it encodes F at exactly one independent location (rate 1) with zero incoherence (zero "error"). Derived locations are redundant symbols determined by the source—they add reliability (the value is readable at multiple locations) without adding rate (no independent information).

Redundancy without rate. In classical coding, redundancy increases reliability at the cost of rate. In DOF-1 encoding, derived locations provide redundancy (multiple copies of F's value) without rate cost—they are deterministic functions of the source. This is the information-theoretic sense in which derivation is "free": derived encodings do not consume encoding rate.

4) Connection to Confusability Graphs: Körner [2] and Lovász [3] characterized zero-error capacity via *confusability graphs*: vertices are channel inputs, edges connect inputs that can produce the same output. Zero-error capacity is determined by the graph's independence number.

We can define an analogous *incoherence graph* for encoding systems:

Definition II.45 (Incoherence Graph)**.** For encoding system C and fact F, the incoherence graph $G = (V, E)$ has:

- Vertices $V =$ independent encoding locations
- Edge $(L_i, L_j) \in E$ iff L_i and L_j can hold different values (are truly independent)

Theorem II.46 (Incoherence Graph Characterization)**.** *Zero incoherence is achievable iff the incoherence graph has at most one vertex:* $|V| \leq 1$.

Proof. If $|V| = 0$, no locations encode F. If $|V| = 1$, DOF $=$ 1 and zero incoherence is achieved by Theorem II.37. If $|V| \geq 2$, DOF ≥ 2 and incoherence is reachable by Theorem II.38. ■

This connects our capacity theorem to the Körner-Lovász theory: zero-error/zero-incoherence capacity is determined by a graph structure, and the capacity-achieving configuration corresponds to a degenerate graph (single vertex).

L. Side Information for Resolution

When an encoding system is incoherent, resolution requires external side information. We establish a tight bound on the required side information, connecting to Slepian-Wolf distributed source coding.

1) The Multi-Terminal View: We can view an encoding system as a *multi-terminal source coding* problem [10], [13]:

- Each encoding location is a *terminal* that can transmit (store) a value
- The fact F is the *source* to be encoded
- *Derivation* introduces correlation: a derived terminal's output is a deterministic function of its source terminal
- The *decoder* (any observer querying the system) must reconstruct F's value

In Slepian-Wolf coding, correlated sources (X, Y) can be encoded at rates (R_X, R_Y) satisfying $R_X \geq H(X|Y)$, $R_Y \geq H(Y|X)$, $R_X + R_Y \geq H(X, Y)$. With perfect correlation ($Y = f(X)$), we have $H(X|Y) = 0$—the correlated source needs zero rate.

Derivation as perfect correlation. In our model, derivation creates perfect correlation: if L_d is derived from L_s, then value$(L_d) = f(\text{value}(L_s))$ deterministically. The "rate" of L_d is effectively zero—it contributes no independent information. This is why derived locations do not contribute to DOF.

Independence as zero correlation. Independent locations have no correlation. Each can hold any value in $\mathcal{V}_F$. The total "rate" is DOF $\cdot H(F)$, but with DOF > 1, the locations can encode *different* values—incoherence.

2) Side Information Bound:

Theorem II.47 (Side Information Requirement)**.** *Given an incoherent encoding system with k independent locations holding distinct values $\{v_1, \ldots, v_k\}$, resolving to the correct value requires at least $\log_2 k$ bits of side information.*

Proof. The k independent locations partition the value space into k equally plausible alternatives. By Theorem II.4, no internal information distinguishes them—each is an equally valid "source."

Let $S \in \{1, \ldots, k\}$ denote the index of the correct source. The decoder must determine S to resolve correctly. Since S is uniformly distributed over k alternatives (by symmetry of the incoherent state):

$$H(S) = \log_2 k$$

Any resolution procedure is a function $R : \text{State} \to \mathcal{V}_F$. Without side information Y:

$$H(S|R(\text{State})) = H(S) = \log_2 k$$

because R can be computed from the state, which contains no information about which source is correct.

With side information Y that identifies the correct source:

$$H(S|Y) = 0 \Rightarrow I(S; Y) = H(S) = \log_2 k$$

Therefore, side information must provide at least $\log_2 k$ bits of mutual information with S. ■

Corollary II.48 (DOF = 1 Requires Zero Side Information)**.** *With DOF = 1, resolution requires 0 bits of side information.*

Proof. With $k = 1$, there is one independent location. $H(S) = \log_2 1 = 0$. No uncertainty exists about the authoritative source. ■

Corollary II.49 (Side Information Scales with Redundancy)**.** *The required side information equals* $\log_2(1+\rho)$ *where* ρ *is encoding redundancy:*

$$\textit{Side information required} = \log_2(\textit{DOF}) = \log_2(1+\rho)$$

Connection to Slepian-Wolf. In Slepian-Wolf coding, the decoder has side information Y and decodes X at rate $H(X|Y)$. Our setting inverts this: the decoder needs side information S (source identity) to "decode" (resolve) which value is correct. The required side information is exactly the entropy of the source-selection variable.

Connection to zero-error decoding. In zero-error channel coding, the decoder must identify the transmitted message with certainty. Without sufficient side information (channel structure), this is impossible. Similarly, without $\log_2 k$ bits identifying the authoritative source, zero-error resolution is impossible.

Example II.50 (Side Information in Practice)**.** Consider a configuration system with DOF = 3:

- `config.yaml`: `threshold: 0.5`
- `settings.json`: `"threshold": 0.7`
- `params.toml`: `threshold = 0.6`

Resolution requires $\log_2 3 \approx 1.58$ bits of side information:

- A priority ordering: "YAML > JSON > TOML" (encodes permutation $\rightarrow$ identifies source)
- A timestamp comparison: "most recent wins" (encodes ordering $\rightarrow$ identifies source)
- An explicit declaration: "params.toml is authoritative" (directly encodes source identity)

With DOF = 1, zero bits suffice—the single source is self-evidently authoritative.

3) Multi-Terminal Capacity Interpretation: The zero-incoherence capacity theorem (Theorem II.36) can be restated in multi-terminal language:

Corollary II.51 (Multi-Terminal Interpretation)**.** *An encoding system achieves zero incoherence iff all terminals are perfectly correlated (every terminal's output is a deterministic function of one source terminal). Equivalently: the correlation structure must be a tree with one root.*

Proof. Perfect correlation means DOF = 1 (only the root is independent). By Theorem II.37, this achieves zero incoherence. If any two terminals are independent (not perfectly correlated through the root), DOF ≥ 2, and by Theorem II.38, incoherence is reachable. ■

This connects to network information theory: in a multi-terminal network, achieving coherence requires complete dependence on a single source—no "multicast" of independent copies.

M. Structure Theorems: The Derivation Lattice

The set of derivation relations on an encoding system has algebraic structure. We characterize this structure and its computational implications.

Definition II.52 (Derivation Relation)**.** A *derivation relation* $D \subseteq L \times L$ on locations L is a directed relation where $(L_s, L_d) \in D$ means L_d is derived from L_s. We require D be acyclic (no location derives from itself through any chain).

Definition II.53 (DOF under Derivation)**.** Given derivation relation D, the degrees of freedom is:

$$\text{DOF}(D) = |\{L : \nexists L'.(L', L) \in D\}|$$

The count of locations with no incoming derivation edges (source locations).

Theorem II.54 (Derivation Lattice)**.** *The set of derivation relations on a fixed set of locations* L*, ordered by inclusion, forms a bounded lattice:*

1) ***Bottom*** *(*$\bot$*):* $D = \emptyset$ *(no derivations, DOF =* $|L|$*)*
2) ***Top*** *(*$\top$*): Maximal acyclic* D *with DOF = 1 (all but one location derived)*
3) ***Meet*** *(*$\wedge$*):* $D_1 \wedge D_2 = D_1 \cap D_2$
4) ***Join*** *(*$\vee$*):* $D_1 \vee D_2 =$ *transitive closure of* $D_1 \cup D_2$ *(if acyclic)*

Proof. **Bottom:** $\emptyset$ is trivially a derivation relation with all locations independent.

Top: For n locations, a maximal acyclic relation has one source (root) and $n-1$ derived locations forming a tree or DAG. DOF = 1.

Meet: Intersection of acyclic relations is acyclic. The intersection preserves only derivations present in both.

Join: If $D_1 \cup D_2$ is acyclic, its transitive closure is the smallest relation containing both. If cyclic, join is undefined (partial lattice).

Bounded: $\emptyset \subseteq D \subseteq \top$ for all valid D. ■

Theorem II.55 (DOF is Anti-Monotonic)**.** *DOF is anti-monotonic in the derivation lattice:*

$$D_1 \subseteq D_2 \Rightarrow \textit{DOF}(D_1) \geq \textit{DOF}(D_2)$$

More derivations imply fewer independent locations.

Proof. Adding a derivation edge (L_s, L_d) to D can only decrease DOF: if L_d was previously a source (no incoming edges), it now has an incoming edge and is no longer a source. Sources can only decrease or stay constant as derivations are added. ■

Corollary II.56 (Minimal DOF = 1 Derivations)**.** *A derivation relation* D *with DOF(*D*) = 1 is* minimal *iff removing any edge increases DOF.*

Computational implication: Given an encoding system, there can be multiple DOF-1-achieving derivation structures. The minimal ones use the fewest derivation edges—the most economical way to achieve coherence.

Representation model for complexity. For the algorithmic results below, we assume the derivation relation D is given explicitly as a DAG over the location set L. The input size is $|L| + |D|$, and all complexity bounds are measured in this explicit representation.

Theorem II.57 (DOF Computation Complexity)**.** *Given an encoding system with explicit derivation relation* D*:*

1) *Computing DOF(D) is $O(|L|+|D|)$ (linear in locations plus edges)*
2) *Deciding if DOF(D) = 1 is $O(|L|+|D|)$*
3) *Finding a minimal DOF-1 extension of D is $O(|L|^2)$ in the worst case*

Proof. **(1) DOF computation:** Count locations with in-degree 0 in the DAG. Single pass over edges: $O(|D|)$ to compute in-degrees, $O(|L|)$ to count zeros.

(2) DOF = 1 decision: Compute DOF, compare to 1. Same complexity.

(3) Minimal extension: Must connect $k-1$ source locations to reduce DOF from k to 1. Finding which connections preserve acyclicity requires reachability queries. Naive: $O(|L|^2)$. With better data structures (e.g., dynamic reachability): $O(|L| \cdot |D|)$ amortized. ■

III. Optimal Encoding Rate (DOF = 1)

Having established the encoding model (Section II-B), we now prove that DOF = 1 is the unique optimal rate guaranteeing coherence under modification constraints.

A. DOF = 1 as Optimal Rate

DOF = 1 is not a design guideline. It is the information-theoretically optimal rate guaranteeing coherence for facts encoded in systems with modification constraints.

Definition III.1 (Optimal Encoding (DOF = 1))**.** Encoding system C achieves *optimal encoding rate* for fact F iff:

$$\text{DOF}(C, F) = 1$$

Equivalently: exactly one independent encoding location exists for F. All other encodings are derived.

This generalizes the "Single Source of Truth" (SSOT) principle from software engineering to universal encoding theory.

Encoding-theoretic interpretation:

- DOF = 1 means exactly one independent encoding location
- All other locations are derived (cannot diverge from source)
- Incoherence is *impossible*, not merely unlikely
- The encoding rate is minimized subject to coherence constraint

Corollary III.2 (DOF = 1 Guarantees Determinacy)**.** *If $\text{DOF}(C, F) = 1$, then for all reachable states of C, the value of F is determinate: all encodings agree.* (This restates Theorem II.6 in terms of determinacy.)

Hunt & Thomas's "single, unambiguous, authoritative representation" [9] (SSOT principle) corresponds precisely to this encoding-theoretic structure:

- **Single:** DOF = 1 (exactly one independent encoding)
- **Unambiguous:** No incoherent states possible (Theorem II.6)
- **Authoritative:** The source determines all derived values (Definition II.19)

Our contribution is proving that SSOT is not a heuristic but an information-theoretic optimality condition.

Corollary III.3 (DOF = 1 Achieves $O(1)$ Update)**.** *If $\text{DOF}(C, F) = 1$, then coherence restoration requires $O(1)$ updates: modifying the single source maintains coherence automatically via derivation.* (See Theorem VI.2 for the full rate-complexity bound.)

Corollary III.4 (Uniqueness of Optimal Rate)**.** *DOF = 1 is the **unique** rate guaranteeing coherence. DOF = 0 fails to represent F; DOF > 1 permits incoherence.* (This is the Zero-Incoherence Capacity theorem; see Theorem II.36.)

Corollary III.5 (Incoherence Under Redundancy)**.** *Multiple independent sources encoding the same fact permit incoherent states. DOF $> 1 \Rightarrow$ incoherence reachable.*

Proof. Direct application of Theorem II.7. With DOF > 1, independent locations can be modified separately, reaching states where they disagree. ■

See the formal Zero-Incoherence Capacity statement in Section II-K (Theorem II.36). A concise proof sketch appears below; the full mechanized proof is provided in Supplement A.

B. Information-theoretic proof sketches

We give concise proof sketches connecting the combinatorial DOF arguments to standard information-theoretic tools:

- **Converse (Fano):** To show that limited side information cannot enable vanishing-error recovery of F, apply the Fano-style bound (Theorem VI.5). If the mutual information $I(F; S)$ provided by derived locations is $< \log K - o(1)$ for K possible values, then any estimator has P_e bounded away from zero. Thus perfect recoverability requires $I(F; S) \approx H(F)$, i.e. essentially $\log K$ bits.
- **Confusability graph:** Lemma IV.1 converts combinatorial indistinguishability (cliques) into information lower bounds: a k-clique forces $\log k$ bits of side information for vanishing error. This gives a language-independent certificate that side information is insufficient in environments with limited introspection.
- **DOF implication:** If side information cannot supply the required mutual information, the system must retain multiple independent encodings (higher DOF) to achieve correctness under modifications. Higher DOF incurs $\Omega(n)$ manual edits (Theorem VI.3), completing the converse chain from information constraints to modification complexity.

C. Rate-Complexity Tradeoff

The DOF metric creates a fundamental tradeoff between encoding rate and modification complexity.

Question: When fact F changes, how many manual updates are required to restore coherence?

- **DOF = 1:** $O(1)$ updates. The single source determines all derived locations automatically.
- **DOF = $n > 1$:** $\Omega(n)$ updates. Each independent location must be synchronized manually.

This is a *rate-distortion* analog: higher encoding rate (DOF > 1) incurs higher modification complexity. DOF = 1 achieves minimal complexity under the coherence constraint.

Key insight: Many locations can encode F (high total encoding locations), but if DOF = 1, coherence restoration requires only 1 manual update. The derivation mechanism handles propagation automatically.

Example III.6 (Encoding Rate vs. Modification Complexity)**.** Consider an encoding system where a fact F = "all processors must implement operation P" is encoded at 51 locations:

- 1 abstract specification location
- 50 concrete implementation locations

Architecture A (DOF = 51): All 51 locations are independent.

- Modification complexity: Changing F requires 51 manual updates
- Coherence risk: After $k < 51$ updates, system is incoherent (partial updates)
- Only after all 51 updates is coherence restored

Architecture B (DOF = 1): The abstract specification is the single source; implementations are derived.

- Modification complexity: Changing F requires 1 update (the specification)
- Coherence guarantee: Derived locations update automatically via enforcement mechanism
- The *specification* has a single authoritative source

Computational realization (software): Abstract base classes with enforcement (type checkers, runtime validation) achieve DOF = 1 for contract specifications. Changing the abstract method signature updates the contract; type checkers flag non-compliant implementations.

Note: Implementations are separate facts. DOF = 1 for the contract specification does not eliminate implementation updates—it ensures the specification itself is determinate.

D. Derivation: The Coherence Mechanism

Derivation is the mechanism by which DOF is reduced without losing encodings. A derived location cannot diverge from its source, eliminating it as a source of incoherence.

Definition III.7 (Derivation)**.** Location L_{derived} is *derived from* L_{source} for fact F iff:

$$\text{updated}(L_{\text{source}}) \rightarrow \text{automatically\_updated}(L_{\text{derived}})$$

No manual intervention is required. Coherence is maintained automatically.

Derivation can occur at different times depending on the encoding system:

Derivation Time	Examples Across Domains
Compile/Build time	C++ templates, Rust macros, database schema generation, infrastructure-as-code compilation
Definition time	Python metaclasses, ORM model registration, dynamic schema creation
Query/Access time	Database views, computed columns, lazy evaluation

Structural facts require definition-time derivation. Structural facts (class existence, schema structure, service topology) are fixed when defined. Compile-time derivation that runs before the definition is fixed is too early (the declarative source is not yet fixed). Runtime is too late (structure already immutable). Definition-time is the unique opportunity for structural derivation.

Theorem III.8 (Derivation Preserves Coherence)**.** *If $L_{derived}$ is derived from L_{source}, then $L_{derived}$ cannot diverge from L_{source} and does not contribute to DOF.*

Proof. By Definition III.7, derived locations are automatically updated when the source changes. Let L_d be derived from L_s. If L_s encodes value v, then L_d encodes $f(v)$ for some function f. When L_s changes to v', L_d automatically changes to $f(v')$.

There is no reachable state where $L_s = v'$ and $L_d = f(v)$ with $v' \neq v$. Divergence is impossible. Therefore, L_d does not contribute to DOF. ■

Corollary III.9 (Derivation Achieves Coherence)**.** *If all encodings of F except one are derived from that one, then $DOF(C, F) = 1$ and coherence is guaranteed.*

Proof. Let L_s be the non-derived encoding. All other encodings $L_1, \ldots, L_k$ are derived from L_s. By Theorem III.8, none can diverge. Only L_s is independent. Therefore, $\text{DOF}(C, F) = 1$, and by Theorem II.6, coherence is guaranteed. ■

E. Computational Realizations of DOF = 1

DOF = 1 is achieved across computational domains using definition-time derivation mechanisms. We show examples from software, databases, and configuration systems.

Software: Subclass Registration (Python)

```
class Registry:
    _registry = {}
    def __init_subclass__(cls, **kwargs):
        Registry._registry[cls.__name__] = cls

class PNGHandler(Registry):  # Automatically
    registered
    pass
```

Encoding structure:

- Source: Class definition (declared once)
- Derived: Registry dictionary entry (computed at definition time via `__init_subclass__`)
- DOF = 1: Registry cannot diverge from class hierarchy

Databases: Materialized Views

```
CREATE TABLE users (id INT, name TEXT, created_at
    TIMESTAMP);
CREATE MATERIALIZED VIEW user_count AS
  SELECT COUNT(*) FROM users;
```

Encoding structure:

- Source: Base table `users`
- Derived: Materialized view `user_count` (updated on refresh)
- DOF = 1: View cannot diverge from base table (consistency guaranteed by DBMS)

Configuration: Infrastructure as Code (Terraform)

```
resource "aws_instance" "app" {
  ami = "ami-12345"
  instance_type = "t2.micro"
}

output "instance_ip" {
  value = aws_instance.app.public_ip
}
```

Encoding structure:

- Source: Resource declaration (authoritative configuration)
- Derived: Output value (computed from resource state)
- DOF = 1: Output cannot diverge from actual resource (computed at apply time)

Common pattern: In all cases, the source is declared once, and derived locations are computed automatically at definition/build/query time. Manual synchronization is eliminated. Coherence is guaranteed by the system, not developer discipline.

IV. Information-Theoretic Realizability Requirements

This section identifies, at an abstract information-theoretic level, the two properties an encoding system must provide to realize DOF = 1. Concrete programming-language and system-level instantiations are presented in the supplementary material; here we present the abstract requirements and their necessity/sufficiency proofs.

Informally, an encoding system provides a set of locations holding values and a space of allowed modifications. DOF = 1 is realizable when exactly one independent location exists (the source) and all other encodings are deterministic functions (derivations) of that source so divergence is impossible.

We show that two encoder properties are necessary and sufficient:

1) **Causal update propagation:** updates to the source must automatically propagate to derived locations (eliminating transient incoherence windows);
2) **Provenance observability:** the system must expose, to verification procedures, which locations are sources and which are derived (providing the side information necessary to verify DOF = 1).

The remainder of this section gives formal statements and proofs of necessity and sufficiency for these properties in the abstract model.

A. Confusability and Side-Information

To connect realizability to the information available at runtime, define the *confusability graph* for a fact F: vertices are distinct values of F, and an undirected edge joins two values x, x' iff they are indistinguishable given the available side information (i.e., no observation of derived locations can reliably separate x from x').

Lemma IV.1 (Confusability Clique Bound)**.** *If the confusability graph for F contains a clique of size k, then any side-information channel that yields vanishing error probability for distinguishing all k values must convey at least $\log k$ bits about F. Equivalently, if the mutual information between F and side information S satisfies $I(F;S) < \log k - \epsilon$, then the error probability for distinguishing the k-clique is bounded away from zero.*

Proof sketch. Restrict F to the k values in the clique; this induces a uniform k-ary hypothesis testing problem. Apply the Fano-style bound (Theorem VI.5 in Section VI-D) to conclude that vanishing error requires $I(F;S) \geq \log k - o(1)$. If $I(F;S)$ is strictly smaller than $\log k$, Fano's inequality implies a non-zero lower bound on error probability. ■

The confusability graph provides a compact combinatorial certificate of necessary side-information: large cliques force large information requirements and hence constrain realizability in languages/environments with limited introspection or definition-time hooks.

B. The Structural Timing Constraint

For certain classes of facts—*structural facts*—there is a fundamental timing constraint that shapes realizability.

Definition IV.2 (Structural Fact)**.** A fact F is *structural* if its encoding locations are fixed at the moment of definition. After definition, the structure cannot be retroactively modified—only new structures can be created.

Examples across domains:

- **Programming languages**: Class definitions, method signatures, inheritance relationships
- **Databases**: Schema definitions, table structures, foreign key constraints
- **Configuration systems**: Resource declarations, dependency specifications
- **Version control**: Branch structures, commit ancestry

The key property: structural facts have a *definition moment* after which their encoding is immutable. This creates a timing constraint for derivation.

Theorem IV.3 (Timing Constraint for Structural Derivation)**.** *For structural facts, derivation must occur at or before the moment the structure is fixed.*

Proof. Let F be a structural fact. Let t_{fix} be the moment F's encoding is fixed. Any derivation D that depends on F must execute at some time t_D.

Case 1: $t_D < t_{\text{fix}}$. Derivation executes before F is fixed. D cannot derive from F because F does not yet exist.

Case 2: $t_D > t_{\text{fix}}$. Derivation executes after F is fixed. D can read F but cannot modify structures derived from F—they are already fixed.

Case 3: $t_D = t_{\text{fix}}$. Derivation executes at the moment F is fixed. D can both read F and create derived structures before they are fixed.

Therefore, structural derivation requires $t_D = t_{\text{fix}}$. ■

This timing constraint is the information-theoretic reason why derivation must be *causal*—triggered by the act of defining the source, not by later access.

C. Requirement 1: Causal Update Propagation

Definition IV.4 (Causal Update Propagation)**.** An encoding system has *causal update propagation* if changes to a source location automatically trigger updates to all derived locations, without requiring explicit synchronization commands.

Formally: let L_s be a source location and L_d a derived location. The system has causal propagation iff:

$$\text{update}(L_s, v) \Rightarrow \text{automatically\_updated}(L_d, f(v))$$

where f is the derivation function. No separate "propagate" or "sync" operation is required.

Information-theoretic interpretation: Causal propagation is analogous to *channel coding with feedback*. In classical channel coding, the encoder sends a message and waits for acknowledgment. With feedback, the encoder can immediately react to channel state. Causal propagation provides "feedback" from the definition event to the derivation mechanism—the encoder (source) and decoder (derived locations) are coupled in real-time.

Connection to multi-version coding: Rashmi et al. [14] formalize consistent distributed storage where updates to a source must propagate to replicas while maintaining consistency. Their "multi-version code" requires that any c servers can decode the latest common version—a consistency guarantee analogous to our coherence requirement. Causal propagation is the mechanism by which this consistency is maintained under updates.

Why causal propagation is necessary:

Without causal propagation, there exists a temporal window between source modification and derived location update. During this window, the system is incoherent—the source and derived locations encode different values.

Theorem IV.5 (Causal Propagation is Necessary for DOF = 1)**.** *Achieving DOF = 1 for structural facts requires causal update propagation.*

Proof. By Theorem IV.3, structural derivation must occur at definition time. Without causal propagation, derived locations are not updated when the source is defined. This means:

1) The source exists with value v
2) Derived locations have not been updated; they either do not exist yet or hold stale values
3) The system is temporarily incoherent

For DOF = 1, incoherence must be *impossible*, not merely transient. Causal propagation eliminates the temporal window: derived locations are updated *as part of* the source definition, not after.

Contrapositive: If an encoding system lacks causal propagation, DOF = 1 for structural facts is unrealizable. ■

Realizations across domains:

Domain	Causal Propagation Mechanism
Python	`__init_subclass__`, metaclass `__new__`
CLOS	`:after` methods on class initialization
Smalltalk	Class creation protocol, `subclass:` method
Databases	Triggers on schema operations (PostgreSQL event triggers)
Distributed systems	Consensus protocols (Paxos, Raft)
Configuration	Terraform dependency graph, reactive bindings

Systems lacking causal propagation:

- **Java**: Annotations are metadata, not executable. No code runs at class definition.
- **C++**: Templates expand at compile time but don't execute arbitrary user code.
- **Go**: No hook mechanism. Interface satisfaction is implicit.
- **Rust**: Proc macros run at compile time but generate static code, not runtime derivation.

D. Requirement 2: Provenance Observability

Definition IV.6 (Provenance Observability)**.** An encoding system has *provenance observability* if the system supports queries about derivation structure:

1) What locations exist encoding a given fact?
2) Which locations are sources vs. derived?
3) What is the derivation relationship (which derived from which)?

Information-theoretic interpretation: Provenance observability is the encoding-system analog of *side information at the decoder*. In Slepian-Wolf coding [10], the decoder has access to correlated side information that enables decoding at rates below the source entropy. Provenance observability provides "side information" about the encoding structure itself—enabling verification that DOF = 1 holds.

Without provenance observability, the encoding system is a "black box"—you can read locations but cannot determine which are sources and which are derived. This makes DOF uncomputable from within the system.

Theorem IV.7 (Provenance Observability is Necessary for Verifiable DOF = 1)**.** *Verifying that DOF = 1 holds requires provenance observability.*

Proof. Verification of DOF = 1 requires confirming:

1) All locations encoding fact F are enumerable
2) Exactly one location is independent (the source)
3) All other locations are derived from that source

Step (1) requires querying what structures exist. Step (2) requires distinguishing sources from derived locations. Step (3) requires querying the derivation relationship.

Without provenance observability, none of these queries are answerable from within the system. DOF = 1 can hold but cannot be verified. Bugs in derivation logic go undetected until coherence violations manifest. ■

Connection to coding theory: In coding theory, a code's structure (generator matrix, parity-check matrix) must be known to the decoder. Provenance observability is analogous: the derivation structure must be queryable for verification.

Realizations across domains:

Domain	Provenance Observability Mechanism
Python	`__subclasses__()`, `__mro__`, `dir()`, `vars()`
CLOS	`class-direct-subclasses`, MOP introspection
Smalltalk	`subclasses`, `allSubclasses`
Databases	System catalogs (`pg_depend`), query plan introspection
Distributed systems	Vector clocks, provenance tracking, `etcd` watch
Configuration	Terraform `graph`, Kubernetes API server

Systems lacking provenance observability:

- **C++**: Cannot query "what types instantiated template `Foo<T>`?"
- **Rust**: Proc macro expansion is opaque at runtime.
- **TypeScript**: Types are erased. Runtime cannot query type relationships.
- **Go**: No type registry. Cannot enumerate interface implementations.

E. Independence of Requirements

The two requirements—causal propagation and provenance observability—are independent. Neither implies the other.

Theorem IV.8 (Requirements are Independent)**.** 1) *An encoding system can have causal propagation without provenance observability*
2) *An encoding system can have provenance observability without causal propagation*

Proof. **(1) Causal without provenance:** Rust proc macros execute at compile time (causal propagation: definition triggers code generation). But the generated code is opaque at runtime—the program cannot query what was generated (no provenance observability).

(2) Provenance without causal: Java provides reflection (`Class.getMethods()`, `Class.getInterfaces()`)—provenance observability. But no code executes when a class is defined—no causal propagation. ■

This independence means both requirements must be satisfied for DOF = 1 realizability.

F. The Realizability Theorem

Theorem IV.9 (Necessary and Sufficient Realizability Conditions)**.** *An encoding system S can achieve verifiable DOF = 1 for structural facts if and only if:*
1) *S provides causal update propagation, AND*
2) *S provides provenance observability*

Proof. ($\Rightarrow$) **Necessity:** Suppose S achieves verifiable DOF = 1 for structural facts.

- By Theorem IV.5, S must provide causal propagation
- By Theorem IV.7, S must provide provenance observability

($\Leftarrow$) **Sufficiency:** Suppose S provides both capabilities.

- Causal propagation enables derivation at the right moment (when structure is fixed)
- Provenance observability enables verification that all secondary encodings are derived
- Therefore, DOF = 1 is achievable: create one source, derive all others causally, verify completeness via provenance queries

■

Definition IV.10 (DOF-1-Complete Encoding System)**.** An encoding system is *DOF-1-complete* if it satisfies both causal propagation and provenance observability. Otherwise it is *DOF-1-incomplete*.

Information-theoretic interpretation: DOF-1-completeness is analogous to *channel capacity achievability*. A channel achieves capacity if there exist codes that approach the Shannon limit. An encoding system is DOF-1-complete if there exist derivation mechanisms that achieve the coherence-optimal rate (DOF = 1). The two requirements (causal propagation, provenance observability) are the "channel properties" that enable capacity achievement.

G. Connection to Write-Once Memory Codes

Our realizability requirements connect to *write-once memory (WOM) codes* [20], [21], an established area of coding theory.

A WOM is a storage medium where bits can only transition in one direction (typically $0 \to 1$). Rivest and Shamir [20] showed that WOMs can store more information than their apparent capacity by encoding multiple "writes" cleverly—the capacity for t writes is $\log_2(t+1)$ bits per cell.

The connection to our framework:

- **WOM constraint**: Bits can only increase (irreversible state change)
- **Structural fact constraint**: Structure is fixed at definition (irreversible encoding)
- **WOM coding**: Clever encoding enables multiple logical writes despite physical constraints
- **DOF = 1 derivation**: Clever derivation enables multiple logical locations from one physical source

Both settings involve achieving optimal encoding under irreversibility constraints. WOM codes achieve capacity via coding schemes; DOF-1-complete systems achieve coherence via derivation mechanisms.

H. The Logical Chain (Summary)

Observation: Structural facts are fixed at definition time (irreversible encoding).
↓ (timing analysis)
Theorem IV.3: Derivation for structural facts must occur at definition time.
↓ (requirement derivation)
Theorem IV.5: Causal update propagation is necessary for DOF = 1.
Theorem IV.7: Provenance observability is necessary for verifiable DOF = 1.
↓ (conjunction)
Theorem IV.9: An encoding system achieves DOF = 1 iff it has both properties.
↓ (evaluation)
Classification: Python, CLOS, Smalltalk are DOF-1-complete. Java, C++, Rust, Go are DOF-1-incomplete.

Every step is machine-checked in Lean 4. The proofs compile with zero `sorry` placeholders.

I. Concrete Impossibility Demonstration

We demonstrate exactly why DOF-1-incomplete systems cannot achieve DOF = 1 for structural facts.

The structural fact: "`PNGHandler` handles `.png` files." This fact must be encoded in two places:

1) The handler definition (where the handler is defined)
2) The registry/dispatcher (where format→handler mapping lives)

Python (DOF-1-complete) achieves DOF = 1:

```
class ImageHandler:
    _registry = {}

    def __init_subclass__(cls, format=None,
    **kwargs):
        super().__init_subclass__(**kwargs)
        if format:
            ImageHandler._registry[format] = cls
    # DERIVED (causal)

class PNGHandler(ImageHandler, format="png"):  #
    SOURCE
    def load(self, path): ...
```

Causal propagation: When `class PNGHandler` executes, `__init_subclass__` fires immediately, adding the registry entry. No temporal gap.

Provenance observability: `ImageHandler.__subclasses__()` returns all handlers. The derivation structure is queryable.

DOF = 1: The `format="png"` in the class definition is the single source. The registry entry is derived causally. Adding a new handler requires changing exactly one location.

Java (DOF-1-incomplete) cannot achieve DOF = 1:

```
// File 1: PNGHandler.java
@Handler(format = "png")  // Metadata, not
    executable
public class PNGHandler implements ImageHandler {
    ... }

// File 2: HandlerRegistry.java (SEPARATE SOURCE)
public class HandlerRegistry {
    static { register("png", PNGHandler.class); }
    // Manual
}
```

No causal propagation: The `@Handler` annotation is data, not code. Nothing executes when the class is defined.

Provenance partially present: Java has reflection, but cannot enumerate "all classes with `@Handler`" without classpath scanning.

DOF = 2: The annotation and the registry are independent encodings. Either can be modified without the other. Incoherence is reachable.

Theorem IV.11 (Generated Files Are Independent Sources)**.** *A generated source file constitutes an independent encoding, not a derivation. Code generation does not achieve DOF = 1.*

Proof. Let E_1 be the annotation on `PNGHandler.java`. Let E_2 be the generated `HandlerRegistry.java`.

Test: If E_2 is deleted or modified, does system behavior change? **Yes**—the handler is not registered.

Test: Can E_2 diverge from E_1? **Yes**—E_2 is a separate file that can be edited, fail to generate, or be stale.

Therefore, E_1 and E_2 are independent encodings. The fact that E_2 was *generated from* E_1 does not make it derived in the DOF sense, because:

1) E_2 exists as a separate artifact that can diverge
2) The generation process is external to the runtime and can be bypassed
3) There is no causal coupling—modification of E_1 does not automatically update E_2

Contrast with Python: the registry entry exists only in memory, created causally by the class statement. There is no second file. DOF = 1. ■

J. Summary: The Information-Theoretic Requirements

Requirement	IT Interpretation	Why Necessary
Causal propagation	Channel with feedback; encoder-decoder coupling	Eliminates temporal incoherence window
Provenance observability	Side information at decoder; codebook visibility	Enables DOF verification

Both requirements are necessary. Neither is sufficient alone. Together they enable DOF-1-complete encoding systems that achieve the coherence-optimal rate.

V. Corollary: Programming-Language Instantiation

We instantiate Theorem IV.9 in the domain of programming languages. This section is a formal corollary of the realizability theorem: once a language's definition-time hooks and

introspection capabilities are fixed, DOF = 1 realizability for structural facts is determined.

Corollary V.1 (Language Realizability Criterion)**.** *A programming language can realize DOF = 1 for structural facts iff it provides both (i) definition-time hooks and (ii) introspectable derivations. This is the direct instantiation of Theorem IV.9.*

Instantiation map. In the abstract model, an independent encoding is a location that can diverge under edits. In programming languages, structural facts are encoded at definition sites; *definition-time hooks* implement derivation (automatic propagation), and *introspection* implements provenance observability. Thus DEF corresponds to causal propagation and INTRO corresponds to queryable derivations; DOF = 1 is achievable exactly when both are present.

We instantiate this corollary over a representative language class (Definition V.2).

Definition V.2 (Mainstream Language)**.** A language is *mainstream* iff it appears in the top 20 of at least two of the following indices consistently over 5+ years:

1) TIOBE Index [22] (monthly language popularity)
2) Stack Overflow Developer Survey (annual)
3) GitHub Octoverse (annual repository statistics)
4) RedMonk Programming Language Rankings (quarterly)

A. Evaluation Criteria

The abstract realizability conditions (causal update propagation and provenance observability) map directly to concrete language capabilities (definition-time hooks and runtime introspection). For TIT readers we summarize this mapping here and defer the full language-by-language evaluation, extended code examples, and detailed rationale to Supplement A.

B. Summary Classification

Theorem IV.9 partitions languages into two classes:

Language	Def-time hooks	Introspection
DOF-1-complete (both requirements):		
Python	`__init_subclass__`	`__subclasses__()`
CLOS	`:after` methods	MOP queries
Smalltalk	class creation protocol	`subclasses`
DOF-1-incomplete (missing ≥ 1 requirement):		
Java	—	reflection
C++	—	—
Rust	proc macros (compile-time)	—
Go	—	—

Interpretation. DOF-1-complete languages can achieve DOF = 1 for structural facts using language-native mechanisms. DOF-1-incomplete languages require external tooling (code generation, IDE refactoring) which operates outside language semantics and can be bypassed (cf. Theorem IV.8 on why external tools do not establish derivation).

Supplement A contains the complete evaluation with justifications, worked code examples demonstrating each mechanism, and discussion of edge cases.

VI. Rate-Complexity Bounds

We now prove the rate-complexity bounds that make DOF = 1 optimal. The key result: the gap between DOF-1-complete and DOF-1-incomplete architectures is *unbounded*—it grows without limit as encoding systems scale.

A. Cost Model

Definition VI.1 (Modification Cost Model)**.** Let δ_F be a modification to fact F in encoding system C. The *effective modification complexity* $M_{\text{effective}}(C, \delta_F)$ is the number of syntactically distinct edit operations that must be performed manually. Formally:

$$M_{\text{effective}}(C, \delta_F) = |\{L \in \text{Locations}(C) : \text{requires\_manual\_edit}(L, \delta_F)\}|$$

where $\text{requires\_manual\_edit}(L, \delta_F)$ holds iff location L must be updated manually (not by automatic derivation) to maintain coherence after δ_F.

Unit of cost: One edit = one syntactic modification to one location. We count locations, not keystrokes or characters. This abstracts over edit complexity to focus on the scaling behavior.

What we measure: Manual edits only. Derived locations that update automatically have zero cost. This distinguishes DOF = 1 systems (where derivation handles propagation) from DOF > 1 systems (where all updates are manual).

Asymptotic parameter: We measure scaling in the number of encoding locations for fact F. Let $n = |\{L \in C : \text{encodes}(L, F)\}|$ and $k = \text{DOF}(C, F)$. Bounds of $O(1)$ and $\Omega(n)$ are in this parameter; in particular, the lower bound uses $n = k$ independent locations.

B. Upper Bound: DOF = 1 Achieves O(1)

Theorem VI.2 (DOF = 1 Upper Bound)**.** *For an encoding system with DOF = 1 for fact F:*

$$M_{\textit{effective}}(C, \delta_F) = O(1)$$

Effective modification complexity is constant regardless of system size.

Proof. Let $\text{DOF}(C, F) = 1$. By Definition III.1, C has exactly one independent encoding location. Let L_s be this single independent location.

When F changes:

1) Update L_s (1 manual edit)
2) All derived locations $L_1, \ldots, L_k$ are automatically updated by the derivation mechanism
3) Total manual edits: 1

The number of derived locations k can grow with system size, but the number of *manual* edits remains 1. Therefore, $M_{\text{effective}}(C, \delta_F) = O(1)$. ■

Note on "effective" vs. "total" complexity: Total modification complexity $M(C, \delta_F)$ counts all locations that change. Effective modification complexity counts only manual edits. With DOF = 1, total complexity can be $O(n)$ (many derived locations change), but effective complexity is $O(1)$ (one manual edit).

C. Lower Bound: DOF > 1 Requires $\Omega(n)$

Theorem VI.3 (DOF $>$ 1 Lower Bound)**.** *For an encoding system with DOF $>$ 1 for fact F, if F is encoded at n independent locations:*

$$M_{\textit{effective}}(C, \delta_F) = \Omega(n)$$

Proof. Let $\text{DOF}(C, F) = n$ where $n > 1$.

By Definition II.18, the n encoding locations are independent—updating one does not automatically update the others. When F changes:

1) Each of the n independent locations must be updated manually
2) No automatic propagation exists between independent locations
3) Total manual edits: n

Therefore, $M_{\text{effective}}(C, \delta_F) = \Omega(n)$. ■

D. Information-theoretic converse (mutual information / Fano)

To make the converse argument explicit in information-theoretic terms we give a concise mutual-information / Fano-style bound showing that insufficient side information forces a non-vanishing error probability when attempting to recover structural facts from distributed encodings.

Remark VI.4 (Formalization status). The results in this subsection apply Fano's inequality, a standard tool in information theory [13]. We state and apply these results without separate Lean formalization; the proofs are direct applications of the classical inequality. The core capacity and complexity theorems (Sections II-K–IV) are fully machine-checked.

Theorem VI.5 (Fano-style Converse)**.** *Let F be a discrete fact taking values in a set of size K. Let S denote the available side information (derived locations, observations). Let $\hat{F}(S)$ be any estimator of F based on S, and let $P_e = \Pr(\hat{F} \neq F)$. Then*

$$I(F; S) \geq H(F) - H_b(P_e) - P_e \log(K - 1),$$

where $H_b(\cdot)$ is the binary entropy. In particular, if $P_e \to 0$ then $I(F; S) \to H(F)$, so S must contain essentially $\log K$ bits about F.

Proof sketch. This is the standard Fano inequality. Let $\hat{F}$ be any estimator formed from S. Then

$$H(F|S) \leq H(F|\hat{F}) + I(F; \hat{F}|S) = H(F|\hat{F}) \leq H_b(P_e) + P_e \log(K - 1),$$

and $I(F; S) = H(F) - H(F|S)$ yields the stated bound. The terms $H_b(P_e)$ and $P_e \log(K - 1)$ vanish as $P_e \to 0$, forcing $I(F; S) \approx H(F)$. ■

This inequality formalizes the intuition used in the main text: unless the derived/side-information channels collectively convey essentially the full entropy of the fact, perfect (or vanishing-error) recovery is impossible. Translating this to modification complexity shows that when side-information capacity is limited, systems must retain multiple independent encodings (higher DOF), incurring larger manual update costs.

E. Information-constrained DOF lower bound

The following lemma makes the link explicit between mutual-information capacity of side information and the necessity of multiple independent encodings.

Lemma VI.6 (Information-constrained DOF Lower Bound)**.** *Let F be a fact uniformly distributed over K values. Let S be the collective side information provided by derived locations. If*

$$I(F; S) < \log K - \delta$$

for some $\delta > 0$, then any encoding system that guarantees vanishing-error recovery of F from S must include at least two independent encoding locations for some subset of values (i.e., DOF > 1 for those values). Consequently, for facts encoded at n independent locations the effective modification complexity satisfies $M_{\textit{effective}} = \Omega(n)$.

Proof sketch. By Theorem VI.5, if $I(F; S) < \log K - \delta$ then any estimator has error probability bounded away from zero for the K-ary hypothesis. To achieve correctness despite the insufficient side information, the system must retain additional independent encodings that act as auxiliary sources of information about F; these independent encodings increase DOF. Once DOF>1 over a set of n independent locations, Theorem VI.3 implies $M_{\text{effective}} = \Omega(n)$, completing the chain from information constraints to modification complexity. ■

F. Worked example: numeric mutual-information calculation

Consider a fact F uniformly distributed over $K = 4$ values, so $H(F) = \log_2 4 = 2$ bits. Suppose the collective derived side information S conveys only $I(F; S) = 1$ bit about F (for example, a summary or hash with one bit of mutual information).

Apply the Fano-style bound (Theorem VI.5). Let P_e denote the probability of incorrect recovery of F from S. Then

$$H(F|S) = H(F) - I(F; S) = 2 - 1 = 1 \text{ bit.}$$

Fano's inequality gives

$$H_b(P_e) + P_e \log_2(K - 1) \geq H(F|S) = 1.$$

Plugging $K - 1 = 3$ and solving numerically yields a lower bound $P_e \gtrsim 0.19$ (approximately 19% error). Concretely: $H_b(0.19) \approx 0.695$ and $0.19 \log_2 3 \approx 0.301$, summing to $\approx 0.996 \approx 1$.

Interpretation: with only 1 bit of side information for a 2-bit fact, no estimator can recover F with error lower than roughly 19%; therefore, to guarantee vanishing error one must supply more side information (increase $I(F; S)$) or retain extra independent encodings (increase DOF), which in turn raises manual modification costs per Theorem VI.3.

As a second illustration, if $I(F; S) = 1.5$ bits then $H(F|S) = 0.5$ and Fano's bound requires a much smaller P_e (solve $H_b(P_e) + P_e \log_2 3 \geq 0.5$ numerically to obtain $P_e \approx 0.08$), showing the smooth tradeoff between conveyed information and achievable error.

Tightness (Achievability + Converse). Theorems VI.2 and VI.3 form a tight information-theoretic bound: DOF =

1 achieves constant modification cost (achievability), while any encoding with more than one independent location incurs linear cost in the number of independent encodings (converse). There is no intermediate regime with sublinear manual edits when $k > 1$ independent encodings are permitted.

G. The Unbounded Gap

Theorem VI.7 (Unbounded Gap)**.** *The ratio of modification complexity between DOF-1-incomplete and DOF-1-complete architectures grows without bound:*

$$\lim_{n\to\infty} \frac{M_{DOF>1}(n)}{M_{DOF=1}} = \lim_{n\to\infty} \frac{n}{1} = \infty$$

Proof. By Theorem VI.2, $M_{\text{DOF}=1} = O(1)$. Specifically, $M_{\text{DOF}=1} = 1$ for any system size.

By Theorem VI.3, $M_{\text{DOF}>1}(n) = \Omega(n)$ where n is the number of independent encoding locations.

The ratio is:

$$\frac{M_{\text{DOF}>1}(n)}{M_{\text{DOF}=1}} = \frac{n}{1} = n$$

As $n \to \infty$, the ratio $\to \infty$. The gap is unbounded. ■

Corollary VI.8 (Arbitrary Reduction Factor)**.** *For any constant k, there exists a system size n such that DOF = 1 provides at least $k\times$ reduction in modification complexity.*

Proof. Choose $n = k$. Then $M_{\text{DOF}>1}(n) = n = k$ and $M_{\text{DOF}=1} = 1$. The reduction factor is $k/1 = k$. ■

H. The (R, C, P) Tradeoff Space

We now formalize the complete tradeoff space, analogous to rate-distortion theory in classical information theory.

Definition VI.9 ((R, C, P) Tradeoff)**.** For an encoding system, define:

- R = *Rate* (DOF): Number of independent encoding locations
- C = *Complexity*: Manual modification cost per change
- P = *Coherence indicator*: $P = 1$ iff no incoherent state is reachable; otherwise $P = 0$

The *(R, C, P) tradeoff space* is the set of achievable (R, C, P) tuples.

Theorem VI.10 (Operating Regimes)**.** *The (R, C, P) space has three distinct operating regimes:*

Rate	***Complexity***	***Coherence***	***Interpretation***
$R = 0$	$C = 0$	$P =$ *undefined*	*Fact not encoded*
$R = 1$	$C = O(1)$	$P = 1$	***Optimal (capacity-achieving)***
$R > 1$	$C = \Omega(R)$	$P = 0$	*Above capacity*

Proof. $R = 0$**:** No encoding exists. Complexity is zero (nothing to modify), but coherence is undefined (nothing to be coherent about).

$R = 1$**:** By Theorem VI.2, $C = O(1)$. By Theorem II.36, $P = 1$ (coherence guaranteed). This is the capacity-achieving regime.

$R > 1$**:** By Theorem VI.3, $C = \Omega(R)$. By Theorem II.7, incoherent states are reachable, so $P = 0$. ■

Definition VI.11 (Pareto Frontier)**.** A point (R, C, P) is *Pareto optimal* if no other achievable point dominates it (lower R, lower C, or higher P without worsening another dimension).

The *Pareto frontier* is the set of all Pareto optimal points.

Theorem VI.12 (Pareto Optimality of DOF = 1)**.** $(R = 1, C = 1, P = 1)$ *is the unique Pareto optimal point for encoding systems requiring coherence ($P = 1$).*

Proof. We show $(1, 1, 1)$ is Pareto optimal and unique:

Existence: By Theorems VI.2 and II.36, the point $(1, 1, 1)$ is achievable.

Optimality: Consider any other achievable point (R', C', P') with $P' = 1$:

- If $R' = 0$: Fact is not encoded (excluded by requirement)
- If $R' = 1$: Same as $(1, 1, 1)$ (by uniqueness of C at $R = 1$)
- If $R' > 1$: By Theorem II.7, $P' < 1$, contradicting $P' = 1$

Uniqueness: No other point achieves $P = 1$ except $R = 1$. ■

Information-theoretic interpretation. The Pareto frontier in rate-distortion theory is the curve $R(D)$ of minimum rate achieving distortion D. Here, the "distortion" is $1 - P$ (indicator of incoherence reachability), and the Pareto frontier collapses to a single point: $R = 1$ is the unique rate achieving $D = 0$.

Corollary VI.13 (No Tradeoff at $P = 1$)**.** *Unlike rate-distortion where you can trade rate for distortion, there is no tradeoff at $P = 1$ (perfect coherence). The only option is $R = 1$.*

Proof. Direct consequence of Theorem II.36. ■

Comparison to rate-distortion. In rate-distortion theory:

- You can achieve lower distortion with higher rate (more bits)
- The rate-distortion function $R(D)$ is monotonically decreasing
- $D = 0$ (lossless) requires $R = H(X)$ (source entropy)

In our framework:

- You *cannot* achieve higher coherence (P) with more independent locations
- Higher rate ($R > 1$) *eliminates* coherence guarantees ($P = 0$)
- $P = 1$ (perfect coherence) requires $R = 1$ exactly

The key difference: redundancy (higher R) *hurts* rather than helps coherence (without coordination). This inverts the intuition from error-correcting codes, where redundancy enables error detection/correction. Here, redundancy without derivation enables errors (incoherence).

I. Practical Implications

The unbounded gap has practical implications:

1. DOF = 1 matters more at scale. For small systems ($n = 3$), the difference between 3 edits and 1 edit is minor. For large systems ($n = 50$), the difference between 50 edits and 1 edit is significant.

2. The gap compounds over time. Each modification to fact F incurs the complexity cost. If F changes m times over the system lifetime, total cost is $O(mn)$ with DOF > 1 vs. $O(m)$ with DOF = 1.

3. The gap affects error rates. Each manual edit is an opportunity for error. With n edits, the probability of at least one error is $1-(1-p)^n$ where p is the per-edit error probability. As n grows, this approaches 1.

Example VI.14 (Error Rate Calculation)**.** Assume a 1% error rate per edit ($p = 0.01$).

Edits (n)	P(at least one error)	Architecture
1	1.0%	DOF = 1
10	9.6%	DOF = 10
50	39.5%	DOF = 50
100	63.4%	DOF = 100

With 50 independent encoding locations (DOF = 50), there is a 39.5% chance of introducing an error when modifying fact F. With DOF = 1, the chance is 1%.

J. Amortized Analysis

The complexity bounds assume a single modification. Over the lifetime of an encoding system, facts are modified many times.

Theorem VI.15 (Amortized Complexity)**.** *Let fact F be modified m times over the system lifetime. Let n be the number of independent encoding locations. Total modification cost is:*

- *DOF = 1: $O(m)$*
- *DOF = $n > 1$: $O(mn)$*

Proof. Each modification costs $O(1)$ with DOF = 1 and $O(n)$ with DOF = n. Over m modifications, total cost is $m \cdot O(1) = O(m)$ with DOF = 1 and $m \cdot O(n) = O(mn)$ with DOF = n. ■

For a fact modified 100 times with 50 independent encoding locations:

- DOF = 1: 100 edits total
- DOF = 50: 5,000 edits total

The 50× reduction factor applies to every modification, compounding over the system lifetime.

VII. Corollary: Realizability Patterns (Worked Example)

This section contained an extended worked example and multiple detailed code snippets illustrating realizability patterns from the OpenHCS case study. For the revised TIT-focused manuscript we move the full worked example, all code excerpts, and the detailed DOF measurements to Supplement A ("Practical Patterns and Case Study").

Below we give a short summary and pointer to the supplement; the complete instantiation, PR traces, and verifiable diffs are available in Supplement A.

Summary (short):

- Four recurring patterns realize DOF = 1: contract enforcement, automatic registration, automatic discovery, and introspection-driven code generation.
- A publicly verifiable OpenHCS pull request (PR 44) demonstrates a 47→1 DOF reduction; full diffs and measurements are in Supplement A.
- The worked examples illustrate why both definition-time hooks and runtime introspection are necessary for DOF = 1; language-by-language impossibility notes are in Supplement A.

For the interested reader, Supplement A contains the complete before/after code, the methodology, measurement tables, and links to the public PR and repository snapshots.

VIII. Related Work

This section surveys related work across five areas: zero-error and multi-terminal source coding, interactive information theory, distributed systems, computational reflection, and formal methods.

A. Zero-Error and Multi-Terminal Source Coding

Our zero-incoherence capacity theorem extends classical source coding to interactive multi-terminal systems.

Zero-Error Capacity. Shannon [1] introduced zero-error capacity: the maximum rate achieving exactly zero error probability. Körner [2] connected this to graph entropy, and Lovász [3] characterized the Shannon capacity of the pentagon graph. Our zero-incoherence capacity is the storage analog: the maximum encoding rate achieving exactly zero incoherence probability. The achievability/converse structure (Theorems II.37, II.38) parallels zero-error proofs. The key parallel: zero-error capacity requires distinguishability between codewords; zero-incoherence capacity requires indistinguishability (all locations must agree).

Multi-Terminal Source Coding. Slepian and Wolf [10] characterized distributed encoding of correlated sources: sources (X, Y) can be encoded at rates satisfying $R_X \geq H(X|Y)$ when Y is decoder side information. We model encoding locations as terminals (Section II-L). Derivation introduces *perfect correlation*: a derived terminal's output is a deterministic function of its source, so $H(L_d|L_s) = 0$. The capacity result shows that only complete correlation (all terminals derived from one source) achieves zero incoherence.

Multi-Version Coding. Rashmi et al. [14] formalize consistent distributed storage where multiple versions must be accessible while maintaining consistency. They prove an "inevitable price, in terms of storage cost, to ensure consistency." Our DOF = 1 theorem is analogous: we prove the *encoding rate* cost of ensuring coherence. Where multi-version coding trades storage for version consistency, we trade encoding rate for location coherence.

Write-Once Memory Codes. Rivest and Shamir [20] introduced WOM codes for storage media where bits can only

transition $0 \to 1$. Despite this irreversibility constraint, clever coding achieves capacity $\log_2(t+1)$ for t writes—more than the naive 1 bit.

Our structural facts have an analogous irreversibility: once defined, structure is fixed. The parallel:

- **WOM:** Physical irreversibility (bits only increase) $\Rightarrow$ coding schemes maximize information per cell
- **DOF = 1:** Structural irreversibility (definition is permanent) $\Rightarrow$ derivation schemes minimize independent encodings

Wolf [21] extended WOM capacity results; our realizability theorem (Theorem IV.9) characterizes what encoding systems can achieve DOF = 1 under structural constraints.

Classical Source Coding. Shannon [19] established source coding theory for static data. Slepian and Wolf [10] extended to distributed sources with correlated side information, proving that joint encoding of (X, Y) can achieve rate $H(X|Y)$ for X when Y is available at the decoder.

Our provenance observability requirement (Section IV-D) is the encoding-system analog: the decoder (verification procedure) has "side information" about the derivation structure, enabling verification of DOF = 1 without examining all locations independently.

Rate-Distortion Theory. Cover and Thomas [13] formalize the rate-distortion function $R(D)$: the minimum encoding rate to achieve distortion D. Our rate-complexity tradeoff (Theorem VI.7) is analogous: encoding rate (DOF) trades against modification complexity. DOF = 1 achieves $O(1)$ complexity; DOF > 1 incurs $\Omega(n)$.

Interactive Information Theory. The BIRS workshop [11] identified interactive information theory as an emerging area combining source coding, channel coding, and directed information. Ma and Ishwar [12] showed that interaction can reduce rate for function computation. Xiang [23] studied interactive schemes including feedback channels.

Our framework extends this to *storage* rather than communication: encoding systems where the encoding itself is modified over time, requiring coherence maintenance.

Minimum Description Length. Rissanen [4] established MDL: the optimal model minimizes total description length (model + data given model). Grünwald [5] proved uniqueness of MDL-optimal representations.

DOF = 1 is the MDL-optimal encoding for redundant facts: the single source is the model; derived locations have zero marginal description length (fully determined by source). Additional independent encodings add description length without reducing uncertainty—pure overhead. Our Theorem II.24 establishes analogous uniqueness for encoding systems under modification constraints.

a) Closest prior work and novelty.: The closest IT lineage is multi-version coding and zero-error/interactive source coding. These settings address consistency or decoding with side information, but they do not model *modifiable* encodings with a coherence constraint over time. Our contribution is a formal encoding model with explicit modification operations, a coherence capacity theorem (unique rate for guaranteed coherence), an iff realizability characterization, and tight rate–complexity bounds.

B. Distributed Systems Consistency

We give formal encoding-theoretic versions of CAP and FLP in Section II-I. The connection is structural: CAP corresponds to the impossibility of coherence when replicated encodings remain independently updatable, and FLP corresponds to the impossibility of truth-preserving resolution in incoherent states without side information. Consensus protocols (Paxos [24], Raft [25]) operationalize this by enforcing coordination, which in our model corresponds to derivation (reducing DOF).

C. Computational Reflection and Metaprogramming

Metaobject protocols and reflection. Kiczales et al. [26] and Smith [27] provide the classical foundations for systems that can execute code at definition time and introspect their own structure. These mechanisms correspond directly to causal propagation and provenance observability in our realizability theorem, explaining why MOP-equipped languages admit DOF = 1 for structural facts.

Generative complexity. Heering [16], [28] formalizes minimal generators for program families. DOF = 1 systems realize this minimal-generator viewpoint by construction: the single source is the generator and derived locations are generated instances.

D. Software Engineering Principles

Classical software-engineering principles such as DRY [9], information hiding [29], and code-duplication analyses [30], [31] motivate coherence and single-source design. Our contribution is not another guideline, but a formal encoding model and theorems that explain when such principles are forced by information constraints. These connections are interpretive; the proofs do not rely on SE assumptions.

E. Formal Methods

Our Lean 4 [15] formalization follows the tradition of mechanized theory (e.g., Pierce [32], Winskel [33], CompCert [34]), but applies it to an information-theoretic encoding model.

F. Novelty and IT Contribution

To our knowledge, this is the first work to:

1) **Define zero-incoherence capacity**—the maximum encoding rate guaranteeing zero probability of location disagreement, extending zero-error capacity to multi-location storage.
2) **Prove a capacity theorem with achievability/converse**—$C_0 = 1$ exactly, with explicit achievability (Theorem II.37) and converse (Theorem II.38) following the Shannon proof structure.
3) **Quantify side information for resolution**—$\geq \log_2 k$ bits for k-way incoherence (Theorem II.47), connecting to Slepian-Wolf decoder side information.
4) **Characterize encoder realizability**—causal propagation (feedback) and provenance observability (side information) are necessary and sufficient for achieving capacity (Theorem IV.9).

5) **Establish rate-complexity tradeoffs**—$O(1)$ at capacity vs. $\Omega(n)$ above capacity, with unbounded gap (Theorem VI.7).

Relation to classical IT.

Classical IT Concept	This Work	Theorem
Zero-error capacity	Zero-incoherence capacity	II.36
Channel capacity proof	Achievability + converse	II.37, II.38
Slepian-Wolf side info	Resolution side info	II.47
Multi-terminal correlation	Derivation as correlation	Def. II.19
Feedback channel	Causal propagation	Thm. IV.5
Rate-distortion tradeoff	Rate-complexity tradeoff	VI.7

What is new: The setting (interactive multi-location encoding with modifications), the capacity theorem for this setting, the side information bound, the encoder realizability iff, and the machine-checked proofs. The instantiations (programming languages, databases) are corollaries illustrating the abstract theory.

IX. Conclusion

We have established a new capacity theorem extending zero-error source coding to interactive multi-location encoding systems. The key contributions are:

1. Zero-Incoherence Capacity Theorem: We define zero-incoherence capacity C_0 as the maximum encoding rate guaranteeing zero probability of location disagreement, and prove $C_0 = 1$ exactly (Theorem II.36). The proof follows the achievability/converse structure of Shannon's channel capacity theorem.

2. Side Information Bound: We prove that resolution of k-way incoherence requires $\geq \log_2 k$ bits of side information (Theorem II.47). This connects to Slepian-Wolf distributed source coding: provenance information acts as decoder side information.

3. Multi-Terminal Interpretation: We model encoding locations as terminals in a multi-terminal source coding problem. Derivation introduces perfect correlation (deterministic dependence), reducing effective rate. Only complete correlation (all terminals derived from one source) achieves zero incoherence.

4. Rate-Complexity Tradeoffs: We establish tradeoffs analogous to rate-distortion: $O(1)$ modification complexity at capacity vs. $\Omega(n)$ above capacity. The gap is unbounded (Theorem VI.7).

5. Encoder Realizability: Achieving capacity requires two encoder properties: causal propagation (analogous to feedback) and provenance observability (analogous to decoder side information). Both necessary; together sufficient (Theorem IV.9).

Corollary instantiations. The abstract theory instantiates across domains (programming languages, distributed databases, configuration systems). Sections V and VII provide illustrative corollaries; the core theorems are domain-independent.

Implications:

1) **For information theorists:** Zero-error capacity theory extends to interactive multi-location encoding. The setting (modifiable encodings, coherence constraints) is new; the achievability/converse structure and side information bounds connect to established IT.
2) **For coding theorists:** Derivation is the mechanism that introduces correlation, reducing effective encoding rate. The encoder realizability theorem characterizes what encoder properties enable capacity-achieving codes.
3) **For system designers:** The capacity theorem is a forcing result: if coherence is required, encoding rate must be ≤ 1. Systems operating above capacity require external side information for resolution.

Limitations:

- Results apply primarily to facts with modification constraints. Streaming data and high-frequency updates have different characteristics.
- The complexity bounds are asymptotic. For small encoding systems (DOF < 5), the asymptotic gap is numerically small.
- Computational realization examples are primarily from software systems. The theory is general, but database and configuration system case studies are limited to canonical examples.
- Realizability requirements focus on computational systems. Physical and biological encoding systems require separate analysis.

Future Work:

- **Probabilistic coherence:** Extend to soft constraints where incoherence probability is bounded but non-zero, analogous to the transition from zero-error to vanishing-error capacity.
- **Network encoding:** Study coherence capacity in networked encoding systems with communication constraints, connecting to network information theory.
- **Rate-distortion extension:** Characterize the full rate-complexity function $R(M)$ trading encoding rate against modification complexity, analogous to rate-distortion $R(D)$.
- **Interactive capacity:** Study capacity under multi-round modification protocols, connecting to interactive information theory and directed information.
- **Partial correlation:** Characterize coherence guarantees when derivation introduces partial (non-deterministic) correlation, extending beyond the perfect-correlation case.

A. Artifacts

The Lean 4 formalization is included as supplementary material [35]. The OpenHCS case study and associated code references are provided for the worked instantiation (Section VII).

References

[1] C. E. Shannon, "Zero-error capacity of a noisy channel," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 8–19, 1956, introduces zero-error capacity; fundamental limit for error-free communication over noisy channels.

[2] J. Körner, "Coding of an information source having ambiguous alphabet and the entropy of graphs," *Transactions of the 6th Prague Conference on Information Theory*, pp. 411–425, 1973, zero-error source coding; graph entropy characterizes zero-error capacity.

[3] L. Lovász, "On the shannon capacity of a graph," *IEEE Transactions on Information Theory*, vol. 25, no. 1, pp. 1–7, 1979, theta function bounds Shannon capacity; foundational for zero-error information theory.

[4] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978, foundational paper on Minimum Description Length (MDL) principle: total description = model + data given model; optimal model minimizes this sum.

[5] P. D. Grünwald, *The Minimum Description Length Principle*. MIT Press, 2007, comprehensive treatment of MDL; DOF=1 (SSOT) can be framed as the MDL-optimal representation for redundant facts.

[6] E. A. Brewer, "Towards robust distributed systems," in *Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing*. ACM, 2000, keynote introducing CAP conjecture: Consistency, Availability, Partition tolerance—pick two.

[7] J. J. Hunt, K.-P. Vo, and W. F. Tichy, "Delta algorithms: An empirical analysis," *ACM Transactions on Software Engineering and Methodology*, vol. 7, no. 2, pp. 192–214, 1998, delta compression for version control; coherence under branch divergence.

[8] T. Delaet, W. Joosen, and J. Van de Craen, "A survey of system configuration tools," *Proceedings of the 24th Large Installation System Administration Conference (LISA)*, 2010, survey of configuration management systems; coherence challenges in multi-file configurations.

[9] A. Hunt and D. Thomas, *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley Professional, 1999.

[10] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, vol. 19, no. 4, pp. 471–480, 1973, distributed source coding with side information; decoder uses correlated information to reduce required rate.

[11] Banff International Research Station, "Interactive information theory (12w5119)," Workshop Report, 2012, workshop on interactive IT: source coding, channel coding, directed information; https://www.birs.ca/events/2012/5-day-workshops/12w5119.

[12] N. Ma and P. Ishwar, "Some results on distributed source coding for interactive function computation," *IEEE Transactions on Information Theory*, vol. 57, no. 9, pp. 6180–6195, 2011, interactive distributed source coding; shows interaction can reduce rate for function computation.

[13] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley-Interscience, 2006, comprehensive textbook covering rate-distortion theory, source coding, channel coding.

[14] K. V. Rashmi, N. B. Shah, K. Ramchandran, and D. Gu, "Multi-version coding—an information-theoretic perspective of consistent distributed storage," *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 4111–4128, 2017, information-theoretic framework for consistent distributed storage; proves inevitable storage cost for consistency guarantees.

[15] L. de Moura and S. Ullrich, "The Lean 4 theorem prover and programming language," in *Automated Deduction – CADE 28*. Springer, 2021, pp. 625–635.

[16] J. Heering, "Generative software complexity," *Science of Computer Programming*, vol. 97, pp. 82–85, 2015, proposes Kolmogorov complexity to measure software structure; the shortest generator for a set of programs indicates maximal complexity reduction.

[17] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *ACM SIGACT News*, vol. 33, no. 2, pp. 51–59, 2002, formal proof of CAP theorem.

[18] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM*, vol. 32, no. 2, pp. 374–382, 1985, fLP impossibility: deterministic consensus impossible in async systems with one failure.

[19] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948, foundational paper establishing information theory; source coding and channel capacity.

[20] R. L. Rivest and A. Shamir, "How to reuse a "write-once" memory," in *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*. ACM, 1982, pp. 105–113, foundational paper on write-once memory codes; capacity for t writes is $\log_2(t+1)$ bits per cell.

[21] J. K. Wolf *et al.*, "Coding for a write-once memory," *AT&T Bell Laboratories Technical Journal*, vol. 63, no. 6, pp. 1089–1112, 1984, extends WOM codes; establishes capacity results for irreversible storage.

[22] TIOBE Software BV, "TIOBE index for programming languages," TIOBE Programming Community Index, 2024, online: tiobe.com/tiobe-index/.

[23] Y. Xiang, "Interactive schemes in information theory and statistics," Ph.D. dissertation, University of California, Berkeley, 2013, interactive information theory; feedback channels, distributed inference.

[24] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems*, vol. 16, no. 2, pp. 133–169, 1998, paxos consensus algorithm; achieves consistency under partial failure.

[25] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proceedings of the 2014 USENIX Annual Technical Conference*. USENIX, 2014, pp. 305–319, raft consensus algorithm; designed for understandability.

[26] G. Kiczales, J. des Rivières, and D. G. Bobrow, *The Art of the Metaobject Protocol*. MIT press, 1991.

[27] B. C. Smith, "Reflection and semantics in lisp," in *Proceedings of the 11th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, 1984, pp. 23–35.

[28] J. Heering, "Software architecture and software configuration management," in *Software Configuration Management*, ser. Lecture Notes in Computer Science. Springer, 2003, vol. 2649, pp. 1–16, introduces generative complexity as structural complexity measure for software.

[29] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Communications of the ACM*, vol. 15, no. 12, pp. 1053–1058, 1972.

[30] M. Fowler, *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 1999.

[31] C. K. Roy and J. R. Cordy, "A survey on software clone detection research," *School of Computing TR 2007-541, Queen's University*, vol. 115, pp. 64–68, 2007.

[32] B. C. Pierce, *Types and programming languages*. MIT press, 2002.

[33] G. Winskel, *The Formal Semantics of Programming Languages: An Introduction*. MIT press, 1993.

[34] X. Leroy, "Formal verification of a realistic compiler," *Communications of the ACM*, vol. 52, no. 7, pp. 107–115, 2009.

[35] T. Simas, "Lean 4 formalization: SSOT requirements theorems," Supplementary material, 2025, 9,369 lines across 29 files, 441 theorems, 0 `sorry` placeholders. Included with paper submission.

APPENDIX

MECHANIZED PROOFS (LEAN 4)

The core theorems in this paper (capacity, realizability, complexity bounds) are machine-checked in Lean 4. The complete proof corpus (9,369 lines across 29 files, 441 theorems/lemmas, 0 `sorry` placeholders) is provided in Supplement A. The entropy and mutual-information results in Section VI-D apply standard information theory (Fano's inequality) and are not separately formalized.

Theorem Correspondence

The following table maps core paper theorems to their Lean formalizations:

Paper Theorem	Lean Theorem	File
Capacity Theorem (Section II-K)		
Thm. II.6	`dof_one_implies_coherent`	`Coherence.lean`
Thm. II.7	`dof_gt_one_incoherence_possible`	`Coherence.lean`
Thm. II.36	`determinate_truth_forces_ssot`	`Coherence.lean`
Thm. II.37	`dof_one_implies_coherent`	`Coherence.lean`
Thm. II.38	`non_ssot_permits_incoherence`	`Coherence.lean`
Thm. II.4	`oracle_arbitrary`	`Coherence.lean`
Side Information (Section II-L)		
Thm. II.47	`coherence_restoration_eq_dof`	`Coherence.lean`
Realizability (Section IV)		
Thm. IV.3	`structural_facts_fixed_at_definition`	`Foundations.lean`
Thm. IV.5	`definition_hooks_necessary`	`Requirements.lean`
Thm. IV.7	`introspection_necessary_for_verification`	`Requirements.lean`
Thm. IV.8	`both_requirements_independent`	`Requirements.lean`
Thm. IV.9	`model_completeness`	`Foundations.lean`
Rate-Complexity Bounds (Section VI)		
Thm. VI.2	`ssot_upper_bound`	`Bounds.lean`
Thm. VI.3	`non_ssot_lower_bound`	`Bounds.lean`
Thm. VI.7	`ssot_advantage_unbounded`	`Bounds.lean`

Verification Summary

Component	Lines	Theorems
Core encoding theory (`SSOT.lean`, `Coherence.lean`, `Dof.lean`)	509	29
Grounded operational semantics (`AbstractClassSystem.lean`, etc.)	5,963	180
Encoding theory bridge (`SSOTGrounded.lean`, `Foundations.lean`)	1,184	45
Computational system instantiations (`Lang*.lean`, `*Instantiation.lean`)	1,713	187
Total	**9,369**	**441**

Verification Instructions

To verify locally:

1) Install Lean 4 and Lake: https://leanprover.github.io/
2) From the repository root: `lake build -project proofs`
3) The build verifies all 441 theorems; there are no `sorry` placeholders.

Proof Architecture

The formalization grounds the abstract encoding theory in operational semantics:

- **Core definitions** (`SSOT.lean`): DOF as `Nat`, `satisfies_SSOT dof := dof = 1`
- **Coherence** (`Coherence.lean`): `EncodingSystem` with `is_coherent`/`is_incoherent` predicates; achievability and converse proofs
- **Timing trichotomy** (`Foundations.lean`): `TimingRelation` type with exhaustiveness proof; derivation mechanism enumeration
- **Realizability** (`Requirements.lean`): `LanguageFeatures` structure; necessity proofs for definition hooks and introspection
- **Impossibility** (`Foundations.lean`): Constructive witnesses showing only `source_hooks` achieves SSOT

The key insight: computational system claims (e.g., "Python achieves DOF = 1") derive from formalized operational semantics, not boolean flags. The proof chain traces from abstract theorems through `execute_class_statement` to concrete language features.

Full listings, the model correspondence argument, and detailed proof commentary are in Supplement A.