Requisite Information for Automation

Structural Feasibility, Functional Complexity Bounds, and Requirements-to-Test Certificates

Joshua Farrow[1]

[1]Independent Researcher

July 2026

Revised draft manuscript

Abstract

Large language models, tool-using agents, and workflow systems can pass sampled tests while remaining structurally incapable of the stated use case. This paper develops a requirements-to-verification method for determining whether an automation has enough observable distinctions, functional states, memory, queries, outputs, tools, and information coupling to realize a specification. For finite one-shot tasks, exact deterministic feasibility holds precisely when every set of environment states collapsed to the same exposed input shares a permissible action. Unique-action requirements therefore factor through the input interface, and their task quotient gives the coarsest exact representation. Candidate inputs induce separation problems; adaptive acquisition induces decision-tree query bounds; permissible outputs induce acceptable-action covers; and sequential behavior induces future-distinguishable memory classes. These structural quantities are connected to a task-action rate–distortion curve using the classical indirect-distortion reduction. The framework compiles requirements into a requisite complexity certificate and generated separation, invariance, trace, and refinement tests. A reproducible 32-scenario appointment case study finds four required action classes, 0.731 bits of actionable entropy, and exact optimal query costs of five worst-case and 1.9375 expected queries. Omitting one required field creates a 6.25% irreducible error floor while a deficient implementation still passes all sampled tests. The contribution is a pre-coding, cross-domain assurance method, not a new source-coding theorem.

Keywords: automation, rate–distortion theory, requirements engineering, program synthesis, model-based testing, requisite variety, information bottleneck, AI agents, functional complexity

1 Introduction

An automation receives some trace of a world and is expected to do something appropriate. Between the world and the action lies an interface: sensors, forms, prompts, database fields, event streams, retrieval systems, memory, tool schemas, human reports, or combinations of these. The interface determines which environmental distinctions become available to the automation. The implementation then determines whether those distinctions are retained, computed over, and converted into permissible effects.

This boundary has become especially consequential as software is assembled from large language models, agent loops, workflow engines, retrieval systems, and external tools. Such systems can pass a conventional test suite while remaining structurally incapable of the stated use case. A test suite may omit one decision boundary. A prompt may omit one field that changes the permitted action. A memory summary may merge two histories that require different future behavior. A

workflow may have many nodes yet lack one required escalation, rollback, or idempotency state. More downstream intelligence cannot reconstruct a distinction that never entered the system or was irreversibly discarded.

The phrase "complex enough" must therefore be used carefully. The relevant quantity is not lines of code, prompt length, model parameter count, number of workflow nodes, or cyclomatic complexity. A one-line lookup can exactly implement a large task, while a billion-parameter model can be inadequate when its input interface merges two cases requiring incompatible actions. The appropriate question is functional:

> *What distinctions, information, memory, queries, outputs, and computational capabilities must any acceptable implementation possess, given the inputs and behavior specified for the task?*

This paper answers at three linked levels.

1. **Structural realizability.** Before coding or rate calculations, can the required behavior be a function of the proposed inputs at all?
2. **Quantitative lower bounds.** If realizable, how many decision classes, input distinctions, memory states, and average information bits are required at a stated tolerance?
3. **Requirements-derived verification.** How can those lower bounds become pre-coding acceptance criteria and post-coding witness tests?

The paper makes seven contributions.

1. It presents a *requirements-to-certificate pipeline* that compiles a behavioral specification into feasibility results, lower bounds, capability obligations, and generated tests.
2. It gives an exact finite input-fiber criterion for one-shot deterministic automation and identifies the task quotient as the coarsest exact representation in the unique-output case.
3. It formulates minimum candidate-input selection as a separation or hitting-set problem, adaptive field acquisition as an optimal decision-tree problem, and robust permissible-output structure as an acceptable-action set cover, with explicit computational complexity qualifications.
4. It defines a requisite information curve and reduces it to task-action rate–distortion using the classical indirect-distortion construction, while separating one-shot lower bounds from asymptotic coding rates.
5. It connects the zero-error special case to confusability graphs, chromatic number, graph entropy, and side-information coding, while showing why overlapping permissible-action sets generally require a hypergraph or set-system formulation.
6. It extends the framework to programming requirements, reactive realizability, finite-state memory, information-acquisition actions, tools, and computational-resource obligations.
7. It supplies a reproducible computational validation and a finite programming-requirements case study demonstrating a structural failure missed by sampled tests.

The central claim is deliberately narrower than "complex environments require complex automations." Raw environmental complexity can be irrelevant. A vast number of states may justify one action, while a single low-entropy fact may reverse the correct action. The required structure is determined by the task-induced distinctions, observation process, side information, temporal history, loss function, and tolerated error.

2 Conceptual Lineage and Novelty Boundary

The proposed framework is a synthesis and engineering method, not an attempt to rename established information theory or formal methods.

Shannon established entropy, channel capacity, and coding limits [Shannon, 1948a,b]; rate–distortion theory characterized the minimum asymptotic rate for reproduction under a fidelity criterion [Shannon, 1959, Berger, 1971, Cover and Thomas, 2006]. Remote or indirect source coding treats an encoder that observes a noisy proxy rather than the latent source [Dobrushin and Tsybakov, 1962, Wolf and Ziv, 1970, Witsenhausen, 1980]. The observation-conditioned distortion used later is the classical reduction associated with that literature, not a new construction of this paper.

Cybernetics supplies the language of requisite variety and regulation [Ashby, 1956, Conant and Ashby, 1970]. Information-constrained decision theories quantify policy complexity through state–action information or relative entropy [Sims, 2003, Ortega and Braun, 2013, Genewein et al., 2015]. The information bottleneck preserves relevance under compression [Tishby et al., 2000, Strouse and Schwab, 2017]. Networked control derives communication thresholds for stabilization and feedback objectives [Tatikonda and Mitter, 2004, Nair et al., 2004]. Goal-oriented communication and decision-centric learning already place task loss, rather than descriptive reconstruction, inside rate–distortion formulations [Arumugam and Roy, 2021, Stavrou and Kountouris, 2023]. Recent agent-memory work explicitly preserves histories that matter for decisions instead of descriptions [Zou et al., 2026, Colaco and Lahjouji, 2026].

The worst-case layer belongs to the zero-error lineage initiated by Shannon [Shannon, 1956]. Witsenhausen related zero-error source coding with decoder side information to graph coloring [Witsenhausen, 1976]; Körner's graph entropy gives a distribution-sensitive asymptotic quantity [Körner, 1973], and later work systematizes zero-error and function-computation variants [Körner and Orlitsky, 1998, Orlitsky and Roche, 2001]. The acceptable-action cover introduced below is therefore not presented as a replacement for zero-error information theory. It is the no-side-information finite set-system form needed when a requirement permits multiple actions, a case in which pairwise graph conflicts may fail to capture joint incompatibility.

Programming requirements have their own mature lineage. Hoare logic and refinement mappings connect specifications to implementations [Hoare, 1969, Abadi and Lamport, 1991]. Reactive synthesis asks whether a temporal specification is realizable and constructs a program when it is [Pnueli and Rosner, 1989]; bounded synthesis searches implementations by increasing state bounds [Finkbeiner and Schewe, 2013]. Myhill–Nerode theory and its transducer analogues characterize minimal future-distinguishable state structure [Nerode, 1958]. Model-based finite-state conformance testing derives distinguishing test sequences under explicit assumptions about the specification and implementation class [Chow, 1978]. Classical decision-tree theory quantifies the minimum number or cost of adaptive input queries under an explicit query model; constructing optimal binary decision trees is computationally hard in general [Hyafil and Rivest, 1976].

The novelty claim is consequently specific. This paper does *not* claim a new general source-coding theorem, a new reactive-synthesis decidability result, or the first generation of tests from state models. The closest decision-conflict antecedent is DeMem: it refines an agent-memory partition online when a certified merge would create decision conflict, establishing a safe forgetting boundary with regret guarantees [Zou et al., 2026]. The present method differs in stage and scope. It operates before coding on an explicit requirements model, audits the entire automation boundary rather than memory alone, and produces feasibility witnesses, lower bounds, capability obligations, and generated conformance tests across input, decision, information, memory, query, tool, temporal, and computational dimensions. The proposed certificate asks, in one artifact: whether inputs make the requirement realizable; what functional distinctions and memory states are unavoidable; what

average coupling is required; which queries, outputs, and tools must exist; what computational obligations remain; and which tests witness every modeled boundary. This pre-coding requirements-to-lower-bounds-to-tests pipeline is the paper's principal delta from adjacent rate–distortion, online memory-compression, synthesis, and conformance-testing work.

Table 1: Nearest lineages and the narrower contribution of the present framework.

Lineage	Established object	Role in this paper
Indirect rate–distortion	Minimum average rate from noisy observations under latent-source distortion	Supplies the surrogate distortion and probabilistic lower bound
Zero-error graph theory	Exact distinguishability, chromatic growth, graph entropy, decoder side information	Supplies special-case exact and asymptotic structure; generalized here to permissible-action set systems
Reactive synthesis	Realizability and construction from temporal specifications	Supplies the sequential realizability context
Decision-tree/query complexity	Minimum adaptive observations under a declared query primitive and cost model	Supplies workflow-depth and clarification-cost bounds
Model-based conformance testing	Complete tests under finite-state and state-bound assumptions	Supplies post-code conformance mechanisms
DeMem decision-centric memory	Online, single-capability memory refinement upon certified decision conflict, with a safe forgetting boundary and regret guarantees	Closest conflict-witness antecedent; this paper moves certification pre-coding and spans the full automation boundary with generated tests
Present framework	—	Compiles requirements across all of the above into a single structural and information adequacy certificate

3 The Environment–Automation Boundary

3.1 Variables, fixed law, and architecture

Let $E \in \mathcal{E}$ denote the environment state relevant to one task instance. The automation need not observe E directly. It receives an observation

$$X \sim p(x \mid e),$$

where $X \in \mathcal{X}$ may be a sensor record, prompt, database row, image, event bundle, form, or other available evidence. Unless explicitly varied, the joint law $p(e, x) = p(e)p(x \mid e)$ is fixed and given throughout each optimization problem.

An encoder maps X to an internal or transmitted representation $Z \in \mathcal{Z}$ according to $p(z \mid x)$. A

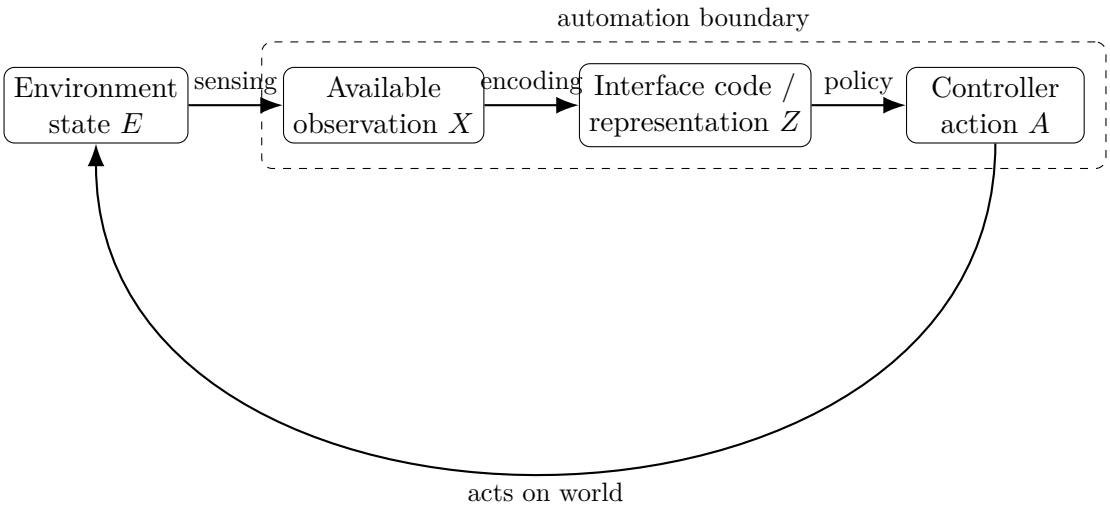

Figure 1: The environment–automation chain. Sensing determines what is available, representation determines what is retained or transferred, and policy determines how retained distinctions affect action.

controller maps Z to an action $A \in \mathcal{A}$ according to $p(a \mid z)$. The memoryless chain is

$$E \longrightarrow X \longrightarrow Z \longrightarrow A.$$

The representation Z is interpreted broadly. It may be a network message, prompt token sequence, retrieved context, typed tool output, workflow state, latent vector, agent memory, or human-authored plan. Mutual information $\mathrm{I}(X; Z)$ measures statistical dependence retained across the boundary; it is not identical to raw byte count, model size, or computational effort.

3.2 Operational distortion

Let $\ell(e, a)$ be task loss. Define direct-access optimal loss

$$\ell^*(e) = \min_{a' \in \mathcal{A}} \ell(e, a')$$

and nonnegative operational regret

$$d(e, a) = \ell(e, a) - \ell^*(e) \geq 0. \tag{1}$$

A tolerance $D \geq 0$ specifies acceptable expected distortion,

$$\mathbb{E}[d(E, A)] \leq D.$$

Other nonnegative task-specific distortions can be used directly. The choice is normative: the theory exposes the information consequence of a loss function but does not decide which harms are acceptable. Hard permission sets $C(e)$ may be audited separately, as in Section 4, or represented by prohibitive loss when an extended-valued distortion model is appropriate.

For hard software invariants, an average criterion may be inappropriate. We will therefore distinguish expected distortion from pointwise or worst-case tolerance. Examples of pointwise requirements include "never expose another tenant's data," "never charge twice," and "always preserve an audit record."

3.3 One-shot lower bound versus operational coding rate

The mutual-information statements below are valid for one random task instance. If a one-shot implementation communicates a prefix-free message M and then produces A, data processing and source-coding inequalities imply

$$\mathbb{E}[L(M)] \geq H(M) \geq \mathrm{I}(X;M) \geq \mathrm{I}(X;A),$$

with the analogous conditional statement when encoder and decoder share randomness independent of X. Thus $\mathrm{I}(X;A)$ is a genuine one-shot lower bound on expected communication length under the stated coding model.

Equality with an operational rate in bits per task instance is a different statement. The classical rate–distortion theorem supplies that interpretation asymptotically for independent repetitions, block coding, and average distortion. In a genuinely one-shot setting, attainable message lengths generally exceed mutual information; functional-representation and one-shot channel-simulation results provide explicit overhead bounds rather than exact equality [Li and Gamal, 2018]. The paper therefore uses "information coupling" for the one-shot quantity and reserves "achievable rate" for a specified operational coding regime.

4 Structural Feasibility Before Information Rate

The rate question is secondary if the required behavior cannot be a function of the proposed inputs. This section begins with finite deterministic specifications; measurable or infinite extensions require corresponding selection conditions.

Let $\mathcal{E}$, $\mathcal{X}$, and $\mathcal{A}$ be finite. A deterministic observation map $s : \mathcal{E} \to \mathcal{X}$ exposes inputs to the automation. Each state e has a nonempty set $C(e) \subseteq \mathcal{A}$ of actions permitted by the specification. A deterministic implementation $\pi : \mathcal{X} \to \mathcal{A}$ is exact when

$$\pi(s(e)) \in C(e) \qquad \forall e \in \mathcal{E}. \tag{2}$$

Theorem 1 (Input-fiber feasibility)**.** *An exact deterministic implementation satisfying Eq. (2) exists if and only if*

$$\boxed{\bigcap_{e:\, s(e)=x} C(e) \neq \varnothing \quad \text{for every } x \in s(\mathcal{E}).} \tag{3}$$

Proof. If π is feasible, the one action $\pi(x)$ must be permissible for every state in the fiber $s^{-1}(x)$, so it lies in the intersection. Conversely, if every intersection is nonempty, select one a_x from each and set $\pi(x) = a_x$. □

Remark 1 (Stochastic observation support)**.** *For a finite stochastic sensor $p(x \mid e)$, the zero-error analogue replaces $s^{-1}(x)$ by the support fiber $\{e : p(e,x) > 0\}$. A deterministic policy based on x exists if and only if the permissible-action intersection over every positive-probability support fiber is nonempty. The expected-loss case is treated by the sensor floor in Eq. (22).*

An empty fiber intersection is a constructive impossibility witness. If two states produce the same exposed input but have disjoint permissible-action sets, no downstream model, prompt, workflow, or code using only that input can satisfy both.

Corollary 1 (Functional factorization)**.** *If every state has one required action $g : \mathcal{E} \to \mathcal{A}$, exact automation is feasible if and only if*

$$s(e_1) = s(e_2) \Longrightarrow g(e_1) = g(e_2), \tag{4}$$

or equivalently if and only if there exists π such that

$$\boxed{g = \pi \circ s.} \tag{5}$$

The required functionality must factor through the available inputs. This is the simplest exact structural requirement for automation.

4.1 The coarsest exact task representation

Define task equivalence

$$e_1 \sim_T e_2 \quad \Longleftrightarrow \quad g(e_1) = g(e_2). \tag{6}$$

Let $q : \mathcal{E} \to \mathcal{E}/\!\sim_T$ be the quotient map.

Theorem 2 (Minimal exact task quotient)**.** *The quotient $\mathcal{E}/\!\sim_T$ is sufficient for exact execution of g. Any representation $z : \mathcal{E} \to \mathcal{Z}$ from which g can be recovered must refine this quotient:*

$$z(e_1) = z(e_2) \Longrightarrow g(e_1) = g(e_2). \tag{7}$$

Consequently,

$$|\mathcal{Z}| \geq |\operatorname{im} g|. \tag{8}$$

Proof. Because g is constant on every equivalence class, there is a well-defined $\bar{g}$ such that $g = \bar{g} \circ q$. If $g = \rho \circ z$ and $z(e_1) = z(e_2)$, then $g(e_1) = \rho(z(e_1)) = \rho(z(e_2)) = g(e_2)$. Distinct task classes therefore require distinct representation symbols. □

The quotient specifies *which* distinctions must survive. It does not impose a code architecture, number of branches, model size, or line count.

4.2 The minimal nonconstant example

Let $\mathcal{E} = \{e_0, e_1\}$ with $g(e_0) = a_0$, $g(e_1) = a_1$, and $a_0 \neq a_1$. If $s(e_0) = s(e_1)$, exact automation is impossible. If $s(e_0) = x_0 \neq x_1 = s(e_1)$, the policy $\pi(x_i) = a_i$ succeeds. The smallest nonconstant one-shot automation therefore requires one preserved binary distinction. Under a balanced distribution this has one bit of entropy; under an imbalanced distribution its asymptotic average description cost is the binary entropy of the required action.

4.3 Minimum-cost candidate input basis

Pre-coding requirements often name a finite collection of candidate fields rather than permit an arbitrary sensor. Let $v_j : \mathcal{E} \to \mathcal{V}_j$, $j = 1, \ldots, m$, be candidate fields with nonnegative costs c_j. For $J \subseteq \{1, \ldots, m\}$, define

$$s_J(e) = (v_j(e))_{j \in J}.$$

For a unique required action g, let

$$\mathcal{P} = \{\{e, e'\} : g(e) \neq g(e')\}$$

be the set of action-conflicting pairs. A field j separates pair $\{e, e'\}$ when $v_j(e) \neq v_j(e')$.

Proposition 1 (Candidate-field separation)**.** *A selected field set J supports exact execution if and only if every pair in $\mathcal{P}$ is separated by at least one field in J. The minimum-cost candidate input basis is the binary program*

$$\min_{y\in\{0,1\}^m} \quad \sum_{j=1}^{m} c_j y_j \tag{9}$$

$$s.t. \quad \sum_{j:\, v_j(e)\neq v_j(e')} y_j \geq 1, \qquad \forall\{e,e'\}\in\mathcal{P}. \tag{10}$$

Proof. By Corollary 1, exact execution fails exactly when some conflicting pair remains in the same selected-field fiber. The constraints exclude every such pair, and the objective selects the least-cost exclusion set. □

This is a weighted hitting-set formulation and is NP-hard in general by the classical set-cover lineage [Karp, 1972]. For multiple permissible actions, pairwise conflict is not sufficient: a collection of states may have pairwise-overlapping action sets but an empty joint intersection. The exact extension uses minimal infeasible hyperedges $U\subseteq\mathcal{E}$ satisfying $\cap_{e\in U}C(e)=\varnothing$; selected fields must split every such hyperedge.

4.4 Adaptive input acquisition and query complexity

The preceding input basis is nonadaptive: it asks which fields must be available as a set. Agents and workflows can often inspect fields or ask questions sequentially and stop once one action is permissible for every state still consistent with the answers. This creates a second, operationally distinct lower bound. Combine hard permissions with a pointwise loss tolerance by defining

$$C_D(e) = C(e)\cap\{a\in\mathcal{A} : d(e,a)\leq D\}. \tag{11}$$

Let $S\subseteq\mathcal{E}$ be the current candidate-state set, let J be the remaining queryable fields, and let $S_{j,y}=\{e\in S : v_j(e)=y\}$. A leaf is valid at tolerance D when

$$\bigcap_{e\in S} C_D(e)\neq\varnothing. \tag{12}$$

For query costs $c_j\geq 0$, define the exact worst-case recurrence

$$Q_D^{\mathrm{wc}}(S,J) = \begin{cases} 0, & \bigcap_{e\in S} C_D(e)\neq\varnothing, \\ \min_{j\in J}\left[c_j + \max_{y:S_{j,y}\neq\varnothing} Q_D^{\mathrm{wc}}(S_{j,y}, J\setminus\{j\})\right], & \text{otherwise,} \end{cases} \tag{13}$$

where queries that do not split S may be omitted without loss, and the value is $+\infty$ if no valid continuation exists. Given a fixed full-support state law, with $P(S_{j,y}\mid S)=P(E\in S_{j,y}\mid E\in S)$, the minimum expected query cost obeys

$$Q_D^{\mathrm{avg}}(S,J) = \begin{cases} 0, & \bigcap_{e\in S} C_D(e)\neq\varnothing, \\ \min_{j\in J}\left[c_j + \sum_{y:S_{j,y}\neq\varnothing} P(S_{j,y}\mid S) Q_D^{\mathrm{avg}}(S_{j,y}, J\setminus\{j\})\right], & \text{otherwise.} \end{cases} \tag{14}$$

Proposition 2 (Optimal adaptive query bound)**.** *Under the declared primitive model—each field v_j may be queried at most once at cost c_j, and leaves select one common permissible action—Eqs. (13) and (14) equal the minimum worst-case and expected costs of any exact deterministic adaptive query tree.*

Proof. At a valid leaf no further query is required. Otherwise every admissible tree chooses some first field j and then contains a valid subtree for each possible answer y. Worst-case cost is the first-query cost plus the maximum subtree cost; expected cost is the first-query cost plus the answer-probability-weighted subtree cost. Minimizing over the first query and applying induction on $|J|$ proves both recurrences. □

These are bounds on a specified observation protocol, not intrinsic line-of-code counts. Changing the primitive—for example, permitting one trusted derived predicate, a database join, or a parallel batch query—changes the decision-tree problem. Computing optimal decision trees is NP-hard in general [Hyafil and Rivest, 1976], so large certificates should report exact dynamic-program or integer-program solutions only when tractable and otherwise provide bounds or heuristic trees.

4.5 Worst-case tolerance and acceptable-action covers

Using the acceptable sets $C_D(e)$ from Eq. (11), define

$$S_a(D) = \{e : a \in C_D(e)\}. \tag{15}$$

Assume no decoder side information and a deterministic representation whose symbols decode to actions.

Theorem 3 (Minimum robust action-labeled representation)**.** *The minimum number of representation symbols needed to select $A \in C_D(e)$ for every state is*

$$K_D^{\mathrm{wc}} = \min_{B \subseteq \mathcal{A}} |B| \quad s.t. \quad \mathcal{E} \subseteq \bigcup_{a \in B} S_a(D). \tag{16}$$

Thus a fixed-length action-label code requires at least

$$\lceil \log_2 K_D^{\mathrm{wc}} \rceil \tag{17}$$

bits in the worst case.

Proof. Any deterministic representation and decoder use a set B of decoded actions. Feasibility requires each state to lie in at least one $S_a(D)$ for $a \in B$, so B is a cover. Conversely, from any such cover assign each state one covering action, encode that action's index, and decode it identically. □

The optimization is set cover and is NP-hard [Karp, 1972]. Small finite certificates can solve it exactly by integer programming; larger audits should report lower and upper bounds or use approximation methods such as greedy covering [Chvátal, 1979]. Cardinality is necessary but not sufficient: an implementation may have many nominal branches yet place the wrong states together.

4.6 Relation to zero-error graphs and graph entropy

The graph-theoretic special case becomes exact when the controller has side information and the required output is unique. Let an encoder observe X, let the decoder additionally observe S, and let $g(x, s)$ be the required action. Define a graph G on $\mathcal{X}$ by connecting x and x' when there exists an s with both (x, s) and (x', s) possible and

$$g(x, s) \neq g(x', s).$$

A zero-error one-shot encoder is feasible exactly when its code labels form a proper coloring of G; the minimum signal alphabet is $\chi(G)$ [Witsenhausen, 1976]. With trivial side information, G is complete multipartite across required-action classes and $\chi(G) = |\operatorname{im} g|$. Under repeated source instances, chromatic growth and graph-entropy quantities provide worst-case and distribution-sensitive asymptotic analogues [Körner, 1973, Körner and Orlitsky, 1998, Orlitsky and Roche, 2001].

For general permissible-action sets $C(e)$, a pairwise conflict graph can be inexact. For example, $\{a, b\}$, $\{b, c\}$, and $\{a, c\}$ intersect pairwise but have empty three-way intersection. The corresponding exact object is a hypergraph or set system, which is why Theorem 3 is stated as a cover rather than as a universal chromatic-number identity.

5 The Requisite Information Curve

Definition 1 (Requisite information for automation)**.** *For the fixed joint law $p(e, x)$ and memoryless architecture $E \to X \to Z \to A$, define*

$$R_{\text{aut}}(D) = \inf_{p(z|x),\, p(a|z)} \mathrm{I}(X; Z) \quad s.t. \quad \mathbb{E}[d(E, A)] \leq D. \tag{18}$$

If the distortion target is infeasible, set $R_{\text{aut}}(D) = +\infty$.

This quantity asks how much statistical dependence on the available observation must survive the interface. It is not the raw length of a chosen encoding. Redundancy, protocol structure, shared context, side information, and temporal correlation can make physical messages longer or shorter than naive field counts suggest.

5.1 Classical indirect-distortion reduction

Define

$$\bar{d}(x, a) = \mathbb{E}[d(E, a) \mid X = x]. \tag{19}$$

For every policy based on X,

$$\mathbb{E}[d(E, A)] = \mathbb{E}[\bar{d}(X, A)]. \tag{20}$$

This is the standard reduction of an indirect or remote rate–distortion problem to a direct one over the observation alphabet [Witsenhausen, 1980], building on earlier remote-source formulations [Dobrushin and Tsybakov, 1962, Wolf and Ziv, 1970].

Theorem 4 (Exact task-action reduction)**.** *For the memoryless single-task architecture in Eq. (18),*

$$\boxed{R_{\text{aut}}(D) = \inf_{p(a|x):\, \mathbb{E}[\bar{d}(X,A)] \leq D} \mathrm{I}(X; A).} \tag{21}$$

The infimum is taken with the source law $p(x)$ and surrogate distortion $\bar{d}$ fixed.

Proof. For any feasible architecture, $X \to Z \to A$ is a Markov chain, so data processing gives $\mathrm{I}(X;Z) \geq \mathrm{I}(X;A)$. The architecture induces a feasible channel

$$p(a \mid x) = \sum_z p(z \mid x) p(a \mid z).$$

Therefore every architecture has objective at least the right-hand side of Eq. (21). Conversely, for any feasible $q(a \mid x)$ choose $Z = A$, set $p(z \mid x) = q(a = z \mid x)$, and use the identity decoder. Then $\mathrm{I}(X;Z) = \mathrm{I}(X;A)$ and the same distortion is achieved. □

Remark 2 (Interpretation)**.** *The reduction is mathematically elementary. It does not state that practical systems should literally transmit an action. A reusable representation, explanation, audit trail, database, or memory designed before the task is known may need substantially more information. Its role is to identify the canonical single-task lower bound against which such systems can be compared.*

Under independent repeated task instances and average distortion, the right-hand side is an ordinary Shannon rate–distortion function with reproduction alphabet $\mathcal{A}$. In one shot, it remains a coupling and message-length lower bound as discussed in Section 3.3; exact one-shot achievability need not equal the mutual information.

5.2 Endpoints and actionable entropy

Define the sensor-limited Bayes floor

$$D_{\min}(X) = \min_{p(a|x)} \mathbb{E}[\bar{d}(X, A)] = \mathbb{E}\Big[\min_a \bar{d}(X, a)\Big] \tag{22}$$

and the best observation-independent loss

$$D_0 = \min_a \mathbb{E}[d(E, a)]. \tag{23}$$

Then $R_{\mathrm{aut}}(D) = +\infty$ for $D < D_{\min}(X)$ and $R_{\mathrm{aut}}(D) = 0$ for $D \geq D_0$. Under finite-alphabet assumptions the curve is nonincreasing and convex.

Corollary 2 (Actionable entropy at zero distortion)**.** *Suppose $X = E$ and every state has a unique zero-regret action $A^* = g(E)$. Then*

$$R_{\mathrm{aut}}(0) = \mathrm{H}(A^*). \tag{24}$$

Proof. Zero distortion forces $A = A^*$ almost surely. Since A^* is a deterministic function of E, $\mathrm{I}(E; A^*) = \mathrm{H}(A^*)$. □

This is an entropy form of requisite variety. A uniform exact task with K required actions has $\log_2 K$ bits per independent instance asymptotically. With imbalanced actions, the average requirement can be much smaller than the fixed-length naming bound $\lceil \log_2 K \rceil$. Both can be dramatically smaller than $H(E)$.

5.3 A tightened nuisance-invariance result

The statement that nuisance complexity does not matter requires a conditional irrelevance hypothesis, not merely absence from the written loss function.

Proposition 3 (Nuisance invariance)**.** *Let $E = (T, N)$, let $X = (X_T, X_N)$, and suppose the loss depends only on (T, A). Assume*

$$\mathbb{E}[d(T, a) \mid X_T, X_N] = \mathbb{E}[d(T, a) \mid X_T] \quad \forall a \in \mathcal{A} \tag{25}$$

almost surely. Then the task-action rate–distortion curve computed from (X_T, X_N) equals the curve computed from X_T alone. Consequently, across any family of nuisance distributions that leaves $p(t, x_T)$ and Eq. (25) unchanged, increasing the entropy or dimension of N does not change $R_{\text{aut}}(D)$.

Proof. By Eq. (25), the surrogate distortion depends only on X_T. Any channel $p(a \mid x_T, x_N)$ induces $q(a \mid x_T)$ by averaging over $X_N \mid X_T$, with identical expected distortion. Data processing gives

$$\mathrm{I}(X_T, X_N; A) \geq \mathrm{I}(X_T; A),$$

so the full-observation curve cannot be lower than the X_T curve. The reverse inequality follows by ignoring X_N. □

The world may become descriptively more complex without becoming harder to automate. Conversely, one low-dimensional variable can contain nearly all action consequence.

5.4 Irreversibility of upstream loss

Proposition 4 (Sensor impossibility condition)**.** *Let*

$$R_E(D) = \inf_{p(a|e):\, \mathbb{E}[d(E,A)] \leq D} \mathrm{I}(E; A) \tag{26}$$

be the task-action curve under direct environmental access. Any automation based only on X that achieves distortion D must satisfy

$$R_E(D) \leq \mathrm{I}(E; X). \tag{27}$$

Proof. For $E \to X \to A$, data processing gives $\mathrm{I}(E; A) \leq \mathrm{I}(E; X)$, while the definition of $R_E(D)$ gives $R_E(D) \leq \mathrm{I}(E; A)$. □

The condition is necessary but not sufficient. A sensor may carry many bits about irrelevant properties while omitting one rare feature that reverses the action. The exact feasibility statement is the Bayes floor: if $D < D_{\min}(X)$, no downstream computation, training, or planning can achieve the target without another information source.

Proposition 5 (Serial bottleneck law)**.** *For any serial chain*

$$X \to Z_1 \to Z_2 \to \cdots \to Z_m \to A$$

that achieves distortion D,

$$\mathrm{I}(X; Z_j) \geq \mathrm{I}(X; A) \geq R_{\text{aut}}(D) \quad \forall j. \tag{28}$$

Proof. For every j, marginalization leaves the Markov chain $X \to Z_j \to A$. Apply data processing and Theorem 4. □

Every lossy stage must preserve sufficient task information. A rich sensor does not rescue an inadequate database projection. A long prompt does not rescue a summarizer that removed the decisive exception. A powerful planner does not rescue a memory system that merged histories requiring different future actions.

6 Additional Fields, Reusable Representations, and Dynamics

6.1 Value of an added field

Let V be a proposed additional field. If

$$\mathbb{E}[d(E,a) \mid X,V] = \mathbb{E}[d(E,a) \mid X] \quad \forall a \tag{29}$$

almost surely, then the field cannot lower either $D_{\min}$ or the task-action rate–distortion curve. This follows directly from Proposition 3 with $X_T = X$ and $X_N = V$. More modalities, columns, tokens, or sensor channels are therefore not automatically beneficial. Added data has value only when it changes conditional action consequences or reduces uncertainty at a later task-relevant stage.

6.2 Reusable representations and future tasks

The exact reduction assumes that the task is known when Z is designed. Databases, foundation-model context, and long-term memories are often encoded before the future query is known. Let Q be a downstream task revealed after Z is formed, with fixed joint law $p(e,x,q)$. Define

$$R_{\text{reuse}}(D) = \inf_{p(z|x),\, p(a|z,q)} \mathrm{I}(X;Z) \quad \text{s.t.} \quad \mathbb{E}[d_Q(E,A)] \le D. \tag{30}$$

In general, $R_{\text{reuse}}(D)$ exceeds the rate for one known task because Z must preserve distinctions that may matter under multiple future objectives. This is where information-bottleneck reasoning, provenance, and decision-centric memory become central [Tishby et al., 2000, Zou et al., 2026, Colaco and Lahjouji, 2026].

If both encoder and controller know side information S, the incremental coupling is naturally conditional, $\mathrm{I}(X;Z \mid S)$. If only the decoder knows S, Wyner–Ziv and zero-error side-information formulations become relevant; one may not simply subtract the context as though it were shared at every stage.

6.3 Dynamic automation

Feedback systems generate new task-relevant uncertainty over time. A one-shot quantity is then insufficient. For causal sequences, directed information

$$\mathrm{I}(X^n \to Z^n) = \sum_{t=1}^{n} \mathrm{I}(X^t; Z_t \mid Z^{t-1}) \tag{31}$$

replaces ordinary mutual information in causal rate–distortion formulations. Networked-control results make the point operational: unstable dynamics can require a positive ongoing data rate for stabilization [Tatikonda and Mitter, 2004, Nair et al., 2004]. For a scalar unstable linear multiplier λ, classical thresholds contain $\log_2 |\lambda|$ bits per step under the relevant assumptions. Delay, ordering, and burstiness matter independently of average rate; information arriving after the action deadline has no value for that decision.

7 Canonical Information-Theoretic Examples

7.1 Balanced binary action

Let $E = X \sim \text{Bernoulli}(1/2)$, $A \in \{0,1\}$, and $d(e,a) = \mathbf{1}\{e \neq a\}$. The classical binary rate–distortion function gives

$$R_{\text{aut}}(D) = 1 - h_2(D), \qquad 0 \le D \le \tfrac{1}{2}, \tag{32}$$

where $h_2(D) = -D \log_2 D - (1 - D) \log_2(1 - D)$. Zero error requires one bit per independent task instance asymptotically. Allowing 10% error lowers the rate to approximately 0.531 bits; chance-level error requires no information.

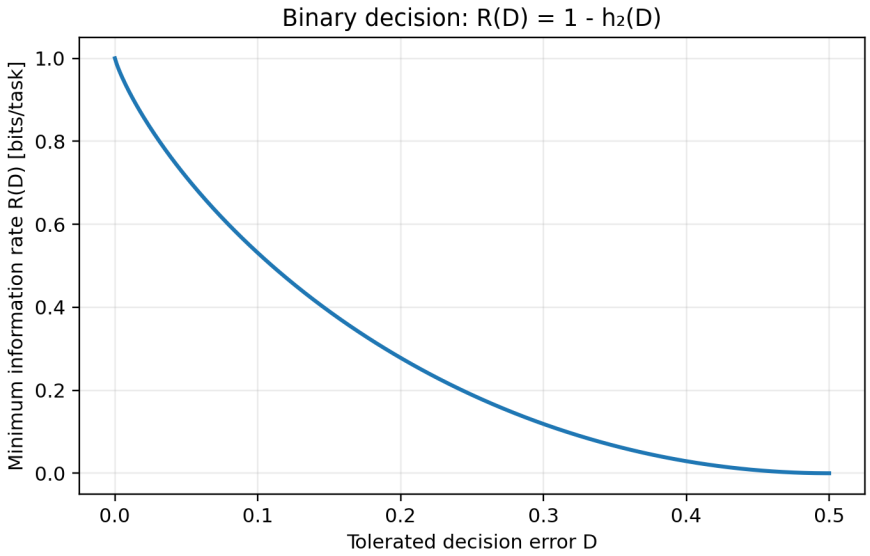

Figure 2: The requisite information curve for a balanced binary decision. Tolerated error removes the need to preserve some distinctions.

7.2 Uniform multiway triage

For a uniform K-class task with Hamming loss,

$$R(D) = \log_2 K - h_2(D) - D \log_2(K - 1), \quad 0 \le D \le 1 - \frac{1}{K}. \tag{33}$$

At zero error the rate is $\log_2 K$. Under unequal frequencies it becomes the entropy of the required action, not the logarithm of the number of descriptive fields or workflow branches.

7.3 Gaussian regulation

Let $E = X \sim \mathcal{N}(0, \sigma^2)$, $A \in \mathbb{R}$, and $d(e, a) = (e - a)^2$. Then

$$R_{\text{aut}}(D) = \begin{cases} \frac{1}{2} \log_2\left(\frac{\sigma^2}{D}\right), & 0 < D < \sigma^2, \\ 0, & D \ge \sigma^2. \end{cases} \tag{34}$$

Halving root-mean-square error quarters D and costs one additional bit per sample asymptotically.

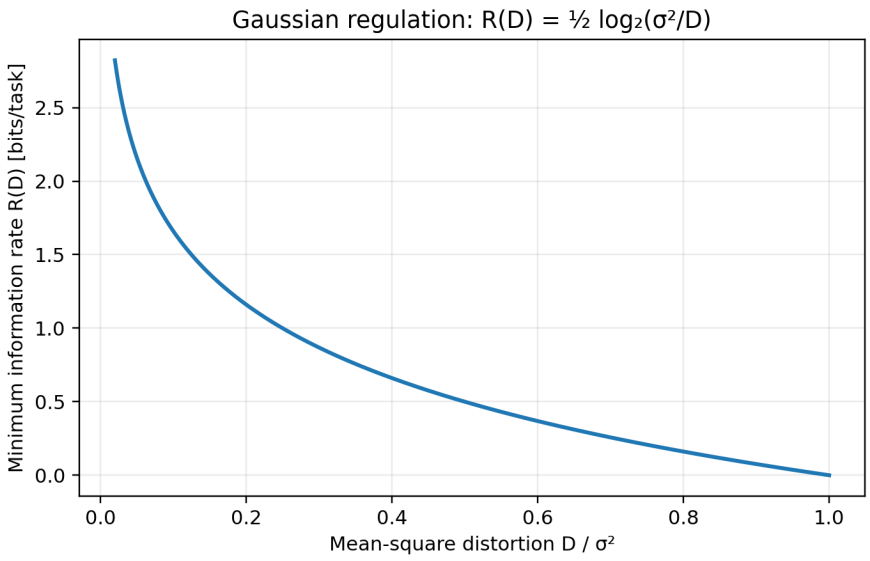

Figure 3: The Gaussian rate–distortion law interpreted as a regulation limit. The horizontal axis is normalized by source variance.

7.4 Noisy binary sensing

Let $E \sim \text{Bernoulli}(1/2)$ and $X = E \oplus N$, where $N \sim \text{Bernoulli}(q)$ and $0 \le q < 1/2$. Under Hamming task loss, the best possible action based on X has error q, so $D < q$ is impossible. The surrogate distortion is

$$\bar{d}(x, a) = q + (1 - 2q)\mathbf{1}\{a \neq x\},$$

and for $q \le D \le 1/2$,

$$R_{\text{aut}}(D) = 1 - h_2\left(\frac{D - q}{1 - 2q}\right). \tag{35}$$

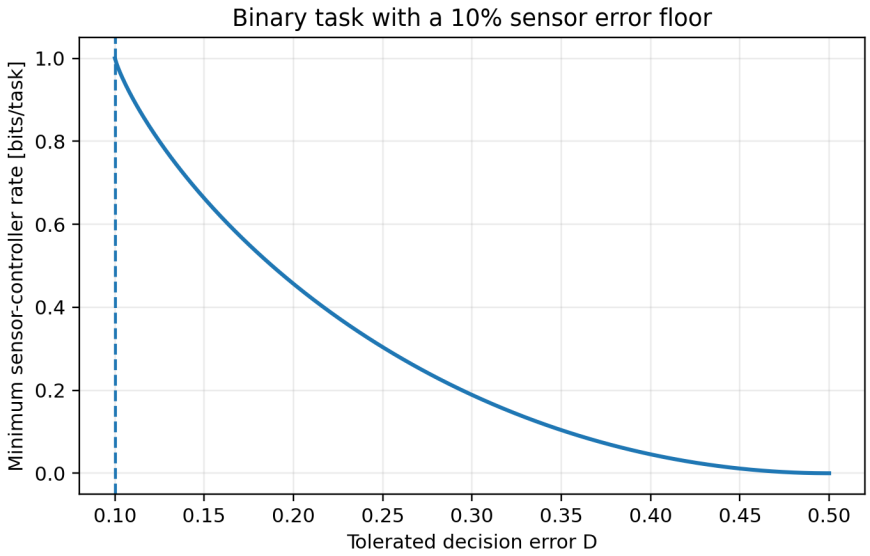

Figure 4: A 10% sensor error creates an irreducible floor. No downstream model can enter the region left of the dashed line without another information source.

These examples separate three quantities that are often conflated: task-class cardinality, average action entropy, and a lossy rate–distortion curve. The structural and probabilistic certificates should report them separately.

8 Extension to Programming Requirements

8.1 Specifications as permitted behavior

A one-shot software specification can be represented as a relation

$$\mathcal{S} \subseteq \mathcal{I} \times \mathcal{O}, \tag{36}$$

where $\mathcal{I}$ is the externally relevant input domain and $\mathcal{O}$ is the observable output or effect domain. Define $C(i) = \{o : (i, o) \in \mathcal{S}\}$. A deterministic program $P : \mathcal{I} \to \mathcal{O}$ satisfies the specification when $P(i) \in C(i)$ throughout the specified domain.

This is consistent with refinement-based correctness, but the present question is prior to a full proof: what must any refining implementation be capable of distinguishing and doing? The fiber criterion is the finite, one-shot edge of realizability. For temporal and interactive specifications, reactive synthesis asks the richer existence question over input/output traces [Pnueli and Rosner, 1989]; bounded synthesis explicitly searches by increasing controller-state limits [Finkbeiner and Schewe, 2013].

8.2 Seven independent adequacy dimensions

A single scalar called "code complexity" is usually misleading. A requirements certificate should separate at least seven dimensions.

1. **Input sufficiency.** Exposed inputs must split every state collection whose permissible-action intersection is empty. A collision gives an impossibility witness.

2. **Output expressiveness.** For every supported state e, $C(e) \cap \mathcal{A}_{\text{impl}} \neq \varnothing$. Binary success/failure cannot implement a requirement that needs success, denial, clarification, escalation, and rollback.

3. **Decision structure.** The implementation must preserve all required task boundaries. Having at least K_D^{wc} nominal branches is not enough if the wrong scenarios are merged.

4. **Memory structure.** If future behavior depends on history, the implementation must preserve future-distinguishable histories, not merely the current request.

5. **Information-acquisition capability.** When initial inputs are insufficient, the system may need to ask, retrieve, inspect, verify, abstain, or escalate rather than force a decision.

6. **Actuator and tool capability.** Every promised external effect must be reachable through an available tool, permission, transaction, compensating action, or human handoff.

7. **Computational and timing sufficiency.** Even structurally adequate information may be unusable if the required mapping cannot be computed within time, space, query, cost, or deadline constraints. Information theory alone does not supply general algorithmic lower bounds.

This decomposition clarifies the phrase "the implementation is not complex enough." It may mean that the input interface is impossible, the representation merges cases, the state machine forgets a needed fact, a required action is absent, the program lacks a query path, or the computation cannot finish on time. Those are different defects with different remedies.

8.3 Sequential memory lower bound

Let h be an input/output history and let F_h denote the residual required behavior after h: the mapping or relation from every permitted future continuation to the corresponding permissible future trace. Define

$$h_1 \sim_F h_2 \quad \Longleftrightarrow \quad F_{h_1} = F_{h_2}. \tag{37}$$

For deterministic finite-state specifications, Myhill–Nerode-style reasoning implies that each future-distinguishable class requires a distinct reachable memory state [Nerode, 1958]. If

$$N_F = |\mathcal{H}/\!\sim_F|, \tag{38}$$

then any exact deterministic finite-state realization requires at least N_F reachable states and at least $\lceil \log_2 N_F \rceil$ fixed-length state bits. For nondeterministic, probabilistic, partially observed, or infinite-state systems, the appropriate simulation, bisimulation, belief-state, or synthesis notion must replace this elementary count.

A common agent example is idempotency. Before a side effect is committed and after it is committed, the current retry request can be identical, but the required future traces differ: "apply and notify" versus "return the prior result without repeating the effect." Those histories therefore require at least two distinguishable memory states, whether represented internally or supplied by an external idempotency ledger.

8.4 Why sampled tests cannot establish unrestricted conformance

Proposition 6 (Sample-test insufficiency)**.** *Let $g : \mathcal{I} \to \mathcal{O}$ be a unique-output specification with $|\mathcal{O}| \geq 2$, and let $T \subsetneq \mathcal{I}$ be a proper test subset. If the implementation class contains all functions $\mathcal{I} \to \mathcal{O}$, there exists a program that passes every test in T and violates the specification outside T.*

Proof. Choose $i_0 \notin T$ and an output $o \neq g(i_0)$. Define $P(i) = g(i)$ for $i \in T$, set $P(i_0) = o$, and define P arbitrarily elsewhere. □

This proposition does not imply that internal complexity should be maximized. If tests exhaust a finite formal domain and all trace obligations, a simple passing implementation is acceptable. The problem is that ordinary test suites rarely provide such completeness. Model-based conformance methods can be complete under stronger assumptions, such as deterministic finite-state specifications, reset or access sequences, and an upper bound on implementation states [Chow, 1978]. A requisite certificate contributes the missing precondition: an explicit model of the distinctions and state bounds the tests must cover.

8.5 The requisite complexity certificate

The certificate is a structured record rather than a single score. For a target tolerance D, it should include

$$\mathsf{RCC}_D = (\mathsf{Feas}, J_{\min}, K_D^{\mathrm{wc}}, R_{\mathrm{aut}}(D), N_F, Q_D^{\mathrm{wc}}, Q_D^{\mathrm{avg}}, \mathcal{Q}_{\mathrm{req}}, \mathcal{A}_{\mathrm{req}}, \mathcal{T}_{\mathrm{req}}, \mathcal{B}_{\mathrm{comp}}), \tag{39}$$

where the components denote structural feasibility, a minimum or bounded candidate-input basis, robust action-cover size, average information coupling, future-distinguishable memory classes, worst-case and expected adaptive query costs, required information-acquisition actions, required outputs, required tools or actuators, and computational-resource obligations. Some components are numbers; others are capability sets, witnesses, or unresolved proof obligations.

A pre-coding workflow is:

1. Formalize scenarios or histories, candidate inputs, permissible outputs, hard invariants, probabilistic losses, side information, and timing assumptions.
2. Search for fiber conflicts. If found, add sensing, interaction, side information, escalation, or narrow the promised domain.
3. Compute exact or bounded candidate-field, adaptive-query, acceptable-action, rate–distortion, and memory quantities.
4. List required output variants, queries, tools, permissions, compensations, timers, and computational constraints.
5. Generate separation tests across every required boundary, invariance tests across nuisance variation, missing-information and escalation tests, and sequential trace tests.
6. After coding, abstract reachable implementation behavior and check refinement, required state distinguishability, and absence of unsafe collisions.

8.6 Post-code structural conformance

Let $z(i)$ be an observable or instrumented implementation representation before final output. In the unique-output one-shot case, a necessary structural condition is

$$z(i_1) = z(i_2) \Longrightarrow g(i_1) = g(i_2). \tag{40}$$

For permissible-action relations, every implementation fiber must retain a common permissible output. If z is inaccessible, black-box tests can only probe the induced behavior, not certify internal state cardinality without an implementation-class assumption.

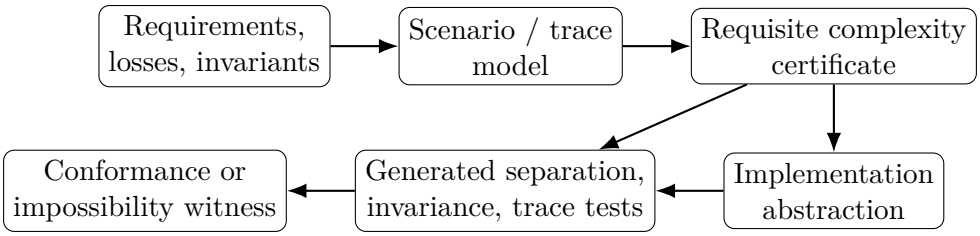

Figure 5: Requirements-derived structural QA. The certificate supplies lower bounds and witness obligations; it does not replace refinement, synthesis, model checking, or empirical generalization testing.

For sequential systems, refinement mappings connect implementation states to specification states [Abadi and Lamport, 1991]. Reachability matters: dead code, unreachable workflow nodes, and dormant prompt instructions do not contribute functional capacity. A system may distinguish more states than required; it may not merge across a required boundary unless a common permissible continuation remains.

The certificate therefore supports two asymmetric conclusions. A conflict witness can *prove insufficiency*. A lower-bound comparison can show that a proposed finite architecture is too small. But showing that the implementation is sufficient requires behavioral refinement, exhaustive finite checking, formal proof, or justified statistical evidence over the relevant domain. Complexity capacity alone never proves the boundaries are correct.

9 Computational Verification and a Programming-Requirements Case Study

The accompanying source bundle contains the Python program, machine-readable results, and generated figure used in this section. The calculations are finite and deterministic; they verify the implementation of the formulas and illustrate the certificate, not the novelty of the underlying coding theorems.

9.1 Closed-form and indirect-reduction checks

A finite-alphabet Blahut–Arimoto implementation was evaluated against the balanced binary, uniform four-way, and noisy binary formulas in Section 7. Across the nondegenerate points checked, the largest absolute rate discrepancy was below 7×10^{-16} bits, consistent with floating-point roundoff. One hundred randomly generated indirect tasks also verified Eq. (20); the maximum direct-versus-surrogate expectation residual was 3.3×10^{-16}, and the sensor floor in Eq. (22) matched brute-force enumeration exactly.

Table 2: Reproducible numerical checks.

Check	Instances / points	Maximum absolute residual
Balanced binary Hamming curve	182	3.3×10^{-16} bits
Uniform four-way Hamming curve	184	4.4×10^{-16} bits
Noisy binary sensor, $q = 0.1$	187	4.4×10^{-16} bits
Random indirect expectation identity	100 tasks	3.3×10^{-16}
Random sensor floor versus brute force	100 tasks	0

9.2 Finite appointment-rescheduling specification

Consider a synthetic but realistic feature specification for an appointment-rescheduling automation. Each scenario has five binary variables:

$$(u, m, c, b, r) = (\text{authorized}, \text{modifiable}, \text{within cutoff}, \text{billing lock}, \text{series scope}).$$

There are $2^5 = 32$ scenarios. The required action is unique and follows Table 3.

Table 3: Appointment feature requirement rules.

Condition	Required action
$\neg u \vee \neg m \vee \neg c$	`DENY`
$u \wedge m \wedge c \wedge b$	`ESCALATE_BILLING`
$u \wedge m \wedge c \wedge \neg b \wedge \neg r$	`RESCHEDULE_SINGLE`
$u \wedge m \wedge c \wedge \neg b \wedge r$	`RESCHEDULE_SERIES`

Under a uniform distribution over the 32 scenarios, the action counts are $(28, 2, 1, 1)$. The structural and probabilistic certificate is summarized in Table 4.

Table 4: Requisite certificate for the appointment feature.

Quantity	Result
Exact task classes, K_0	4
Worst-case fixed-length action label	$\lceil \log_2 4 \rceil = 2$ bits
Actionable entropy, $H(A^*)$	0.731064 bits/task
Optimal worst-case raw-field queries	5
Optimal expected raw-field queries, uniform scenarios	1.9375
Sensor floor with all five fields	0
Sensor floor if `billing_lock` is omitted	$1/16 = 0.0625$
Unsafe input fibers after omission	2
Minimum sufficient subset among the five raw candidate fields	all five fields
Sufficient derived handoff	`eligible_to_modify`, `billing_lock`, `series_scope`
Retry/idempotency memory classes	2 states, hence at least 1 state bit

The raw field result is stage-specific. Upstream logic still needs authorization, modifiability, and cutoff information to construct the trusted predicate `eligible_to_modify`; once constructed, those three variables may be compressed into one decision-relevant handoff field. The framework therefore distinguishes sensing requirements from inter-service representation requirements.

With unit-cost raw-field access, the optimal adaptive tree short-circuits denial by querying u, m, and c in any order, queries b only when all three are true, and queries r only when the case is eligible and not billing-locked. It requires five queries in the worst case but only

$$1 + \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \tfrac{1}{16} = 1.9375$$

on average under the uniform scenario law. This number is a workflow/query bound under a declared primitive model; a trusted derived predicate can change it.

9.3 One missing field creates a nonzero feasibility floor

Suppose the PRD or tool schema exposes (u, m, c, r) but omits billing lock b. Two observed fibers become unsafe:

$$(1,1,1,r=0): \quad \texttt{RESCHEDULE_SINGLE} \text{ versus } \texttt{ESCALATE_BILLING},$$
$$(1,1,1,r=1): \quad \texttt{RESCHEDULE_SERIES} \text{ versus } \texttt{ESCALATE_BILLING}.$$

Each fiber contains equally likely hidden states with incompatible required actions. The best controller must be wrong on one state in each pair, producing the irreducible uniform error

$$D_{\min}(X_{-b}) = \frac{2}{32} = 0.0625.$$

No prompt revision, larger model, chain-of-thought procedure, or additional code operating on the same four fields can reduce the error below that floor.

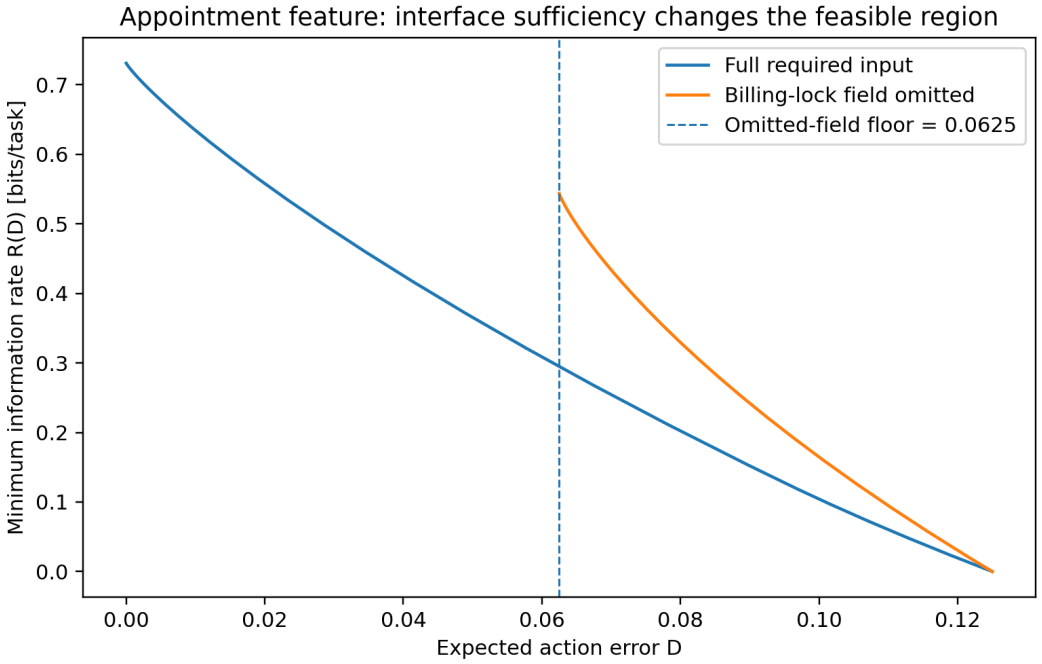

Figure 6: Rate–distortion curves for the appointment specification. Omitting billing lock removes the entire low-error region below $D = 0.0625$; downstream controller quality cannot recover it.

9.4 Why an ordinary passing test suite can miss the defect

A deficient implementation that ignores billing lock was tested on eight conventional examples covering denial reasons, single rescheduling, series rescheduling, and precedence of denial over other conditions. Because none of those tests included an otherwise eligible billing-locked case, the implementation passed 8/8. Across the complete finite scenario set it passed $30/32 = 93.75\%$, failing precisely the two structurally identified cases.

The point is not that 93.75% is universally inadequate; the distribution and harms in a real system determine that judgment. The point is logical: the implementation could never satisfy the exact use case because the proposed interface made the two failed decisions indistinguishable. The certificate supplied the missing-field proof and generated the two boundary tests before code was written.

9.5 Sequential extension: exactly-once effects

Add the requirement that a successful reschedule and its notification occur exactly once under retries. The current retry request can be identical before and after commit, yet the required future action differs:

$$\text{uncommitted} \to \texttt{APPLY_AND_NOTIFY}, \qquad \text{committed} \to \texttt{RETURN_PRIOR_RESULT}.$$

These histories are future-distinguishable, so a stateless implementation is insufficient. At least two reachable memory classes are required, represented either internally or by equivalent external state such as an idempotency record. Single-call tests cannot establish this trace property.

9.6 Interpretation

For this feature, "complex enough" does not mean a minimum number of code branches. It means, at minimum:

- the environment-to-feature boundary exposes or reliably derives every distinction needed to separate the four action classes;
- the output schema can express denial, billing escalation, single rescheduling, and series rescheduling;
- an adaptive raw-field workflow supports a five-query worst case, or the boundary supplies an equivalent trusted derived predicate;
- the workflow retains two retry histories when exactly-once behavior is promised; and
- the acceptance suite includes witnesses for each required separation and invariance.

An implementation meeting those obligations may still be very small. Simplicity is not a defect when the complete required behavior factors through it.

10 A Requisite Information and Complexity Audit

The practical method should report both exact structural results and uncertainty-qualified probabilistic estimates.

10.1 Step 1: Specify behavior and consequence

Enumerate or model the relevant scenarios and histories, permissible actions, hard invariants, losses, abstention or escalation options, timing constraints, and externally observable effects. Separate requirements from implementation choices. For stochastic tasks, identify the population or source law to which the expectation refers.

10.2 Step 2: Declare the observation boundary

Distinguish latent environment state E from exposed input X. Record how fields are generated, which facts are omitted, when they become stale, which side information humans use, and which facts can be acquired interactively. A field available elsewhere in the organization is not available to the automation unless it crosses the audited boundary.

10.3 Step 3: Test structural feasibility and candidate inputs

Group finite scenarios by identical proposed inputs and compute the intersection of permissible-action sets. Empty intersections are impossibility certificates. For unique outputs, solve or bound the candidate-field separation program in Eq. (10). When the candidate set is large, report an exact integer-program optimum where feasible and otherwise a lower bound, an achieved upper bound, and the approximation method. If fields can be acquired adaptively, compute or bound Q_D^{wc} and Q_D^{avg} under an explicit query-cost and stopping model.

10.4 Step 4: Compute robust output structure

For hard tolerance D, construct the acceptable regions $S_a(D)$ and solve or bound Eq. (16). Use zero-error graph methods when the specification fits their side-information and unique-output assumptions; use a hypergraph or set-system representation when group compatibility is not pairwise.

10.5 Step 5: Estimate the sensor floor and average curve

Given $p(e, x)$ and $d(e, a)$, compute $\bar{d}(x, a)$ and

$$D_{\min}(X) = \sum_x p(x) \min_a \bar{d}(x, a).$$

If the target is below this floor, stop tuning the model. Add information, permit a query or escalation, weaken the target, or reduce the supported domain. For finite alphabets, trace $R_{\mathrm{aut}}(D)$ with Blahut–Arimoto. For large or continuous spaces, use justified abstractions, variational bounds, or task-specific estimators.

When $p(e, x)$ or the loss matrix is estimated from data, the audit quantities are themselves uncertain. Report bootstrap confidence bands, posterior intervals, or distributionally robust bounds for $D_{\min}$ and $R(D)$ rather than false-precision point estimates. Sampling uncertainty is separate from unmodeled-scenario uncertainty.

10.6 Step 6: Audit memory, tools, queries, and computation

Construct future-distinguishable history classes or an appropriate state-machine abstraction. List every required tool, permission, external effect, retry state, compensation, and information-acquisition action. Attach known time, space, query, latency, and cost obligations. The information certificate does not prove that the required mapping is computationally tractable.

10.7 Step 7: Generate witness tests

At minimum, generate:

- **separation tests** where a requirement-relevant input change must alter action;
- **invariance tests** where nuisance variation must not alter action;
- **missing-information tests** that require clarification, retrieval, abstention, or escalation;
- **output-capability tests** for every required action and tool effect;
- **trace tests** for ordering, retry, rollback, liveness, and exactly-once behavior; and
- **collision searches** for scenarios mapped to one effective state despite incompatible requirements.

10.8 Step 8: Compare reachable implementation behavior

Instrument the implementation where possible. Count only reachable states and actually available tools. Check refinement against the formalized specification, not merely code coverage. For black-box LLM systems, estimate behavioral distributions across repeated runs and environmental perturbations; a single successful trajectory does not establish that a stochastic policy preserves a boundary reliably.

Algorithm 1 Finite requisite complexity audit

Require: Finite scenarios $\mathcal{E}$; candidate fields $v_{1:m}$; permissible actions $C(e)$; optional fixed law $p(e, x)$ and loss $d(e, a)$; target D

Group scenarios by proposed observation $s(e)$
for all observation fibers F **do**
    if $\bigcap_{e \in F} C(e) = \varnothing$ **then**
        record F as an input-impossibility witness
    end if
end for
Solve or bound minimum candidate-field separation
Solve or bound adaptive worst-case and expected query cost
Solve or bound robust acceptable-action cover K_D^{wc}
if $p(e, x)$ and $d(e, a)$ are supplied **then**
    compute $\bar{d}(x, a)$ and $D_{\min}(X)$
    if $D < D_{\min}(X)$ **then**
        record probabilistic infeasibility at target D
    else
        trace $R_{\mathrm{aut}}(D)$ and uncertainty bands
    end if
end if
derive memory, query, output, tool, and computational obligations
generate separation, invariance, missing-information, and trace tests
compare reachable implementation abstraction with the certificate

11 Implications for AI Agents and LLM Workflows

For a tool-using language-model agent, the automation boundary extends beyond the context window. It includes retrieval queries, tool schemas, permissions, memory writes, summaries, clocks, environment observations, and the policy governing when more information may be acquired.

11.1 Prompts and PRDs

A long PRD can be structurally insufficient, while a compact typed schema can be sufficient. The relevant completeness criterion is not the proportion of a template filled out but the unresolved conditional entropy of consequential implementation choices. A requirement should state or make reliably inferable every distinction that changes acceptance, safety, or externally visible behavior. Examples, counterexamples, state tables, and acceptance tests function as error-correcting redundancy over a noisy human–agent communication channel.

11.2 Tool outputs and retrieval

Tool outputs should be designed around decision conflicts rather than descriptive abundance. Retrieval is conditional sensing: a retrieved item has value when it changes action consequences or lowers the sensor floor. Similarity-only retrieval metrics can miss this. The certificate asks whether retrieved evidence separates states that otherwise require incompatible actions.

11.3 Memory and summarization

Agent summaries can preserve narrative meaning while erasing exact commitments, dates, exceptions, provenance, or causal order. Future-distinguishable histories should not be merged. The appropriate lower bound depends on the distribution of future tasks and on whether raw history remains recoverable; decision-centric memory work directly addresses this rate–distortion problem [Zou et al., 2026, Colaco and Lahjouji, 2026].

11.4 Interactive information acquisition

An agent able to ask a question, call a tool, inspect a repository, or escalate changes the feasibility region. Clarification is an epistemic action with cost, delay, and expected reduction in future loss. A mature workflow should not force an irreversible world action when the current information state lies below the target's feasible region.

11.5 Stochastic policies

LLM behavior is stochastic, context-sensitive, and often nonstationary across model or prompt versions. Structural requirements remain applicable, but conformance becomes distributional. One should report conditional action probabilities, tail risks, repeatability, and confidence bounds, not merely whether one run found the expected answer. Hard safety invariants may require deterministic guards or formally verified tool wrappers outside the model policy.

12 Design Laws

The framework supports eight compact laws.

1. Task variety, not world variety. Required structure is set by distinctions that change action value, not by all measurable environmental variation.

2. Structural feasibility precedes model quality. If required behavior does not factor through the available inputs, downstream intelligence cannot make the task exact.

3. No inference after irreversible loss. Once a serial stage collapses a required distinction, later stages need another information source to recover it.

4. Tolerance is an interface parameter. Tighter error, safety, or service-level targets raise structural and information requirements.

5. More data is not necessarily more useful. A field earns its cost only through conditional decision value; irrelevant fields can fragment finite budgets.

6. Shared context is real capacity only when truly shared. Stable protocols, memory, and side information reduce incremental transfer only if both relevant stages possess and interpret them consistently.

7. Dynamic tasks generate fresh requirements. Feedback systems need causal information and memory at the rate the environment creates decision-relevant uncertainty.

8. Escalation can be optimal sensing. When current information cannot support the requested guarantee, the correct action may be to acquire information rather than guess.

13 Limitations and Open Problems

Several qualifications are essential.

First, the certificate is only as complete as the formalized requirement domain. It can prove that a modeled interface is impossible or that an implementation violates a modeled boundary; it cannot guarantee that stakeholders listed every important state, harm, or future continuation. Requirements elicitation remains a substantive empirical and normative task.

Second, functional structure is not identical to computational complexity. Adaptive query bounds also depend on the declared observation primitive, field costs, and whether derived or parallel queries are allowed. The task quotient, action cover, and memory classes lower-bound distinctions and state, not the time or space needed to compute the mapping. A task can have two outputs yet require difficult computation; another can have many output classes implemented by a direct lookup. Complexity-theoretic obligations must be attached separately.

Third, average mutual information can hide rare catastrophic cases. High-stakes systems may require zero-error, maximum-loss, tail-risk, or distributionally robust criteria. Conversely, a strict worst-case cover may be too conservative for benign approximate tasks. The loss and tolerance must match the actual governance problem.

Fourth, the operational meaning of mutual information depends on the coding model. The one-shot lower bound is general under the stated message assumptions, but exact rate achievability is asymptotic. Finite block length, latency, privacy, interpretability, restricted encoder classes, and no-shared-randomness constraints can raise practical requirements.

Fifth, $p(e, x)$ and conditional losses are frequently estimated. Finite samples, drift, strategic behavior, and dataset shift can move both $D_{\min}$ and $R(D)$. Confidence intervals do not address unobserved states; conservative uncertainty sets and ongoing drift audits may be necessary.

Sixth, exact candidate-field and acceptable-action optimization is computationally hard. Integer programming is practical only for bounded finite instances. Large open-world systems will require abstractions, decomposition, counterexample-guided refinement, and upper/lower bounds. Approximation error should be reported as part of the certificate.

Seventh, black-box LLM agents do not expose a stable finite state machine. Prompt context, model weights, sampling, tools, and provider updates jointly determine behavior. The certificate can define required observable distinctions and generate tests, but internal-state conformance may remain unidentifiable without instrumentation or architectural guards.

Eighth, actions can change what becomes observable. Active sensing, experimentation, dual control, and tool use require sequential formulations in which epistemic and world actions are optimized jointly. The one-shot theory is a foundation, not a complete treatment of interactive agency.

Finally, the choice of distortion and permissible action sets is normative. Information theory can reveal what follows from those choices; it cannot determine rights, consent, safety priorities, or acceptable tradeoffs.

14 Conclusion

Passing a sampled test suite does not by itself answer whether an automation has enough functional structure for its stated use case. That question can be made more precise.

For a finite one-shot specification, exact behavior is possible only when every observed input fiber retains a common permissible action. With a unique output, the required functionality must factor through the available inputs, and the task quotient is the coarsest exact representation. Candidate fields induce a minimum separation problem; permissible actions induce a robust cover; future traces induce memory classes. Under a fixed probability law and expected loss, the minimum statistical coupling is a task-action rate–distortion function obtained through the classical indirect-source reduction. These are distinct but compatible levels of one automation-boundary analysis.

The engineering consequence is a requirements-derived certificate rather than a demand for complexity for its own sake. Before coding, the certificate can expose impossible interfaces, missing actions, insufficient memory, absent tools, and unresolved computational obligations. After coding, it can generate separation, invariance, missing-information, and trace tests and support refinement against reachable behavior. A conflict witness proves that no larger model operating on the same collapsed input can solve the problem. A complete conforming implementation may nevertheless be very small.

The core design question is therefore:

> **Has the environment been encoded into the automation with every distinction, state, capability, and information path required for the promised behavior to be possible?**

That question is increasingly central for LLM workflows and agents. The proposed framework supplies a mathematical vocabulary and a practical audit path for asking it before a system's test suite creates false confidence.

A Blahut–Arimoto Form for the Task-Action Curve

For finite $\mathcal{X}$ and $\mathcal{A}$, fixed $p(x)$, and fixed surrogate distortion $\bar{d}(x, a)$, trace the constrained problem through

$$\mathcal{L}_\beta[p(a \mid x)] = \mathrm{I}(X; A) + \beta \mathbb{E}[\bar{d}(X, A)], \qquad \beta \geq 0. \tag{41}$$

The stationary channel has Gibbs form

$$p_\beta(a \mid x) = \frac{p_\beta(a) 2^{-\beta \bar{d}(x,a)}}{\sum_{a'} p_\beta(a') 2^{-\beta \bar{d}(x,a')}}, \tag{42}$$

with marginal

$$p_\beta(a) = \sum_x p(x) p_\beta(a \mid x). \tag{43}$$

Alternating Eqs. (42) and (43) yields points on the convex lower envelope. Large β prioritizes distortion; small β prioritizes rate. The exact zero-rate endpoint is obtained by the best constant action.

B Integer Programs for Finite Certificates

For unique required actions and candidate fields, Eq. (10) is a weighted hitting-set program. If y_j selects field j, every action-conflicting pair must be separated by at least one selected field.

For the robust action cover, let $w_a \in \{0, 1\}$ indicate selection of action a. The exact finite formulation is

$$\min \quad \sum_{a \in \mathcal{A}} w_a \tag{44}$$

$$\text{s.t.} \quad \sum_{a:\, e \in S_a(D)} w_a \geq 1, \qquad \forall e \in \mathcal{E}, \tag{45}$$

$$w_a \in \{0, 1\}. \tag{46}$$

The objective value is K_D^{wc}. Linear-program relaxations provide lower bounds; feasible integral solutions provide upper bounds.

C Reproducibility

The source bundle contains:

- `experiments/synthetic_verification.py`;
- `results/results.json`;
- `results/closed_form_verification.csv`;
- `results/appointment_rate_distortion.csv`;
- `results/appointment_scenarios.csv`;
- `results/appointment_requisite_complexity_certificate.json`;
- `results/appointment_generated_tests.json`; and

- the generated appointment-interface figure.

From the bundle root, run

```
python experiments/synthetic_verification.py
```

to regenerate the numerical results and figure. The script uses only finite enumeration, NumPy, and Matplotlib.

D Notation Summary

Table 5: Core notation.

Symbol	Meaning
E	latent environment state
X	observation available at the audited boundary
Z	transmitted or internal representation
A	selected action or output
$C(e)$	permissible actions in state e
$s(e)$	deterministic exposed input for a finite structural specification
$g(e)$	unique required action, when one exists
$d(e, a)$	operational distortion or regret
D	tolerated expected or pointwise distortion, as specified
$D_{\min}(X)$	irreducible sensor-limited expected distortion floor
D_0	best expected distortion without observation-dependent action
K_D^{wc}	minimum robust acceptable-action cover size
$R_{\mathrm{aut}}(D)$	minimum task-specific information coupling at expected distortion D
N_F	number of future-distinguishable history classes
$Q_D^{\mathrm{wc}}, Q_D^{\mathrm{avg}}$	minimum worst-case and expected adaptive query cost under the declared primitive
RCC_D	structured requisite complexity certificate

References

Martín Abadi and Leslie Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991. doi: 10.1016/0304-3975(91)90224-P.

Dilip Arumugam and Benjamin Van Roy. Deciding what to learn: A rate-distortion approach. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 373–382, 2021.

W. Ross Ashby. *An Introduction to Cybernetics*. Chapman and Hall, London, 1956.

Toby Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Prentice-Hall, Englewood Cliffs, NJ, 1971. ISBN 0137531036.

Tsun S. Chow. Testing software design modeled by finite-state machines. *IEEE Transactions on Software Engineering*, SE-4(3):178–187, 1978. doi: 10.1109/TSE.1978.231496.

Vašek Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979. doi: 10.1287/moor.4.3.233.

Ashwin Gerard Colaco and Nada Lahjouji. What to keep, what to forget: A rate–distortion view of memory compaction in LLMs and agents. *arXiv preprint arXiv:2607.08032*, 2026.

Roger C. Conant and W. Ross Ashby. Every good regulator of a system must be a model of that system. *International Journal of Systems Science*, 1(2):89–97, 1970. doi: 10.1080/00207727008920220.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, Hoboken, NJ, 2 edition, 2006. doi: 10.1002/047174882X.

R. L. Dobrushin and B. S. Tsybakov. Information transmission with additional noise. *IEEE Transactions on Information Theory*, 8(5):293–304, 1962. doi: 10.1109/TIT.1962.1057738.

Bernd Finkbeiner and Sven Schewe. Bounded synthesis. *International Journal on Software Tools for Technology Transfer*, 15(5–6):519–539, 2013. doi: 10.1007/s10009-012-0228-z.

Tim Genewein, Felix Leibfried, Jordi Grau-Moya, and Daniel A. Braun. Bounded rationality, abstraction, and hierarchical decision-making: An information-theoretic optimality principle. *Frontiers in Robotics and AI*, 2:27, 2015. doi: 10.3389/frobt.2015.00027.

C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12 (10):576–580, 583, 1969. doi: 10.1145/363235.363259.

Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976. doi: 10.1016/0020-0190(76)90095-8.

Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972. doi: 10.1007/978-1-4684-2001-2_9.

János Körner. Coding of an information source having ambiguous alphabet and the entropy of graphs. In *Transactions of the Sixth Prague Conference on Information Theory, Statistical Decision Functions, Random Processes*, pages 411–425, Prague, 1973. Academia.

János Körner and Alon Orlitsky. Zero-error information theory. *IEEE Transactions on Information Theory*, 44(6):2207–2229, 1998. doi: 10.1109/18.720537.

Cheuk Ting Li and Abbas El Gamal. Strong functional representation lemma and applications to coding theorems. *IEEE Transactions on Information Theory*, 64(11):6967–6978, 2018. doi: 10.1109/TIT.2018.2865570.

Girish N. Nair, Robin J. Evans, Iven M. Y. Mareels, and William Moran. Topological feedback entropy and nonlinear stabilization. *IEEE Transactions on Automatic Control*, 49(9):1585–1597, 2004. doi: 10.1109/TAC.2004.834105.

Anil Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958. doi: 10.1090/S0002-9939-1958-0135681-9.

Alon Orlitsky and James R. Roche. Coding for computing. *IEEE Transactions on Information Theory*, 47(3):903–917, 2001. doi: 10.1109/18.915643.

Pedro A. Ortega and Daniel A. Braun. Thermodynamics as a theory of decision-making with information-processing costs. *Proceedings of the Royal Society A*, 469(2153):20120683, 2013. doi: 10.1098/rspa.2012.0683.

Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 179–190. Association for Computing Machinery, 1989. doi: 10.1145/75277.75293.

Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27 (3):379–423, 1948a. doi: 10.1002/j.1538-7305.1948.tb01338.x.

Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27 (4):623–656, 1948b. doi: 10.1002/j.1538-7305.1948.tb00917.x.

Claude E. Shannon. The zero error capacity of a noisy channel. *IRE Transactions on Information Theory*, 2(3):8–19, 1956. doi: 10.1109/TIT.1956.1056798.

Claude E. Shannon. Coding theorems for a discrete source with a fidelity criterion. In *IRE National Convention Record*, volume 7, pages 142–163, 1959.

Christopher A. Sims. Implications of rational inattention. *Journal of Monetary Economics*, 50(3): 665–690, 2003. doi: 10.1016/S0304-3932(03)00029-1.

Photios A. Stavrou and Marios Kountouris. The role of fidelity in goal-oriented semantic communication: A rate distortion approach. *IEEE Transactions on Communications*, 71(7):3918–3931, 2023. doi: 10.1109/TCOMM.2023.3274122.

D. J. Strouse and David J. Schwab. The deterministic information bottleneck. *Neural Computation*, 29(6):1611–1630, 2017. doi: 10.1162/NECO_a_00961.

Sekhar Tatikonda and Sanjoy K. Mitter. Control under communication constraints. *IEEE Transactions on Automatic Control*, 49(7):1056–1068, 2004. doi: 10.1109/TAC.2004.831187.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Hans S. Witsenhausen. The zero-error side information problem and chromatic numbers. *IEEE Transactions on Information Theory*, 22(5):592–593, 1976. doi: 10.1109/TIT.1976.1055607.

Hans S. Witsenhausen. Indirect rate distortion problems. *IEEE Transactions on Information Theory*, 26(5):518–521, 1980. doi: 10.1109/TIT.1980.1056251.

Jack K. Wolf and Jacob Ziv. Transmission of noisy information to a noisy receiver with minimum distortion. *IEEE Transactions on Information Theory*, 16(4):406–411, 1970. doi: 10.1109/TIT.1970.1054469.

Mingxi Zou, Zhihan Guo, Langzhang Liang, Zhuo Wang, Qifan Wang, Qingsong Wen, Irwin King, Lizhen Qu, and Zenglin Xu. Remember the decision, not the description: A rate-distortion framework for agent memory. *arXiv preprint arXiv:2605.10870*, 2026.