SCIDIALECT: SYMBOLIC COMPRESSION AS AN INTRINSIC REWARD FOR SCIENTIFIC DISCOVERY AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

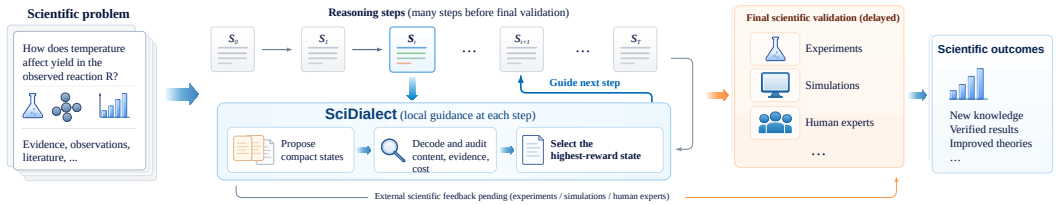
Autonomous scientific discovery often depends on expensive or delayed external validation, leaving intermediate reasoning decisions without a task reward. We ask whether abstraction itself can provide useful intrinsic guidance. Our principle, *grounded symbolic compression*, rewards shorter representations of task-visible scientific content while checking that their claims, conditions, and evidence remain recoverable through a designed decoder. We instantiate it in SCIDIALECT, a black-box controller that calibrates reusable scientific primitives on development tasks, then uses independently validated, query-specific instantiations to condition the next solver step. Its online representation reward requires no external correctness score. Our analysis establishes when verified predictive compression improves expected log score after representation cost, and when the scientific-state representation preserves downstream decision information, with a qualified minimality result. Across PreScience, DiscoveryBench, and SciCode, our proposed SCIDIALECT achieves the highest mean contribution token F1 on three model families, improves adapted discovery score over prompt compression, and improves mean observable code execution. Four equal-budget scientific optimization tasks, *i.e.*, MBO, NHO, SPO, and TMC, provide complementary evidence of improved trajectory quality when the intrinsic controller augments an existing objective-based search. Together, these results support grounded abstraction as an intrinsic signal for scientific reasoning, without treating it as a substitute for scientific validation.

1 INTRODUCTION

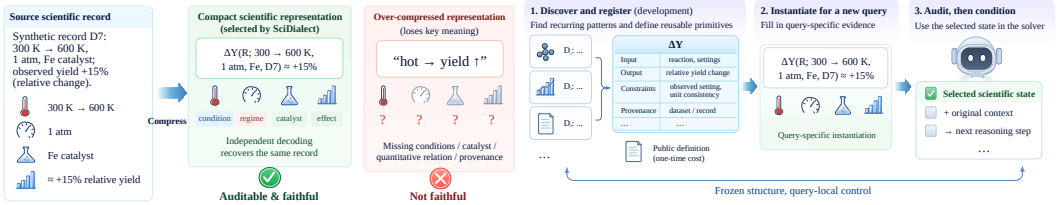
Autonomous scientific discovery (ASD) increasingly uses language-model agents to propose hypotheses, write scientific code, and choose experiments. Prominent systems obtain decisive evidence of progress from explicit external evaluation: FunSearch and AlphaEvolve score executable candidates, while AI Scientist workflows execute experiments and evaluate their outcomes (Romera-Paredes et al., 2024; Novikov et al., 2025; Lu et al., 2024; Yamada et al., 2025). Such feedback can be expensive, delayed, or unavailable during the reasoning steps that precede a complete candidate. A multi-agent system (MAS) must nevertheless decide which observations, assumptions, and relations should guide its next computation. This raises our scientific question: *can an intrinsic reward guide scientific reasoning without an external correctness signal?*

Intrinsic motivation offers several starting points, but their signals address different questions. Prediction-error and information-gain rewards encourage exploration, including settings without extrinsic rewards (Pathak et al., 2017; Houthooft et al., 2016); self-feedback can refine an answer without a separate evaluator (Madaan et al., 2023). Compression progress measures improved prediction, and recent sealed-audit analysis clarifies when its accumulated reward resists exploitation (Schmidhuber, 2008; Gupta, 2026). These approaches need not depend on external correctness feedback, but they do not specify which scientific commitments a shorter intermediate state must retain. Novelty, agreement, or brevity can coexist with a missing unit, a reversed relation, or an unsupported assumption. Therefore, the unresolved question is, *what intrinsic criterion can efficiently reward useful abstraction rather than information loss?*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Guiding reasoning before final validation.



(b) Checking what compression preserves..

(c) Reusing symbolic templates for new queries.

Figure 1: **SciDialect guides scientific reasoning with symbolic compression.** **(a)** Before final validation, the controller proposes compact representations, decodes and checks them, and selects the highest-reward candidate that passes the checks. **(b)** Compression is rewarded only after passing checks for recoverable claims, conditions, and evidence. **(c)** Reusable symbolic templates are built offline, frozen, and filled with information from each new query. The selected representation is added to the solver's original context to guide its next reasoning step.

Human reasoning suggests a motivation. The working memory of the human brain is limited; thus, we use familiar chunks to handle structured information as units (Cowan, 2001). Scientific notation similarly makes explicit structure easier to manipulate, *e.g.*, $\sum_{i=1}^{n} x_i$ replaces an expanded sum without changing its terms when the indexing convention is known. This motivates *lossless abstraction of declared scientific content*: an agent should be rewarded for expressing the same task-specific facts with fewer tokens, provided that a decoder can recover them using shared definitions. CLSR shows that LLM agents can autonomously develop reusable symbolic communication protocols (Pei et al., 2026); thus, the question here is whether validated abstraction can itself provide guidance without online correctness rewards.

We formulate this idea as *grounded symbolic compression*. A scientific state records the claims, assumptions, and evidence currently relevant to the next reasoning step. Its symbolic abstraction receives compression credit only after reconstruction, and grounding checks assess whether the required content survives. Public definitions are charged rather than hidden outside the expression, following the minimum-description-length accounting (Grünwald, 2007). The reward thus concerns a verifiable relation between an available source state and its abstraction; it does not require the agent to know the final answer.

SciDialect implements the principle as a black-box test-time controller (Figure 1). Offline, development tasks supply recurring scientific structures from which a reusable dictionary is calibrated and frozen. For a new query, the controller identifies the source claims and their evidence anchors, then proposes compact instantiations of that dictionary. Blinded reconstruction channels recover the proposed meanings, which are checked against the source before selection. The highest-reward admissible state augments the next solver call; when no candidate passes, the solver receives its original context alone. The reward therefore changes what the agent reasons from, without updating its parameters or consulting a hidden task score.

Experiments validate this mechanism across three LLM-facing benchmarks and four controlled scientific optimization tasks. On PreScience (Ajith et al., 2026), SciDialect improves token F1 by 2.3–3.9 percentage points over paired backbone-specific comparators while preserving substantially more task-visible grounding. On DiscoveryBench (Majumder et al., 2025), a compact one-call adapter improves the adapted official-code score by 4.1 points over prompt compression. On SciCode (Tian et al., 2024), a contract-aware adapter improves the mean observable execution on both evaluated backbones. We also show that SciDialect can improve the cases with typical external feedback. Under four equal-budget scientific optimization tasks, *i.e.*, molecular bioactivity

(MBO), nanomaterial optical-property (NHO), superconductor critical-temperature (SPO) (Pu et al., 2025), and transition-metal-complex (TMC) optimization (Song et al., 2025), SCIDIALECT improves endpoint and cumulative-best trajectory quality over PIEVO (Pu et al., 2026) while eliminating invalid proposals.

Our theory addresses two questions (Section 4). First, under proper coding and independent verification, net compression progress equals predictive log-score improvement after representation cost and controls a bound on scientific decision regret. Second, public primitives, together with evidence-linked arguments and residual text, can represent the declared scientific state without losing downstream decision information. The resulting semantic state is minimal in the distinctions it must preserve under exact recovery; its particular symbolic spelling is not claimed to be globally shortest. These results validate the intrinsic principle from the practical audit scores used to approximate it.

Our contributions are:

- An intrinsic objective that rewards grounded abstraction of scientific content without an online external correctness score.
- A black-box controller that turns independently validated scientific symbolic states into solver-visible guidance.
- A two-part theoretical analysis of predictive utility and representation sufficiency, including explicit conditions for semantic minimality.
- Evaluation across three scientific benchmarks, three LLM families, and four equal-budget scientific optimization tasks.

2 RELATED WORK

Scientific discovery agents. Evaluator-guided search systems such as FunSearch, AlphaEvolve, and PiEvo use executable or scientific objectives to compare candidates (Romera-Paredes et al., 2024; Novikov et al., 2025; Pu et al., 2026). AI co-scientist and AI Scientist systems also organize generation, critique, experimentation, and review (Gottweis et al., 2025; Lu et al., 2024; Yamada et al., 2025). SCIDIALECT addresses a complementary decision: which evidence-linked intermediate state should guide the next computation when an external correctness score is unavailable.

Intrinsic motivation and symbolic abstraction. Curiosity, information gain, and compression progress provide internal objectives for exploration and learning (Oudeyer & Kaplan, 2007; Pathak et al., 2017; Houthooft et al., 2016; Schmidhuber, 2008). Sealed-audit compression analysis establishes an endpoint accounting guarantee under a fixed verification protocol (Gupta, 2026). DreamCoder, LILO, and PySR learn reusable programs or symbolic expressions (Ellis et al., 2020; Grand et al., 2024; Cranmer, 2023); CLSR evolves symbolic communication using correctness and token-cost feedback (Pei et al., 2026). Our contribution is the scientific-state objective and its control role: public abstraction is ranked only after checking recoverable commitments and accounting for definitions, without an online correctness reward.

Reasoning and context compression. Deliberation, retrieval, debate, communication, and self-repair improve how agents construct and revise answers (Wei et al., 2022; Wang et al., 2023; Yao et al., 2023b; Shinn et al., 2023; Madaan et al., 2023; Chen et al., 2024; Du et al., 2024; Lewis et al., 2020; Yao et al., 2023a; Pei et al., 2024; 2025). Prompt compression and concise reasoning reduce context or generation length (Jiang et al., 2023; Cheng & Van Durme, 2024; Aytes et al., 2025; Nayab et al., 2024). SCIDIALECT instead treats brevity as a conditional reward: the compact scientific state must remain reconstructable and grounded before it can guide the solver.

3 METHODOLOGY: GROUNDED COMPRESSION FOR TEST-TIME CONTROL

3.1 PROBLEM FORMULATION AND METHOD OVERVIEW

Problem and interface. A task τ provides an instruction and ordered context $\mathcal{K}_\tau$, such as prior papers, dataset metadata, or a programming specification. The required output is $y_\tau \in \mathcal{Y}_\tau$, where $\mathcal{Y}_\tau$ is the task's output space, *e.g.*, a contribution forecast, a data-supported hypothesis, or an executable program. An external evaluator $R_{\mathrm{ext},\tau} : \mathcal{Y}_\tau \to \mathbb{R}$ may eventually assess the output, but its score is not an input to the representation reward. The goal is to improve scientific reasoning within a fixed

budget using permitted evidence and intrinsic representation checks. The controller returns y_τ and a structured audit log $\mathcal{L}_\tau$.

Methodological instantiation. SCIDIALECT is a controller rather than a trained network. Its solver is an agent/MAS workflow with a callable interface and fixed model weights. A task *adapter* v is a fixed bundle of programs, prompts, and settings that connect this interface to the evidence and output contract of the task. The controller also receives a frozen dictionary $\mathcal{D}^\star$ of public primitive definitions and positive-integer budgets T, K, J for rounds, candidates per round, and reconstruction channels. The proposer and decoder roles can use separate calls to the same pretrained model; parsing, scoring, selection, and memory assembly are programmatic (Appendix A.1).

Overview. Offline calibration constructs reusable primitives from development tasks. Online, the controller first identifies the scientific content that must survive abstraction. It proposes compact expressions and asks blinded decoders to recover their meanings from public definitions. Only candidates that pass the configured validations are eligible for intrinsic-reward selection. The selected representation augments the solver's original context; the solver then returns an answer or continues reasoning, generating the next round's visible state. Algorithm 1 gives the complete loop, and Appendix B presents four scientific discovery samples using SCIDIALECT.

3.2 SCIENTIFIC STATES AND PUBLIC PRIMITIVES

At round t, $X_t = \mathcal{K}_\tau \oplus H_t$ concatenates the original context with the finite history H_t of non-final solver responses. The extractor Prop_v uses task-specific parsing or a fixed LLM prompt with schema checks to return $P_t = (p_{t,1}, \ldots, p_{t,n_t})$, an ordered list of relevant claims, assumptions, relations, and requirements. This *source scientific state* is a structured record rather than a neural vector. An *evidence anchor* is a typed record $(\text{type}, \text{value}, \text{source}, \text{role})$ that identifies a detail that must remain recoverable, such as a value with its unit or a function with its signature. The abstraction target is this declared state and its anchors; hypothetical claims retain their hypothetical status.

A *scientific primitive* is a reusable symbolic template with public meaning, typed arguments, evidence slots, validity conditions, and a decoding rule. The primitive is stored as structured data. A *dialect expression* binds a primitive's arguments to query-specific content and retains residual natural language for details that cannot be safely abbreviated (Appendix A.3). Offline, a prompted primitive inducer proposes reusable templates; programmatic registration and reconstruction checks retain them only when verified reuse repays their configured cost. The dictionary is frozen before held-out inference (Appendix A.6). Any online definitions $\Delta\mathcal{D}^{\mathrm{loc}}$ are public, charged to the candidate, and discarded after the query rather than promoted into $\mathcal{D}^\star$.

3.3 FROM A SOURCE STATE TO A VALIDATED ABSTRACTION

The proposer is an LLM-based agent invoked with a fixed encoding prompt or a deterministic template binder in a compact adapter. Given P_t, the visible context and $\mathcal{D}^\star$, it returns at most K candidate expression sequences Z with any local definitions. After reconstruction, a candidate is the finite record $c = (Z, \widehat{P}, \Delta\mathcal{D}^{\mathrm{loc}})$, where $\widehat{P}$ contains the decoded propositions. Each source proposition has a nonempty required anchor set; when typed extraction fails, a verbatim-content anchor requires retaining that proposition in residual text. Appendix A.4 specifies this implementation details.

A reconstruction channel is a separate agent call or a deterministic public-template decoder. It receives the expressions, public definitions, and allowed evidence, but not the explicit source-state list or the proposer rationale. The *auditor* then compares its reconstruction with the source for semantic fidelity and anchor recovery. For example, "hot $\rightarrow$ yield $\uparrow$" loses the conditions and provenance in Figure 1b. The audit checks record preservation, not chemical truth. For $J \geq 2$, channels do not share reconstructions; operational separation does not imply statistically independent errors. A one-channel adapter omits agreement scoring. The exact inputs and score definitions are presented in the Appendices A.5 and A.7.

3.4 INTRINSIC RANKING AND SOLVER CONDITIONING

The programmatic scoring assigns compression gain $G_{\mathrm{comp}}(c) \in \mathbb{R}$; reconstruction, grounding and agreement scores $S_{\mathrm{rec}}(c), S_{\mathrm{ground}}(c), S_{\mathrm{agree}}(c) \in [0, 1]$; and dictionary penalty $B_{\mathrm{dict}}(c) \in [0, \infty)$.

Algorithm 1 SCIDIALECT: intrinsic control for one held-out task

Require: Task τ, visible context $\mathcal{K}_\tau$, frozen adapter v, dictionary $\mathcal{D}^\star$, and budgets T, K, J.
Ensure: Final output $y_\tau \in \mathcal{Y}_\tau$ and audit log $\mathcal{L}_\tau$.
Initialize history $H_0 \leftarrow \varnothing$ and log $\mathcal{L}_\tau \leftarrow \varnothing$.
for $t = 0, \ldots, T-1$ **do**
  Form $X_t \leftarrow \mathcal{K}_\tau \oplus H_t$; extract $P_t \leftarrow \mathrm{Prop}_v(X_t)$; set $M_t \leftarrow \varnothing$.
  if $P_t \neq \varnothing$ **then**
    Extract required anchors and propose at most K public symbolic candidates.
    Reconstruct through J blinded channels and audit, obtaining $\mathcal{C}_t$.
    Apply the frozen gates to obtain $\mathcal{F}_t$ (Eq. 13).
    if $\mathcal{F}_t \neq \varnothing$ **then**
      Select $c_t^\star \leftarrow \arg\max_{c \in \mathcal{F}_t} R_{\mathrm{SD}}(c)$ and render $M_t \leftarrow \mathrm{Mem}_v(c_t^\star)$.
    end if
  end if
  Call the solver with $X_t \oplus M_t$ to obtain o_{t+1}; log the audit, memory, and response.
  if $\mathrm{Final}_v(o_{t+1}) = 1$ **then**
    Parse y_τ; save the log, discard query-local state, and **return** $(y_\tau, \mathcal{L}_\tau)$.
  end if
  Update $H_{t+1} \leftarrow H_t \oplus o_{t+1}$.
end for
Obtain y_τ using the predeclared budget-exhaustion rule; save the log and discard query-local state.
return $(y_\tau, \mathcal{L}_\tau)$, leaving $\mathcal{D}^\star$ unchanged.

Compression compares the source and expression lengths under fixed coding rules. The grounding measures recovered anchors, whereas reconstruction compares their scientific meaning and relations. The intrinsic score uses the predefined formula

$$R_{\mathrm{SD}}(c) = w_c G_{\mathrm{comp}}(c) + w_r S_{\mathrm{rec}}(c) + w_g S_{\mathrm{ground}}(c) + w_a S_{\mathrm{agree}}(c) - w_b B_{\mathrm{dict}}(c), \tag{1}$$

with nonnegative, frozen weights and no external correctness term. The Appendices A.1 and A.7 specify the comparison routines and cost terms; no separately trained reward model is used.

The Ranking follows the admissibility check. Let $\mathcal{C}_t$ be the finite candidate set and $\mathcal{F}_t \subseteq \mathcal{C}_t$ the subset that satisfies the reconstruction, grounding, agreement, dictionary-cost, leakage and contract gates in Eq. 13. Brevity cannot compensate for a failed gate. The expected lossless condition preserves all declared content; practical numerical audits approximate it, and grounding thresholds below one permit some anchor loss. A passed audit is therefore not a proof of exact recovery.

If the candidate subset $\mathcal{F}_t$ is not empty, the controller selects its highest-reward member $c_t^\star$ and serializes its expressions, recovered meanings, definitions, and provenance as memory M_t, with deterministic linking. Otherwise, M_t is empty. The solver produces

$$o_{t+1} = \mathrm{Solve}_v(\tau, X_t \oplus M_t) \in \mathcal{O}_v, \tag{2}$$

where $\mathcal{O}_v$ is the adapter's space of solver responses. The original context remains available, so representation compression does not need to reduce the total inference cost. A programmatic final-response check tests the output contract, not the scientific correctness. A non-final response enters the history for the next round; a fixed finalization rule handles budget exhaustion (Appendix A.8).

3.5 TASK ADAPTERS AND THE FEEDBACK BOUNDARY

Adapters implement task-specific interfaces to the same representation-control principle. Data discovery uses dataset and metadata readers; forecasting binds the scholarly context; coding uses a contract parser and a visible execution harness. A compact realization can use $T = K = J = 1$ with deterministic reconstruction; a repair adapter can alternate code generation and permitted feedback within a fixed budget (Appendix A.9). The *representation reward* needs no external correctness score, but a resulting workflow is allowed to use the provided feedback. SciCode can provide visible verifier responses as evidence. Compatible-core optimization interfaces with a numerical optimizer, retaining its ordinary objective-based acquisition and surrogate-model updates while dialect checks constrain principle states (Appendix A.10). These permissions neither expose hidden held-out targets nor allow online changes to the persistent primitive dictionary.

4 THEORY: WHY ABSTRACTION CAN PROVIDE INTRINSIC GUIDANCE

The analysis answers two questions: when compression is informative about scientific prediction; and when the representation retains the information needed to use that signal. The results concern a precisely specified coding objective and recovery condition; they do not assume that arbitrary token savings or judge scores satisfy them. The proofs and formal statements are given in Appendix D.

4.1 FROM VERIFIED COMPRESSION TO PREDICTIVE UTILITY

Let $S \in \mathcal{S}$ be a canonical scientific record, where $\mathcal{S}$ is a finite or countable set of proposition-and-anchor records with equivalent surface wordings identified. Let $\mathcal{K}$ denote fixed side information supplied to both codes, not an external correctness score. A public representation scheme d induces a normalized distribution $q_d(S \mid \mathcal{K}) > 0$ on the relevant support and the ideal codelength $\ell_d(S \mid \mathcal{K}) = -\log_2 q_d(S \mid \mathcal{K})$ in bits (Rissanen, 1978; Delétang et al., 2024). Its interpretation cost $C(d) \geq 0$ is charged for all required definitions. For two schemes d, d', define $\Delta C = C(d') - C(d)$ and the intrinsic net reward

$$r_{\text{dial}} = \ell_d(S \mid \mathcal{K}) - \ell_{d'}(S \mid \mathcal{K}) - \Delta C. \tag{3}$$

This reward is computable from the observed record and public codes without a success label.

Theorem 1 (Verified abstraction and predictive improvement)**.** *Fix d, d' before drawing a verification record $S \sim p(\cdot \mid \mathcal{K})$, where p is the distribution of unknown records. Assume both code distributions are positive in its support and the following expectations are finite. Then*

$$\mathbb{E}[r_{\text{dial}}] = D_{\text{KL},2}(p \| q_d) - D_{\text{KL},2}(p \| q_{d'}) - \Delta C, \tag{4}$$

where $D_{\text{KL},2}$ is the Kullback–Leibler divergence in bits, with the conditioning on $\mathcal{K}$ suppressed. Equivalently, the expected intrinsic reward is the expected log-score improvement minus the representation cost.

Thus, maximizing expected reward minimizes predictive mismatch plus representation cost. If the added cost is nonnegative, the expected positive reward implies an improved prediction of independently verified scientific records. For a decision rule that maximizes predicted bounded utility, the lower mismatch also tightens a regret bound; the decision can be a hypothesis choice, an experiment, or a joint MAS action (Corollary 4). This establishes a conditional route from intrinsic abstraction reward to scientific decision quality, rather than a universal guarantee that every shorter prompt improves a fixed solver.

The distinction from the implementation matters. Separate decoders check the reconstruction of a construction record; they do not turn that record into an independent prediction sample. Moreover, tokenizer lengths need not be proper predictive codelengths, and the fidelity terms in Eq. 1 can alter candidate rankings. Appendix D.1 states the correspondence conditions; Proposition 3 gives a same-record overfitting counterexample.

4.2 SUFFICIENT SCIENTIFIC STATES AND QUALIFIED MINIMALITY

For a current canonical record S, write $R = (Z, \Delta\mathcal{D}^{\text{loc}})$ for its symbolic expressions and local definitions; the frozen dictionary is included in the shared side information $\mathcal{K}$. A decoder g maps $(R, \mathcal{K})$ back to a canonical record. Residual text makes the language expressive even when no useful primitive applies.

Theorem 2 (Sufficiency of the scientific-state representation)**.** *Assume every field and relation in the finite source record has a public literal serialization, and the expression language permits residual text. Then an encoder E and decoder g exist such that $g(E(S, \mathcal{K}), \mathcal{K}) = S$. For any such exact encoding $R = E(S, \mathcal{K})$ and any downstream variable Y with a finite or countable outcome space,*

$$p(Y \mid R, \mathcal{K}) = p(Y \mid S, \mathcal{K}). \tag{5}$$

Every decision rule based on $(S, \mathcal{K})$ can therefore be implemented from $(R, \mathcal{K})$ with the same expected utility. If reconstruction instead fails with probability at most $\varepsilon \in [0, 1]$, emulating a source-state decision rule loses at most ε expected utility for utilities in $[0, 1]$.

This is sufficiency relative to the declared source record. It extends to the full task only if that record contains all task-relevant information; the anchor extractor is not assumed to know every scientifically relevant fact. The construction also does not guarantee positive compression or that the implemented proposer finds the best encoding. Under a fixed representation budget, exact shortening can make a previously infeasible source-state policy feasible without removing information (Corollary 5).

Proposition 1 (What can be minimal)**.** *Fix side information $\mathcal{K} = k$ and require exact recovery of every declared scientific commitment. Any admissible representation must distinguish source states with different commitments. The canonical semantic record is therefore minimal in these distinctions, up to relabeling. For a random record S with finite entropy, every conditional prefix code satisfies $\mathbb{E}[L(R) \mid k] \geq H_2(S \mid k)$, where H_2 denotes entropy in bits.*

This does not establish that SCIDIALECT's dictionary or token string is globally shortest. Nor does it equate recoverability with scientific truth: a faithfully preserved hypothesis still requires external validation. Appendix D covers approximate recovery and definition amortization; Appendix C gives a separate evidence-acquisition extension.

5 EXPERIMENTAL PROTOCOL

Comparisons share the benchmark, split, evaluator, metric definitions, and backbone where applicable. LLM budgets are reported per configuration; compatible-core optimization fixes objective-evaluation and proposal budgets.

PreScience (Ajith et al., 2026). We evaluate contribution generation on 850 held-out papers for each of Qwen3.5-35B-A3B (Team, 2026), DeepSeek-V3 (Liu et al., 2024), and GLM-4-32B-0414 (Glm et al., 2024). Token F1 measures target contribution-content recovery; grounding measures preservation of task-visible prior-work and contribution anchors.

DiscoveryBench (Majumder et al., 2025). Adapter selection uses 48 development tasks; frozen evaluation uses 191 disjoint tasks, three generation seeds, and 5,157 evaluator rows using Qwen3.5-35B-A3B during generation and evaluation. The primary measure is the adapted official-code final score, with recall-context coverage as an evidence-recovery diagnostic.

SciCode (Tian et al., 2024). We evaluate 50 held-out scientific-programming problems for Qwen3.5-35B-A3B and GLM-4-32B-0414; observable execution records whether a program runs in the provided harness, and parse rate records syntactic validity. Because a recent domain-expert audit reports specification and test defects in the original benchmark, we interpret these numbers as behavior under the released evaluation contract rather than as an absolute estimate of scientific-coding competence (Hu et al., 2026).

Scientific optimization. We study molecular bioactivity (MBO), nanomaterial optical-property (NHO), superconductor critical-temperature (SPO) (Pu et al., 2025), and transition-metal-complex (TMC) optimization (Song et al., 2025) through a backbone-free compatible-core implementation of the PIEVO interface (Pu et al., 2026). Every method receives the same deterministic surrogate, 24 objective evaluations, eight proposals per selection step, and 20 paired seeds. Solution quality (SQ) is final best normalized objective, and area under the cumulative-best curve (AUOC) measures trajectory quality.

5.1 COMPARATORS AND STATISTICAL PROTOCOL

LLM-facing comparators cover direct generation, chain-of-thought, self-consistency, prompt compression, retrieval augmentation, multi-agent forecasting, candidate reranking, iterative critique, and repair (Wei et al., 2022; Wang et al., 2023; Jiang et al., 2023; Lewis et al., 2020; Du et al., 2024; Zheng et al., 2023; Madaan et al., 2023; Shinn et al., 2023); SciCode additionally includes ReAct, Self-Debug, Code Rerank, and visible-test repair (Yao et al., 2023b; Chen et al., 2024; 2023). The optimization study compares random search, a genetic algorithm, differential evolution, a vanilla proposer–critic MAS policy, and PIEVO (Bergstra & Bengio, 2012; Holland, 1992; Storn & Price, 1997; Pu et al., 2026). Appendix F specifies all roles and configurations. Held-out means use task-level bootstrap 95% confidence intervals (CIs); paired comparisons use shared tasks or seeds, paired bootstrap intervals, and two-sided sign tests.

Method	Qwen3.5-35B-A3B			DeepSeek-V3			GLM-4-32B-0414		
	F1	Grounding	Tokens	F1	Grounding	Tokens	F1	Grounding	Tokens
Direct	16.7	29.2	862	15.2	30.9	817	14.6	29.8	821
CoT	17.2	24.9	867	15.1	36.3	830	14.4	35.9	834
Self-consistency	16.6	22.9	1183	15.8	19.3	938	15.2	24.4	987
Prompt compression	16.8	27.6	911	14.8	**37.1**	872	13.5	**41.9**	875
Multi-agent forecasting	17.0	24.0	1180	15.7	19.8	929	14.4	21.4	974
Retrieval only	16.1	24.4	1170	16.0	22.2	958	15.0	25.8	977
Reviewer rerank	16.1	22.8	1159	15.8	18.9	922	14.5	22.5	975
Self-refine	16.2	23.6	2364	13.9	18.1	1870	14.0	22.0	1940
Reflexion	16.1	23.3	2358	13.4	17.0	1862	14.0	20.9	1909
Frontier direct	16.3	22.9	1152	15.6	19.8	923	14.7	23.5	959
SCIDIALECT	**19.6**	**39.8**	909	**19.8**	**37.1**	854	**19.6**	38.5	865

Table 1: Held-out PreScience contribution-generation results on 850 tasks per backbone. F1 is token F1, Grounding is task-visible anchor coverage, and Tokens is the per-generation token count. Cache-free cost is reported in Appendix H.

Method	Final score [95% CI]	Recall	Tokens	Calls
Direct	13.9 [10.7, 17.2]	**37.5**	2232	1
Multi-agent-3x	12.9 [9.5, 16.5]	33.7	5782	3
Self-consistency-3x	12.9 [9.8, 16.2]	35.1	5526	3
ConceptBank	12.8 [9.8, 16.0]	36.0	1742	1
CoT	12.5 [9.5, 15.6]	37.0	1739	1
Prompt compression	11.7 [8.9, 14.8]	33.9	1783	1
SCIDIALECT	**15.8 [12.1, 19.9]**	36.3	1976	1

Table 2: DiscoveryBench adapted official-code results on 191 frozen held-out tasks and three generation seeds. The generator is Qwen3.6-35B-A3B and the evaluator is fixed across methods. Tokens and calls describe generation only; complete cache-free cost is reported in Appendix G.

6 RESULTS

6.1 PRESCIENCE: CROSS-BACKBONE CONTRIBUTION FORECASTING

Table 1 reports the complete held-out comparison. SCIDIALECT has the highest token F1 on all three backbones and preserves strong task-visible grounding across all three model families. Cache-free cost and the matched paired-comparator analysis appear in Appendix H. Against the paired Qwen CoT, DeepSeek Retrieval Only, and four-call GLM reviewer-rerank comparators, SCIDIALECT improves token F1 by 2.3, 3.8, and 3.9 percentage points, with paired 95% CIs of $[2.0, 2.7]$, $[3.4, 4.1]$, and $[3.5, 4.2]$. Grounding increases by 14.9, 14.9, and 16.8 percentage points; the corresponding cache-free cost analysis is reported in Appendix H.

6.2 DISCOVERYBENCH: FROZEN HELD-OUT QUALITY

Table 2 reports the frozen held-out comparison. The compact one-call dialect has the highest adapted official-code mean. It exceeds Prompt Compression by 4.1 percentage points and is 1.9 points above Direct. The comparison with Prompt Compression isolates the role of public semantics: both methods shorten the working context, but only SCIDIALECT requires independent reconstruction and anchor recovery before compression receives credit. The selected renderer therefore emphasizes the smallest evaluator-relevant state, *e.g.*, proposition, evidence, and relation, while Appendix G records the frozen development protocol used to select it.

6.3 SCICODE: CONTRACT-AWARE OBSERVABLE EXECUTION

The contract-aware adapter represents function signatures, dependencies, units, edge cases, output formats, and visible verifier feedback before repair. Table 3 reports the strongest execution-oriented comparator for each backbone along with the full adapter's cost. Contract repair improves the mean execution by 5.7 points in Qwen and 3.2 in GLM (Appendix I). Thus, the full adapter supports a

Backbone	Method	Execution	Parse	Tokens	CF cost
Qwen3.5-35B-A3B	ReAct/tool coding	47.5	94.0	28649	0.0400
	Self-debug	49.5	98.0	27765	0.0396
	Code rerank	45.0	92.0	28625	0.0399
	Reflexion repair	41.3	88.0	29177	0.0412
	SCIDIALECT (visible repair)	53.5	90.0	25868	0.0362
	SCIDIALECT (contract repair)	**55.2**	98.0	31695	0.0438
GLM-4-32B-0414	ReAct/tool coding	44.8	82.0	24147	0.0619
	Self-debug	47.3	90.0	22802	0.0597
	Code rerank	52.8	86.0	24609	0.0640
	Reflexion repair	42.3	82.0	24466	0.0632
	SCIDIALECT (visible repair)	49.0	90.0	21479	0.0546
	SCIDIALECT (contract repair)	**56.0**	90.0	34964	0.0689

Table 3: SciCode contract-repair comparison on 50 held-out problems per backbone. Execution differences are percentage points; CF cost is per problem under the fixed pricing configuration. Full parse, token, and variant results are in Appendix I.

quality–cost trade-off; the lighter variant provides a lower-cost comparison, not the cost of the full result.

	MBO		NHO		SPO		TMC	
Method	SQ	AUOC	SQ	AUOC	SQ	AUOC	SQ	AUOC
Random search	50.2	43.0	93.1	80.9	71.6	59.2	43.4	35.4
Genetic algorithm	46.3	41.5	75.3	70.1	65.2	55.3	32.0	29.2
Differential evolution	48.4	42.1	73.8	67.8	54.8	48.0	37.2	32.1
Vanilla MAS	69.1	50.3	**99.8**	85.3	85.0	69.9	55.1	38.9
PIEVO	74.1	48.6	97.0	88.1	86.2	64.5	56.2	42.8
SCIDIALECT	**78.3**	**59.0**	**99.8**	**91.4**	**95.9**	**78.9**	**66.5**	**47.3**

Table 4: Twenty-seed compatible-core optimization results. Every method receives 24 objective evaluations and eight proposals per selection step. SQ is endpoint solution quality and AUOC is cumulative-best trajectory quality; higher is better.

6.4 BACKBONE-FREE COMPATIBLE-CORE OPTIMIZATION

Table 4 compares search policies and principle-state control in a backbone-free numeric harness. Every method uses the same deterministic surrogate and identical objective and proposal budgets; These controls remove model-provider variation. Compared to PIEVO, SCIDIALECT improves both SQ and AUOC on every task; Figure K.4 reports the paired gains. It also attains 100.0% validity with zero constraint violations, compared with PIEVO validity of 55.8%, 81.2%, 66.7%, and 49.6% on MBO, NHO, SPO, and TMC. The consistent trajectory and validity gains support the proposed mechanism: reconstruction and grounding gates prevent invalid principle states from consuming a sparse objective budget.

7 CONCLUSION AND LIMITATIONS

SCIDIALECT uses auditable abstraction as an intrinsic reward for selecting the scientific state that guides the next solver call. The two-part analysis explains when verified compression improves predictive log score and when the symbolic state preserves downstream decision information. The experiments support this form of control across forecasting, data discovery, coding, and controlled optimization, without requiring an external correctness score in the representation reward. Its effectiveness still depends on the source-state extractor, audit quality, and inference budget; exact recovery and predictive-code assumptions are stronger than the implemented numerical gates. Grounded compression can therefore guide reasoning before validation, but cannot replace the experiments or expert judgment needed to establish scientific truth.

AI USE STATEMENT

Generative AI assisted with the literature comparison, manuscript revision, validation of theoretical claims and proofs, and editing of the scientific figure. The authors remain responsible for verifying the assumptions, proofs, references, and final claims.

ETHICS STATEMENT

The method is intended to improve auditability by preserving variables, evidence paths, constraints, and code contracts. It does not certify scientific truth and should not replace empirical validation, domain-expert review, or safety review in high-stakes applications.

REPRODUCIBILITY STATEMENT

Section 5 states held-out splits, model identifiers, metrics, cost accounting, and budgets. Appendices A and D provide complete method details and proofs; Appendix E defines all metrics; and Appendices I and K report secondary analyses. The corresponding experiment artifact is uploaded in the supplementary material, including the anonymized code, the configuration, and several samples. No online dialect-construction step accesses hidden held-out references or hidden test scores. Visible verifier outputs and optimization objectives are used only as specified by the corresponding task adapter; they are not online correctness targets for the representation reward.

REFERENCES

Anirudh Ajith, Amanpreet Singh, Jay DeYoung, Nadav Kunievsky, Austin C Kozlowski, Oyvind Tafjord, James Evans, Daniel S Weld, Tom Hope, and Doug Downey. Prescience: A benchmark for forecasting scientific contributions. *arXiv preprint arXiv:2602.20459*, 2026.

Simon A Aytes, Jinheon Baek, and Sung Ju Hwang. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *arXiv preprint arXiv:2503.05179*, 2025.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. Codet: Code generation with generated tests. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ktrw68Cmu9c.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *International Conference on Learning Representations*, volume 2024, pp. 8746–8825, 2024.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

Nelson Cowan. The magical number 4 in short-term memory: a reconsideration of mental storage capacity. *Behavioral & Brain Sciences*, 24(1):87, 2001.

Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression. jl. *arXiv preprint arXiv:2305.01582*, 2023.

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. Language modeling is compression. In *International Conference on Learning Representations*, volume 2024, pp. 14165–14181, 2024.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning. *arXiv preprint arXiv:2006.08381*, 2020.

Team Glm, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

Gabriel Grand, Lionel Wong, Matthew Bowers, Theo X Olausson, Muxin Liu, Joshua B Tenenbaum, and Jacob Andreas. Lilo: Learning interpretable libraries by compressing and documenting code. In *The Twelfth International Conference on Learning Representations*, 2024.

Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.

Dhruv Gupta. Signed compression progress on a sealed audit is goodhart-resistant. In *COLM 2026 The 2nd Workshop on Lifelong Agents: Learning, Aligning, and Evolving*, 2026. URL https://openreview.net/forum?id=rGe63hRjKq.

John H Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. The MIT press, 1992.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29, 2016.

Sihan Hu, Lyuhan Huang, Youjin Deng, and Kun Chen. Scicode-verified: How benchmark defects underestimated the scientific-coding ability of language models. *arXiv preprint arXiv:2608.04975*, 2026.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 13358–13376, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Dennis V Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

Bodhisattwa Prasad Majumder, Harshit Surana, Dhruv Agarwal, Bhavana Dalvi Mishra, Abhijeetsingh Meena, Aryan Prakhar, Tirth Vora, Tushar Khot, Ashish Sabharwal, and Peter Clark. Discoverybench: Towards data-driven discovery with large language models. In *International Conference on Learning Representations*, volume 2025, pp. 4556–4579, 2025.

Sania Nayab, Giulio Rossolini, Marco Simoni, Andrea Saracino, Giorgio Buttazzo, Nicolamaria Manes, and Fabrizio Giacomelli. Concise thoughts: Impact of output length on llm reasoning and cost. *arXiv preprint arXiv:2407.19825*, 2024.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:108, 2007.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Zhengqi Pei, Anran Zhang, Shuhui Wang, and Qingming Huang. Modeling language tokens as functionals of semantic fields. In *International Conference on Machine Learning*, pp. 40114–40128. PMLR, 2024.

Zhengqi Pei, Qingming Huang, and Shuhui Wang. Neuronal group communication for efficient neural representation. *arXiv preprint arXiv:2510.16851*, 2025.

Zhengqi Pei, Qingming Huang, and Shuhui Wang. When LLMs develop languages: Symbolic communication for efficient multi-agent reasoning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ovpL0ujD6j.

Yingming Pu, Tao Lin, and Hongyu Chen. Piflow: Principle-aware scientific discovery with multi-agent collaboration. *arXiv preprint arXiv:2505.15047*, 2025.

Yingming Pu, Tao Lin, and Hongyu Chen. Principle-evolvable scientific discovery via uncertainty minimization. *arXiv preprint arXiv:2602.06448*, 2026.

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Jonathan Scarlett and Volkan Cevher. An introductory guide to fano's inequality with applications in statistical estimation. *arXiv preprint arXiv:1901.00555*, 2019.

Jürgen Schmidhuber. Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. In *Workshop on anticipatory behavior in adaptive learning systems*, pp. 48–76. Springer, 2008.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Zhangde Song, Jieyu Lu, Yuanqi Du, Botao Yu, Thomas M Pruyn, Yue Huang, Kehan Guo, Xiuzhe Luo, Yuanhao Qu, Yi Qu, et al. Evaluating large language models in scientific discovery. *arXiv preprint arXiv:2512.15567*, 2025.

Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Minyang Tian, Luyu Gao, Shizhuo D Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, et al. Scicode: A research coding benchmark curated by scientists. *Advances in Neural Information Processing Systems*, 37:30624–30650, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

A Complete Method Specification

This appendix expands the definitions and subroutines used by the main-text controller in Section 3. The Algorithm 1 specifies the general online controller. This appendix gives its executable interfaces, data types, score components, and calibration rules. Appendix B separately identifies the concrete configurations and recorded transitions used in the qualitative examples. Table A.1 records the correspondence between the concise main presentation and the details below.

Main-text component	Appendix detail
Executable roles and component realizations	Appendix A.1
Task, external score, source state, and frozen protocol	Appendix A.2
Evidence anchors, primitives, expressions, and decoder-visible context	Appendices A.3–A.4
Compression, reconstruction, grounding, agreement, and dictionary cost	Appendix A.7
Propose–decode–audit–gate–select–condition loop	Algorithm 1 and Appendices A.5 and A.8
Offline primitive calibration and dictionary freezing	Appendix A.6
Data-discovery, forecasting, coding, and optimization adapters	Appendices A.9–A.10
Recorded query-to-output examples	Appendix B

Table A.1: Correspondence between the main method and its formal appendix specification.

A.1 Executable Components and Their Instantiation

Controller versus model. The controller is an ordinary program that calls configured subroutines, stores their outputs, and applies fixed decision rules. A role such as "proposer" or "decoder" identifies a call contract, not a separate neural architecture. The same pretrained checkpoint can serve several roles through separate prompts and contexts. In a MAS, the solver interface can instead invoke an existing multi-agent workflow; the controller need not inspect its internal messages or weights. The recorded text examples use separate calls to one pretrained Qwen3.5-35B-A3B model, rather than distinct trained encoder, decoder, and solver networks.

What is frozen and what can change. Freezing fixes pretrained-model weights, prompts, schemas, score rules, thresholds, budgets, and persistent dictionary definitions; it does not make sampled LLM outputs deterministic. Sampling settings and seeds are part of the execution configuration, and the displayed maps suppress this randomness for readability. Source propositions, candidate expressions, decoded text, solver memory, and history are query-local data that can change at every round. In the compatible-core optimization adapter, Gaussian-process fitting and updates of the optimizer's principle hypotheses remain ordinary task-local search operations. They are not training of a neural reward model or online promotion of public dictionary primitives.

Component	Executable realization	Returned or stored object
Solver, Solve_v	A callable pretrained LLM or fixed agent/MAS workflow; its internal reasoning is outside the controller.	Response $o \in \mathcal{O}_v$: text, code, or an adapter-defined response record.
Adapter, v	Task-specific software and configuration: evidence readers, schemas, prompts, budgets, audits, and output parsing.	A fixed interface bundle, not an embedding or a neural adapter layer.
State and anchor extraction	A task parser and schema validator; a fixed pretrained-model prompt can normalize unstructured content before validation.	Ordered propositions P_t and finite anchor records with declared types and sources.
Proposer, $\mathrm{Propose}_v$	A prompted pretrained LLM, or deterministic slot binding when the public template is fixed.	At most K expression sequences and their local definitions.
Decoder, $\psi_{\mathcal{D}}^{(j)}$	A source-blinded pretrained-model call or a deterministic public-template expansion.	One reconstructed proposition per source position, not a hidden feature vector.
Audit and reward	Fixed comparison routines, arithmetic scores, and Boolean gates; semantic comparison may use a configured pretrained judge without task-specific training.	Component scores, admissibility verdict, and scalar $R_{\mathrm{SD}}(c)$.
Memory and finalization	Deterministic serialization, response parsing, and the predeclared budget-exhaustion routine.	Memory text or structured messages, a final task output, and an audit log.
Primitive dictionary, $\mathcal{D}^\star$	A finite structured registry, stored as JSON/YAML records with public definitions and decoding instructions.	Reusable symbolic data, not trainable model parameters.
Offline calibration	A prompted primitive inducer plus programmatic schema, reuse, cost, and freeze checks.	Retained primitive records; no solver-weight updates.

Table A.2: Executable interpretation of the general controller. Alternative realizations are configured before evaluation; a row does not imply that every option is used in every run. The precise choices in the recorded examples are stated in Appendix B.

A.2 Notation and Execution Phases

Symbol or function	Meaning and domain
$\tau \sim \mathcal{T}$	A scientific task sampled from the task distribution.
$\mathcal{K}_\tau = (e_1, \ldots, e_{N_\tau})$	Finite ordered sequence of text or structured evidence records, including the task instruction.
$\mathcal{Y}_\tau$; $y_\tau \in \mathcal{Y}_\tau$	Task-specific output space and a valid final output.
$R_{\mathrm{ext},\tau} : \mathcal{Y}_\tau \to \mathbb{R}$	Final external evaluator; $r_{\mathrm{ext},\tau} = R_{\mathrm{ext},\tau}(y_\tau)$ is never used by the held-out controller.
$v \in \mathcal{V}$	A program/configuration bundle in the set $\mathcal{V}$ of task adapters; it fixes the schema, prompts, scores, gates, budgets, output contract, and solver interface.
T, K, J	Maximum online rounds, candidates per round, and audit decoders or channels; all are positive integers fixed before evaluation.
H_t; $X_t = \mathcal{K}_\tau \oplus H_t$	Query-local non-final solver history and the controller-visible state at round t.
$P_t = (p_{t,1}, \ldots, p_{t,n_t})$	Ordered source propositions extracted from X_t; $n_t = \lvert P_t \rvert$.
Ω_v; $\mathcal{A}_{t,i}$	Adapter anchor schema and the nonempty finite anchor set for proposition $p_{t,i}$.
$\mathcal{D}^\star$	Persistent primitive dictionary calibrated on development tasks and frozen before held-out evaluation.
$\Delta\mathcal{D}^{\mathrm{loc}}_{t,k}$; $\mathcal{D}_{t,k}$	Candidate k's query-local definitions and the public dictionary $\mathcal{D}_{t,k} = \mathcal{D}^\star \cup \Delta\mathcal{D}^{\mathrm{loc}}_{t,k}$.
$Z_{t,k}$; $\widehat{P}_{t,k}$	Candidate expressions and their independent reconstructions.
$c_{t,k}$; $\mathcal{C}_t$; $\mathcal{F}_t$	One audited candidate, the candidate set, and the hard-gated feasible subset.
M_t; $o_{t+1} \in \mathcal{O}_v$	Selected solver-facing memory and the next response in adapter v's solver-response space.
Final_v; $\mathrm{Finalize}_v$	Frozen final-response predicate and predeclared budget-exhaustion rule.
$\mathcal{L}_\tau$	Finite sequence of per-round audit records returned with the final output; not an input to candidate selection.
$\rho, \gamma, \beta, b_{\mathcal{D}}$	Reconstruction, grounding, agreement, and dictionary-penalty gates.

Table A.3: Notation used by Section 3. Round-indexed objects are query-local unless explicitly marked persistent.

The execution boundary is summarized in Table A.4. The term *persistent* means that an object may be reused across held-out queries; *query-local* means that it is created after one query is observed and deleted after that query; and *round-local* means that it is used only within one iteration of Algorithm 1.

Phase	Operation	Output and lifetime
Setup (offline)	Fix adapter schemas, evidence policy, proposer and decoder instructions, score functions, weights, hard gates, budgets, stopping rule, and output contracts.	Frozen protocol reused across queries.
Calibration (offline)	On development tasks only, propose and audit reusable primitives and retain them under Eq. 16.	Persistent frozen dictionary $\mathcal{D}^\star$; no solver-parameter update.
Control (online)	Run Algorithm 1 using only the held-out task's visible context and query-local computation.	Final output and audit record; all internal query-local state is deleted.

Table A.4: Execution phases and state lifetimes. Held-out references, hidden tests, evaluator scores, and post-query feedback cannot update the protocol or $\mathcal{D}^\star$.

A.3 Representation and Primitive Schemas

An evidence anchor is a tuple

$$a = (t_a, v_a, r_a, q_a) \in \mathcal{A}_v, \tag{6}$$

where t_a is an anchor type, v_a is its normalized value, r_a is a pointer to its task-visible source, and q_a is its role in the adapter contract. For example, a numerical anchor may record

(`temperature`, 800 K, `table:D7`, `row:12`, `regime-threshold`). Two anchors match only after the adapter's frozen type-aware normalization, so a number with the wrong unit, a relation with the wrong direction, or a value attached to the wrong variable is not counted as recovered.

A scientific primitive s_q has the public schema

$$s_q = (\sigma_q, \eta_q, \Theta_q, \mathcal{E}_q, \mathcal{C}_q, \overline{\mathcal{C}}_q, \delta_q). \tag{7}$$

Here σ_q is the symbol; η_q is its natural-language definition; Θ_q is the typed argument schema; $\mathcal{E}_q$ lists required evidence slots; $\mathcal{C}_q$ lists validity conditions; $\overline{\mathcal{C}}_q$ lists explicitly excluded nearby meanings; and δ_q is the decoder instruction associated with the primitive. During calibration, $\mathcal{D}$ denotes the current finite dictionary. After calibration, the persistent dictionary is the frozen set $\mathcal{D}^\star = \{s_q\}_{q=1}^m$. A query-local extension $\Delta\mathcal{D}_{t,k}^{\mathrm{loc}}$ must use fresh symbols, may not redefine a symbol in $\mathcal{D}^\star$, and contains the same public fields as a persistent primitive.

Let $\mathfrak{P}_v$ be the space of individual propositions admitted by adapter v, let $\mathfrak{X}_v$ be its visible context/history-block space, and let $\mathfrak{Z}_{\mathcal{D},v}$ be the space of well-formed expressions over dictionary $\mathcal{D}$ plus optional residual natural language. The proposition extractor returns a finite sequence, $\mathrm{Prop}_v : \mathfrak{X}_v \to \bigcup_{n\geq 0} \mathfrak{P}_v^n$. The encoder and the jth reconstruction channel act proposition-wise:

$$\phi_{\mathcal{D}} : \mathfrak{P}_v \times \mathfrak{X}_v \to \mathfrak{Z}_{\mathcal{D},v}, \qquad \psi_{\mathcal{D}}^{(j)} : \mathfrak{Z}_{\mathcal{D},v} \times \mathfrak{X}_v^{\mathrm{dec}} \to \mathfrak{P}_v. \tag{8}$$

The decoder-visible space $\mathfrak{X}_v^{\mathrm{dec}}$ contains permitted evidence and every definition used by the expression, but excludes the explicit source-proposition list and proposer-only rationale. This is an information-separation condition rather than a claim of statistical independence between model parameters.

A.4 Anchor Construction and Decoder-Visible Context

The schema Ω_v specifies the admissible anchor types, normalization rules, required source pointers, and validation rules for adapter v. The extractor g_{anc} is a configured program, not a learned reward network. It reads structured fields directly and applies task-specific parsing rules; when a pretrained LLM is used to normalize free text, the program validates its returned records against Ω_v and visible source locations. The selected parser or model prompt is fixed before evaluation. This defines the typed-anchor interface; the lexical extractor in the recorded core is identified separately in Appendix B. For DiscoveryBench, the schema includes dataset paths, columns, numerical summaries, time windows, effect directions, workflow cues, and task goals. For SciCode, it includes docstrings, callable signatures, visible tests, imports, units, dependencies, edge cases, and output constraints. For PreScience, it includes authors, cited work, research objects, contribution facets, venues, dates, and novelty constraints. The validator rejects an anchor that reveals a held-out target, depends on a hidden evaluator, reduces to an unconstrained keyword, or lacks a visible source pointer.

The view constructor in Eq. 10 is deterministic once adapter v and candidate k are fixed. Formally, View_v maps (X_t, P_t) to an element of $\mathfrak{X}_v^{\mathrm{dec}}$ by withholding P_t as an explicit source-state input and removing proposer-only rationale, while retaining the task-visible evidence permitted by the anchors. The function $\mathrm{Defs}(\mathcal{D}_{t,k}, Z_{t,k})$ returns a serialized public-definition block containing the definitions, argument types, validity conditions, and excluded meanings of exactly the primitives referenced by $Z_{t,k}$. This withholds the explicit source list and private proposer trace, but the permitted evidence may still reveal equivalent content. The audit therefore tests recoverability conditional on that evidence, not information uniquely carried by the expression. Separate decoder calls also do not supply an independent verification sample for the predictive generalization theorem.

The recovery routine $\mathrm{Recover}_v(z, \hat{p}, \mathcal{K}^{\mathrm{dec}})$ returns a subset of $\mathcal{A}_v$ after applying the same type-aware normalization used by the anchor extractor. It counts an anchor only when the recovered value, role, relation direction, unit or contract field, and visible provenance satisfy the adapter's frozen comparison rule. Equation 11 then averages this recovery fraction across all propositions and audit decoders.

A.5 Round-Level Construction and State Transitions

This subsection expands the operations in Algorithm 1 without changing their order or adding an optimization stage. The domains of the proposition, context, and expression spaces are given in

Appendix A.3. For a nonempty source sequence P_t, the frozen extractor returns

$$\mathcal{A}_{t,i} = g_{\text{anc}}(p_{t,i}, X_t; \Omega_v) \subseteq \mathcal{A}_v, \qquad 1 \le |\mathcal{A}_{t,i}| < \infty. \tag{9}$$

The adapter schema Ω_v fixes permitted anchor types and normalization rules, and $\mathcal{A}_v$ is its universe of valid typed anchors. Every anchor carries a visible source pointer and evidential role. If typed extraction returns no anchor, a verbatim-content anchor retains the full proposition, including its status as a requirement, assumption, or observation. Hidden targets cannot supply anchors; permitted visible verifier responses may do so only in adapters that explicitly expose them.

For $K_t \in \{0, \dots, K\}$ proposals, candidate k uses $\mathcal{D}_{t,k} = \mathcal{D}^\star \cup \Delta\mathcal{D}^{\text{loc}}_{t,k}$ and expressions $Z_{t,k} = (z_{t,1,k}, \dots, z_{t,n_t,k})$, with $z_{t,i,k} = \phi_{\mathcal{D}_{t,k}}(p_{t,i}, X_t)$. Its reconstruction inputs and outputs are

$$\begin{aligned} \mathcal{K}^{\text{dec}}_{t,k} &= \text{View}_v(X_t, P_t) \oplus \text{Defs}(\mathcal{D}_{t,k}, Z_{t,k}), \\ \hat{p}^{(j)}_{t,i,k} &= \psi^{(j)}_{\mathcal{D}_{t,k}}(z_{t,i,k}, \mathcal{K}^{\text{dec}}_{t,k}), \qquad 1 \le i \le n_t, \quad 1 \le j \le J. \end{aligned} \tag{10}$$

Here View_v withholds the explicit source-state list and proposer-only rationale, and Defs serializes the public definitions actually used. The reconstructions form $\widehat{P}_{t,k} = (\hat{p}^{(j)}_{t,i,k})_{i,j}$, and $c_{t,k} = (Z_{t,k}, \widehat{P}_{t,k}, \Delta\mathcal{D}^{\text{loc}}_{t,k})$ is the audited candidate. For $J = 1$, the adapter fixes $w_a = \beta = 0$ and omits agreement comparisons.

Let $\text{Recover}_v(z, \hat{p}, \mathcal{K}^{\text{dec}}) \subseteq \mathcal{A}_v$ return the anchors recoverable under the adapter's type-aware comparison rule. The grounding score is

$$S_{\text{ground}}(c_{t,k}) = \frac{1}{n_t J} \sum_{i=1}^{n_t} \sum_{j=1}^{J} \frac{|\mathcal{A}_{t,i} \cap \text{Recover}_v(z_{t,i,k}, \hat{p}^{(j)}_{t,i,k}, \mathcal{K}^{\text{dec}}_{t,k})|}{|\mathcal{A}_{t,i}|} \in [0,1]. \tag{11}$$

The remaining component functions are defined in Appendix A.7 and combined by Eq. 1. Let $\rho, \gamma, \beta \in [0,1]$ be the frozen reconstruction, grounding, and agreement thresholds, and let $b_{\mathcal{D}} \ge 0$ bound the dictionary penalty. The Boolean validator $V_v(c) \in \{0,1\}$ checks permitted evidence, leakage, and the adapter's output contract. Then

$$\mathcal{C}_t = \{c_{t,k} : 1 \le k \le K_t\}, \qquad K_t \le K, \tag{12}$$

$$\begin{aligned} \mathcal{F}_t = \{c \in \mathcal{C}_t : S_{\text{rec}}(c) \ge \rho,\ S_{\text{ground}}(c) \ge \gamma,\ S_{\text{agree}}(c) \ge \beta, \\ B_{\text{dict}}(c) \le b_{\mathcal{D}},\ V_v(c) = 1\}. \end{aligned} \tag{13}$$

These conditions are numerical audit gates, not logical certificates. In particular, $\gamma < 1$ permits some anchor loss, and $\gamma = 1$ still depends on the correctness of extraction and recovery.

Selection and memory construction are

$$c^\star_t = \begin{cases} \arg\max_{c \in \mathcal{F}_t} R_{\text{SD}}(c), & \mathcal{F}_t \ne \varnothing, \\ \varnothing, & \mathcal{F}_t = \varnothing, \end{cases} \tag{14}$$

$$M_t = \text{Mem}_v(c^\star_t), \qquad \text{Mem}_v(\varnothing) = \varnothing. \tag{15}$$

Ties use the adapter's deterministic rule. The memory renderer includes the expressions, recovered meanings, relevant definitions, provenance, and audit verdict. Equation 2 gives the next solver call. The frozen predicate $\text{Final}_v : \mathcal{O}_v \to \{0,1\}$ checks whether the response parses as a final output, not whether it is correct. If it does not, $H_{t+1} = H_t \oplus o_{t+1}$. After T rounds, $\text{Finalize}_v(\tau, \mathcal{K}_\tau, H_T) \in \mathcal{Y}_\tau$ applies the predeclared budget-exhaustion rule. All paths save the audit log, discard query-local state, and leave $\mathcal{D}^\star$ unchanged.

A.6 Offline Primitive Calibration and Examples

A persistent primitive is created only during development-time calibration. A pretrained LLM serves as the primitive inducer under a fixed prompt: it checks an active or pending definition before proposing a new public schema. A registration program parses the proposal, rejects malformed or conflicting definitions, and tracks distinct-task reuse and charged cost. Separate reconstruction calls supply the audit evidence for retention; the store updates records, not neural weights. The calibration procedure applies the following stages:

1. **Proposal:** identify a proposition pattern that recurs across development tasks.
2. **Canonicalization:** normalize its name and argument order, merge synonyms, and reject vague labels such as "important relation."
3. **Evidence binding:** assign every required evidence slot to an anchor type in Ω_v.
4. **Independent reconstruction:** encode representative uses and reconstruct them without exposing the source propositions.
5. **Audit:** test missing anchors, added assumptions, invalid arguments, ambiguity, unsupported evidence, and overlap with existing primitives.
6. **Reuse accounting:** compare cumulative expression savings with the primitive's definition and verification cost.
7. **Retention:** retain, merge, specialize, or reject the primitive before the dictionary is frozen.

For a proposed primitive s, let $N_{\mathrm{dev}}(s) \in \mathbb{N}$ be its number of independent development-task reuses, let $\Delta L_{\mathrm{dev}}(s) \geq 0$ be its cumulative expression-length saving, and let $C_{\mathcal{D}}(s) \geq 0$ be its definition and verification cost. The development audit uses at least two reconstruction channels when the agreement threshold $\beta_{\mathcal{D}}$ is active. The primitive is retained only if

$$\begin{gathered} N_{\mathrm{dev}}(s) \geq N_{\min}, \qquad \Delta L_{\mathrm{dev}}(s) - C_{\mathcal{D}}(s) > 0, \\ \overline{S_{\mathrm{ground}}}_{\mathrm{dev}}(s) \geq \gamma_{\mathcal{D}}, \qquad \overline{S_{\mathrm{agree}}}_{\mathrm{dev}}(s) \geq \beta_{\mathcal{D}}, \qquad \mathrm{Redundancy}(s, \mathcal{D}) < \delta_{\mathcal{D}}. \end{gathered} \tag{16}$$

Here $N_{\min} \in \mathbb{N}_+$ is the minimum reuse count, the overbars are means over audited development uses, $\gamma_{\mathcal{D}}, \beta_{\mathcal{D}} \in [0, 1]$ are dictionary-level quality thresholds, and $\delta_{\mathcal{D}} \geq 0$ is the maximum accepted redundancy. The resulting dictionary is renamed $\mathcal{D}^\star$ and frozen before held-out evaluation. A query-local definition created during held-out inference is never evaluated for promotion.

A data-driven primitive can be written as

```
EvidencePath(dataset, variables, effect_direction,
             numeric_digest, workflow)
```

and means that a claim is supported by a named dataset, specified variables, an effect direction, numerical evidence, and a plausible analysis workflow. Its excluded meanings cover uses with unnamed variables, absent numerical evidence, or a direction inferred only from background knowledge. A second primitive,

```
RegimeSwitch(response, control, low_regime,
             high_regime, transition_evidence)
```

represents a change in the dominant response across regimes of a control variable and requires explicit transition evidence; it is not a generic synonym for nonlinearity.

Data-driven discovery example. A sociology task may ask how academic ability changes the effect of socioeconomic status (SES) on an academic outcome. The anchor set contains the SES variable, ability variable, outcome, regression context, interaction sign, comparison direction, and source fields. The primitive `InteractionModeration(effect, moderator, sign, outcome, evidence)` is valid only when the independently reconstructed state recovers those anchors and preserves the interaction direction.

Scientific coding example. A SciCode-style task may ask for a numerical method whose function maps an input array and physical constants to an output under a specified unit convention. The anchors include the function name, argument order, return type, imports, unit assumptions, visible tests, edge cases, and output shape. The primitive `NumericalKernel(signature, invariants, boundary_case, tests)` is rejected if it preserves the algorithmic idea but changes the callable contract.

Scientific forecasting example. A PreScience task conditions on authors and prior work and requests a plausible future title and abstract. Anchors include author identities, cited work, method family, research object, novelty facet, venue or date constraints, and expected contribution type. The primitive `ContributionMove(prior_work, method, object, novelty, evidence)` is valid only if the forecast remains tied to the supplied scholarly context.

A.7 Exact Intrinsic-Reward Components

Fix one online round t and candidate k, and suppress their indices in this subsection by writing $P = P_t = (p_i)_{i=1}^n$, $\mathcal{A}_i = \mathcal{A}_{t,i}$, $Z = Z_{t,k} = (z_i)_{i=1}^n$, $\widehat{P} = \widehat{P}_{t,k} = \{\hat{p}_i^{(j)}\}_{i,j}$, $\Delta\mathcal{D} = \Delta\mathcal{D}_{t,k}^{\mathrm{loc}}$, and $c = c_{t,k}$. This convention is the exact correspondence between the shorthand in Eq. 1 and the formulas below. Let $L(x) \geq 0$ be the fixed implementation length of the canonical serialization of object x, measured with the same tokenizer or coding rule for every compared candidate; for a definition set, the serialization includes every public field required by its primitives. Let $\epsilon > 0$ be a fixed numerical stabilizer. The exact prediction-as-bits interpretation in Section 4 applies when L is a proper coding length; otherwise it remains an implementation description-length score.

The charged compression gain is

$$G_{\mathrm{comp}}(c) = \frac{\sum_{i=1}^n L(p_i) - \sum_{i=1}^n L(z_i) - \alpha L(\Delta\mathcal{D})}{\sum_{i=1}^n L(p_i) + \epsilon}, \tag{17}$$

where $\alpha \geq 0$ is the frozen charge applied to query-local definitions. The score may be negative when the candidate plus its definitions is longer than the source state. G_{comp} and B_{dict} are arithmetic functions of serialized data and registry counts; neither invokes an external task evaluator.

The reconstruction score is

$$S_{\mathrm{rec}}(c) = \frac{1}{nJ} \sum_{i=1}^n \sum_{j=1}^J \mathrm{Fidelity}_v\left(p_i, \hat{p}_i^{(j)}, \mathcal{A}_i\right) \in [0,1], \tag{18}$$

where $\mathrm{Fidelity}_v(p, \hat{p}, \mathcal{A}) \in [0,1]$ is adapter v's frozen anchor-sensitive semantic comparison between two individual propositions under required anchor set $\mathcal{A}$. It penalizes missing anchors, relation reversals, unsupported assumptions, incompatible units or contracts, and other semantic drift. The adapter fixes the comparison routine: structured fields can be compared programmatically, whereas open-text equivalence can use a separately prompted pretrained judge with a fixed rubric. Such a judge compares source and reconstruction only; it does not receive a target answer or predict final task success. This is an interface option, not an additional model call claimed for the deterministic lexical audits in Appendix B. The grounding score $S_{\mathrm{ground}}(c)$ is defined by Eq. 11; reconstruction compares proposition meaning, whereas grounding checks recoverable task-visible evidence.

For $J \geq 2$, decoder agreement is

$$S_{\mathrm{agree}}(c) = \frac{2}{nJ(J-1)} \sum_{i=1}^n \sum_{1 \leq j < j' \leq J} \mathrm{Sim}_v\left(\hat{p}_i^{(j)}, \hat{p}_i^{(j')}\right) \in [0,1], \tag{19}$$

where $\mathrm{Sim}_v(\hat{p}, \hat{p}') \in [0,1]$ is a frozen semantic-equivalence score between two reconstructed propositions. For an adapter with one deterministic audit channel, we set $J = 1$, $S_{\mathrm{agree}}(c) = 0$, $w_a = 0$, and $\beta = 0$, so the undefined pairwise term is neither divided by zero nor used for ranking or gating.

The dictionary penalty is the nonnegative function

$$B_{\mathrm{dict}}(c) = \lambda_1 |\Delta\mathcal{D}|_{\text{one-off}} + \lambda_2 L(\Delta\mathcal{D}) + \lambda_3 \,\mathrm{Redundancy}(\mathcal{D}^\star \cup \Delta\mathcal{D}), \tag{20}$$

where $|\Delta\mathcal{D}|_{\text{one-off}} \in \mathbb{N}_0$ counts local definitions without independent reuse, $\mathrm{Redundancy}(\mathcal{D}^\star \cup \Delta\mathcal{D}) \in [0, \infty)$ is a frozen measure of duplicate or near-duplicate definitions, and $\lambda_1, \lambda_2, \lambda_3 \geq 0$ are fixed penalties. The gate $b_{\mathcal{D}}$ in Eq. 13 is therefore a maximum permitted value of the complete dictionary penalty, not merely a token budget.

All score functions, weights, thresholds, normalizers, and prompts are frozen before held-out evaluation. Development-task outcomes may be consulted only for offline primitive retention and adapter calibration; they are not additional terms in the online reward.

A.8 Controller Subroutines, Memory, and Audit Log

Algorithm 1 invokes the following frozen adapter subroutines. They are callable script-based interfaces whose model-backed or deterministic realizations are identified in Appendix A.1. Their contracts complete the operational specification without duplicating the main control flow.

- $\mathrm{Prop}_v(X_t)$ returns the ordered source proposition list $P_t \in \bigcup_{n\geq 0} \mathfrak{P}_v^n$.
- $\mathrm{Propose}_v(P_t, X_t; \mathcal{D}^\star, K)$ returns a finite set of at most K pairs $(Z_{t,k}, \Delta\mathcal{D}_{t,k}^{\mathrm{loc}})$ and cannot access hidden references, tests, or external scores.
- View_v and Defs construct the blinded decoder context in Eq. 10.
- Audit_v invokes the J reconstruction channels, computes Eqs. 11 and 1, applies V_v, and returns the fully audited set $\mathcal{C}_t$.
- Mem_v renders only the selected candidate's public expressions, recovered meanings, anchor provenance, required definitions, and audit verdict; $\mathrm{Mem}_v(\varnothing) = \varnothing$ gives the exact direct-context fallback.
- Solve_v is the fixed black-box solver call with codomain $\mathcal{O}_v$; $\mathrm{Final}_v : \mathcal{O}_v \to \{0, 1\}$ and the frozen adapter parser determine whether a response is well formed and parseable as some $y_\tau \in \mathcal{Y}_\tau$, not whether it is correct.
- $\mathrm{Finalize}_v(\tau, \mathcal{K}_\tau, H_T) \in \mathcal{Y}_\tau$ is the predeclared budget-exhaustion rule (for example, parsing the latest valid answer or using a budgeted direct-context finalization call). It cannot update $\mathcal{D}^\star$ or inspect information unavailable to the ordinary solver interface.

The query audit log $\mathcal{L}_\tau$ records, for every round and candidate, the source propositions, anchor sets, expressions, local definitions, decoder-visible context identifier, reconstructions, component scores, hard-gate outcomes, selected memory, solver response, and final/continuation decision. The log is an output-only observability artifact: it does not feed back into candidate construction or selection unless the same task-facing information has independently entered H_t under the frozen adapter contract. After the record is written, all other query-local objects are discarded, and the persistent dictionary remains unchanged throughout Algorithm 1.

A.9 Task-Adapter Details and Budgeted Realizations

A task adapter is a fixed script layer between the controller and an existing task interface. For data discovery, dataset readers and metadata parsers expose allowed tables, column descriptions, summaries, and evidence paths; a configured LLM call can turn this material into source propositions or a final hypothesis. For contribution forecasting, scholarly-record readers expose the supplied authors, prior work, dates, and contribution constraints; an encoding prompt binds these fields to public primitives before the solver generates title–abstract text. For scientific coding, a contract parser identifies the function signature, inputs, outputs, dependencies, and visible requirements; the code-generating solver and execution harness then operate under that contract. None of these readers supplies a gold hypothesis, a future target contribution, or unavailable reference-test values. A tool result can enter the visible state only if that adapter actually obtains and exposes it.

A one-pass realization sets $T = 1$ and may use $K = 1$. With one deterministic reconstruction channel it uses $J = 1$ and $w_a = \beta = 0$ as specified in Appendix A.7. With several candidates or decoders, the same control interfaces support bounded generation and separate reconstruction calls. When there is only one proposal, the substantive control decision is admission rather than a comparison among proposals. No unrecorded alternative candidates should be inferred from the general argmax notation. Section 5 and the experimental appendices specify the reported task-facing configurations and costs.

A contract-repair realization alternates a code-generating solver with a programmatic visible execution harness within its fixed budget. The resulting feedback is a task-local observation, not an extra term in Eq. 1. In the retained coding example, a contract route and a fallback route are compared using their observable execution scores; this is output-route selection within the coding adapter, distinct from intrinsic ranking of representations. Its planning expression is not presented as a source-blind two-decoder audit (Appendix B.5). Similarly, the complete four-call text records use the simplified single-candidate lexical configuration explicitly documented in Appendix B.1. Their qualitative traces do not replace the benchmark-specific configurations or establish every guarantee of the general controller.

A.10 Formal Principle-Evolution Integration

The compatible-core optimization adapter preserves the original optimizer and changes only how a candidate principle is admitted and scored. Its underlying solver is a numerical search program,

with Gaussian-process prediction and a fixed acquisition routine in the supplied compatible-core realization, rather than a pretrained LLM. A principle is a structured search hypothesis with its associated state, not a neural module; new task-local principles do not modify the frozen primitive dictionary. The following equations specify the general integration interface, not a claim that a proxy diagnostic is an independently decoded measurement. At optimization round t, let h be a candidate hypothesis, let p be a candidate principle, and let $u_t^{\mathrm{PIEVO}}(h,p) \in \mathbb{R}$ be the optimizer's original acquisition utility. The adapter encodes p as $z(p)$, obtains J reconstructions, binds its anchors, and forms the ordinary SCIDIALECT candidate

$$c_t(p) = \big(\{z(p)\}, \{\hat{p}^{(j)}\}_{j=1}^{J}, \Delta\mathcal{D}_t^{\mathrm{loc}}(p)\big). \tag{21}$$

Let $\mathrm{Viol}_v(p) \in \mathbb{N}_0$ count principle-level violations of the optimizer's fixed scientific constraints. The principle is admissible exactly when it passes the same candidate gates as the main controller and has no principle-level violation:

$$\mathrm{Accept}_t(p) = \mathbf{1}\{c_t(p) \in \mathcal{F}_t \text{ and } \mathrm{Viol}_v(p) = 0\}\,. \tag{22}$$

At the level of this general interface, admissibility uses the same reconstruction, grounding, agreement, dictionary, leakage, and contract gates. A concrete proxy realization must separately identify any diagnostic that substitutes for an independently measured audit score.

For an admissible pair (h,p), let $\mathrm{Viol}_v(h,p) \in \mathbb{N}_0$ count candidate-level violations introduced by applying p to h. The integrated acquisition score is

$$\begin{aligned}\widetilde{u}_t(h,p) = u_t^{\mathrm{PIEVO}}(h,p) + \lambda_g S_{\mathrm{ground}}(c_t(p)) + \lambda_r S_{\mathrm{rec}}(c_t(p)) \\ - \lambda_v\,\mathrm{Viol}_v(h,p) - \lambda_b B_{\mathrm{dict}}(c_t(p))\,,\end{aligned} \tag{23}$$

where $\lambda_g, \lambda_r, \lambda_v, \lambda_b \geq 0$ are fixed before evaluation. If $\mathrm{Accept}_t(p) = 0$, the pair is removed before acquisition ranking. The dialect terms therefore filter and regularize the optimizer's existing acquisition policy; they do not replace its external objective or assert that an accepted principle is scientifically true.

B Worked Examples: From a Query to a Scientific Output

We present four samples through concrete operations that connect an input query to its final output. The first two use the contribution-forecasting interface of PreScience (Ajith et al., 2026); the third concerns a DiscoveryBench question (Majumder et al., 2025); the fourth is a SciCode programming task (Tian et al., 2024). The first three are development mechanism examples with complete encoder–decoder–solver records, not additional confirmatory benchmark results. The coding record retains the final program and route feedback, but not every intermediate draft. We simplify surrounding explanations, preserve quoted expressions and recorded scores, and distinguish a generated forecast or hypothesis from a validated discovery. Line breaks in expression listings are editorial; byte lengths refer to the original serialized strings.

B.1 Reading guideline

Actors and state. For Examples B.2–B.4, ordinary Python code extracts task-specific lexical anchors and loads the same frozen dictionary of 24 primitives: 15 seed schemas and nine templates retained during calibration. One Qwen3.5-35B-A3B checkpoint serves four distinct call roles: an encoder, decoder A, decoder B, and a final solver. The encoder receives the original input, the required anchors, and the public definitions. Each decoder receives only the expression and dictionary, without the original input or the encoder's rationale. The calls use temperatures 0.04, 0.04, 0.12, and 0.06, respectively, in these accepted runs. Their separation is an information-access restriction, not a claim of independent model errors. No new primitive is added during any of these queries.

Actual admission rule. These runs use one candidate and two decoders, so the controller makes an admission decision rather than comparing multiple candidates by Eq. 1. Its deterministic checks require at least 0.05 normalized UTF-8 byte reduction, 0.40 lexical-anchor recovery over the union of the reconstructions, and 0.30 decoder agreement. Agreement is the average of token-set F1 between the decoded texts and Jaccard overlap between their recovered anchor sets. Table B.5 reports the recorded values. The byte count charges the expression and any declared local definitions; local definitions are empty in these runs. The shared dictionary is provided to both decoders and is not counted as a newly introduced definition for each query.

Record	Byte reduction	Union recovery	Agreement	Admitted
Glass forecasting	0.873684	0.916667	0.962366	Yes
Gene-network forecasting	0.885017	0.750000	0.895349	Yes
Replication geography	0.972574	1.000000	1.000000	Yes
Admission threshold	0.05	0.40	0.30	—

Table B.5: Recorded audits for the single-candidate qualitative configuration. These lexical, union-based quantities are not substituted for the typed, per-decoder scores in Eqs. 11–13. Passing these checks does not certify exact semantic recovery or full primitive-schema compliance.

Connection to the general method. The source object is the full task input in this configuration, rather than a separately extracted proposition list. The expressions and their arguments are shown as emitted, including incomplete bindings relative to the public schemas. The recorded gates check lexical recovery and agreement, not every semantic or typing condition of the general controller. All three candidates pass, and the solver receives the original input, expression, both reconstructions, and numerical audit verdict. These traces therefore show the accepted-state data flow; they do not exercise the general empty-memory rejection path or establish losslessness. They are also distinct from the benchmark-specific adapters that produced the main result tables.

B.2 Example 1: Forecasting a Contribution in Glass Physics

Input and requested output. The query asks for a future paper title and concise abstract from supplied prior work on amorphous solids and structural glasses. One visible reference connects weakly connected structures to excess low-frequency vibrational modes; another supplies glass-transition theory. The task includes a target date, author context, and subject categories, but no future

target title or abstract is supplied to the controller. The output is a contribution forecast, not a report of a newly executed physical experiment.

Source state and encoding. The program extracts 12 lexical anchors from this input, including `low-frequency`, `vibrational`, `solids`, `Glasses`, and `Transition`. The encoder is instructed to form a public contract before solving, and returns the following expression:

```
MethodExtension(Theory, cond-mat.dis-nn, cond-mat.stat-mech)
| MechanismContrast(weakly connected solids,
    continuous elastic body, vibrational modes, low-frequency)
| LatentMediator(geometry, excess low-frequency modes, Glasses)
```

The three relation labels organize the context as a method extension, a mechanism contrast, and a proposed mediating relation. They are representational choices made by the encoder; their appearance is not new evidence that the proposed physical relations are true. The public dictionary supplies the meanings of the symbols rather than requiring the solver to infer a private code.

Reconstruction and decision. Decoder A expands the first two expressions into an extension of the theory to the named subject areas and a comparison of weakly connected solids with continuous elastic bodies in the low-frequency regime. For the last expression it assigns geometry to the input, excess low-frequency modes to the mediator, and glasses to the output. Decoder B independently returns the same assignments with different wording. The source input occupies 1,805 UTF-8 bytes and the expression 228 bytes, yielding the recorded reduction of 0.873684. Eleven of the twelve lexical anchors are recovered; `Transition` is absent from both decoded texts. Union recovery of 0.916667 and agreement of 0.962366 pass this configuration's thresholds, so its sole candidate is admitted. This is an approximate audit, not full preservation of every source commitment.

Solver use and final output. The final solver call receives the original scholarly context together with the admitted expression, both reconstructions, and the gate verdict. Its recorded title is:

> Geometry of the glass transition: From weakly connected solids to the vibrational spectrum

The generated abstract proposes connecting structural geometry, mechanical stability, and vibrational anomalies to glass-transition behavior. Although the forecast uses publication-style language such as "We demonstrate," the record contains no corresponding experiment or derivation. The example shows how the representation places particular scientific relations in the next solver call; it does not establish that the forecast is novel, correct, or caused by the representation alone.

B.3 Example 2: Reusing the Dictionary for Gene-Network Forecasting

Input and requested output. A second query supplies prior work on ARACNE and requests a future paper title and concise abstract. The visible context concerns reconstructing gene regulatory networks in mammalian cells from expression profiles using an information-theoretic approach. The same frozen dictionary is reused; the change in scientific domain does not trigger dictionary calibration or a model-weight update.

Source state and encoding. The lexical extractor identifies 12 anchors, including `ARACNE`, `Gene`, `Regulatory`, `Networks`, `Mammalian`, and `Reconstruction`. The encoder returns:

```
MethodExtension(ARACNE, Mammalian Cellular Context)
| EvidencePath(Elucidating Gene Regulatory Networks,
    ARACNE Algorithm, q-bio.GN)
```

The method name, biological setting, and network objective now fill the expression arguments. Unlike a task-specific opaque abbreviation, `MethodExtension` retains its public dictionary definition across this query and Example B.2. This reuse concerns the symbolic template, not a finding that the proposed extension is scientifically new.

Reconstruction and decision. Decoder A returns the following contract:

> The ARACNE algorithm is extended to the Mammalian Cellular Context to elucidate Gene Regulatory Networks, supported by evidence from the q-bio.GN domain.

Decoder B likewise preserves ARACNE, mammalian cellular context, and gene regulatory networks, while describing the intended output as novel insights. The 1,148-byte source is encoded in 132 bytes. The controller records byte reduction 0.885017, union recovery 0.750000, and agreement 0.895349, and admits the candidate. The lexical routine does not recover `Reconstruction`, `Background`, or `Elucidating`; it does not count *elucidate* as the literal anchor *Elucidating*. Thus, the reported score reflects the actual comparison rule rather than a retrospectively improved semantic judgment.

Solver use and final output. After receiving the original context and admitted memory, the solver proposes a paper titled:

> ARACNE-NG: A Novel Algorithm for the Reconstruction of Gene Regulatory Networks in Mammalian Cellular Contexts Using Information-Theoretic Approaches

The abstract describes an information-theoretic network-deconvolution method based on microarray expression profiles. This is the generated forecast; the trace contains no ARACNE-NG implementation, biological experiment, or validation of novelty. Together with Example B.2, it makes the two lifetimes concrete: primitive definitions persist across queries, while their arguments, reconstructed meanings, and final outputs are query-local.

B.4 Example 3: Preserving a Data-Discovery Question About Replication Geography

Input and requested output. The query asks:

> In which country's labs were the majority of original studies in both Experimental Economics and Psychology conducted?

The supplied context describes two datasets, distinguishes original-study from replication fields, and includes matching and deduplication guidance. The required answer concerns the location of the *original* studies in both domains, not the location of the replications. The retained four-call run contains dataset descriptions, not an executed table-aggregation transcript.

Source state and encoding. The deterministic extractor retains seven anchors: `country`, `labs`, `majority`, `original`, `Experimental`, `Economics`, and `Psychology`. The encoder returns a primitive followed by residual explanatory text:

```
EvidencePath(country, labs, majority) where {original}
studies in {Experimental} {Economics} and {Psychology}
are aggregated by {country} to identify the {majority}
location of {labs}.
```

The residual text preserves the population, grouping variable, and majority condition without requiring additional primitive definitions. It records an aggregation requirement; it is not a log that the aggregation has already been executed.

Reconstruction and decision. Both decoder calls return the same text:

> The {majority} location of {labs} is identified by aggregating {original} studies in {Experimental} {Economics} and {Psychology} by {country}.

The source occupies 6,709 UTF-8 bytes and the expression 184 bytes. The controller obtains byte reduction 0.972574, recovery of all seven selected lexical anchors, and agreement 1.000000. The sole candidate passes the gates and is supplied to the final solver alongside the original context. Full recovery of this small anchor set does not imply recovery of every dataset field or every requirement in the original input.

Solver use and final output. The recorded answer is:

> The majority of original studies in both Experimental Economics and Psychology were conducted in the USA.

This is a generated hypothesis, not a country count verified by the recorded calls. The expression identifies what the question asks and what an analysis would need to preserve, but it does not

itself provide evidence for the returned country. No CSV execution, deduplicated study counts, or numerical country proportions are claimed for this run. A data-executing adapter could add such observations to a subsequent visible state; that additional execution is not part of this record.

B.5 Example 4: Turning a Scientific Contract into a Weighted Jacobi Program

Input and required output. The task requests `WJ(A, b, eps, x_true, x0, omega)` for weighted Jacobi iteration. Here $A \in \mathbb{R}^{N\times N}$ is the system matrix; $b, x_{\text{true}}, x_0 \in \mathbb{R}^N$ are the right-hand side, reference solution, and initial iterate; $\epsilon > 0$ is the increment tolerance; and $\omega \in \mathbb{R}$ is the relaxation parameter, with $N \in \mathbb{N}_+$. The function must return the two scalar norms $\|Ax - b\|_2$ and $\|x - x_{\text{true}}\|_2$ after the increment falls below ϵ. The reference vector x_{true} is a declared function argument, distinct from the unavailable hidden target values in the evaluation harness.

Representation and concrete adapter. This run uses the contract-repair adapter with Qwen3.5-35B-A3B code generation and a programmatic execution harness. The saved representation is a symbolic planning checklist with an accompanying decoded contract, not the source-blind two-decoder configuration above. For readability, the following are selected entries from that checklist, not a replacement expression:

```
MechanismContrast(WJ, Jacobi)
BottleneckShift(IncrementNorm)
EvidencePath(Residual, Error)
MeasurementConfound(Increment, Residual)
```

The retained decoded contract specifies initialization from `x0`, diagonal-preconditioned updates, stopping on the increment norm, and returning residual and error norms. This distinction matters: replacing the requested increment test by a residual test would change the task contract. With $D = \operatorname{diag}(A)$, the update implemented by the saved program is

$$x^{k+1} = x^k + \omega D^{-1}(b - Ax^k), \qquad \|x^{k+1} - x^k\|_2 < \epsilon \text{ at termination}, \tag{24}$$

where k is the iteration index and the diagonal entries must be nonzero. This equation describes the recorded implementation; it is not a new guarantee that arbitrary inputs converge.

Generation, feedback, and route selection. The retained record shows a repair attempt and the feedback category `hidden_target_unavailable`. This denotes missing reference values, not a demonstrated error in the numerical update or the function name. The contract and fallback routes each receive observable execution score 1.0, and the saved route is `contract`. The supplied walkthrough describes a policy that switches to fallback only for a strictly higher observable score, consistent with retaining the contract route on this tie. This is output-route selection using permitted execution feedback, not an intrinsic representation-reward comparison. Intermediate code drafts absent from the saved row are not reconstructed as recorded steps.

Final executable output. The saved program is shown below with comments and an unused local variable omitted; its numerical operations, guard, stopping condition, and returns are unchanged.

```python
import numpy as np

def WJ(A, b, eps, x_true, x0, omega):
    x = x0.copy()
    diag = np.diag(A)
    if np.any(diag == 0):
        raise ValueError(
            "Diagonal elements cannot be zero for Jacobi method")
    inv_diag = 1.0 / diag
    while True:
        x_old = x.copy()
        Ax = A @ x_old
        residual_vec = b - Ax
        x = x_old + omega * (residual_vec * inv_diag)
        if np.linalg.norm(x - x_old) < eps:
            break
    residual = np.linalg.norm(A @ x - b)
    error = np.linalg.norm(x - x_true)
    return residual, error
```

Observed outcome and its interpretation. All six archived target-free invocations complete, covering three public input setups repeated twice. The harness removes assertions that depend on unavailable target values, so this is six successful executions, not six hidden numerical-correctness passes. The supplied local verification also compares the saved code with an independently implemented iteration on three convergent systems and checks rejection of a zero diagonal; these are separately identified checks, not original hidden-test results. The generated loop has no iteration cap and therefore requires convergent inputs for termination; local verification uses a process timeout. This example shows how a scientific representation can make the callable and numerical contract explicit before generation and repair, while retaining the distinction between contract preservation, executable output, and external scientific correctness.

B.6 What These Examples Establish

Across the recorded text queries, the data flow is explicit: task-visible context determines anchors and a public expression; separate decoders recover a readable contract; programmatic checks decide admission; and the final solver receives that representation together with the original input. The coding example instead exposes contract-oriented generation and bounded feedback through a different adapter. These records illustrate implementation roles, information access, dictionary reuse, and task-specific outputs. They neither provide missing scientific validation nor establish a causal performance gain from an individual selected trajectory. The main experiments provide aggregate comparisons; the theoretical statements require the assumptions stated in Section 4.

C Additional Predictive Identities and Evidence Acquisition

This appendix expands the predictive part of Section 4. It records an autoregressive identity, the induced expected-reward ranking, and a separate evidence-acquisition extension. These are properties of the stated coding objectives, not additional modules evaluated in the experiments.

Let $S = (S_1, \ldots, S_m)$ denote a *canonical grounded scientific state*, obtained after removing surface-form differences while retaining the same propositions, relations, assumptions, and required evidence anchors. Conditioned on visible context $\mathcal{K}$, let representation d induce a normalized coding distribution $q_d(S \mid \mathcal{K})$ and ideal codelength

$$\ell_d(S \mid \mathcal{K}) = -\log_2 q_d(S \mid \mathcal{K}). \tag{25}$$

Let $C(d)$ be the number of bits charged for every persistent and local definition required to interpret d. Net progress r_{dial} is defined in Eq. 3. The unknown record distribution is used only to characterize expectation, not to compute the realized coding reward.

Theorem 3 (Scientific-state prediction–compression equivalence)**.** *If $q_d(S \mid \mathcal{K}) = \prod_{j=1}^{m} q_d(S_j \mid S_{<j}, \mathcal{K})$ and likewise for $q_{d'}$, then*

$$r_{\text{dial}} = \sum_{j=1}^{m} \log_2 \frac{q_{d'}(S_j \mid S_{<j}, \mathcal{K})}{q_d(S_j \mid S_{<j}, \mathcal{K})} - \big(C(d') - C(d)\big). \tag{26}$$

Thus, on the same grounded state, each verified bit saved is exactly one bit of improved autoregressive log score after representation cost.

Corollary 1 (Reward-conditioned state selection)**.** *For a fixed finite set $\mathfrak{F}_t = \{d(c) : c \in \mathcal{F}_t\}$ of code schemes induced by feasible candidates, with a common baseline and verification distribution,*

$$\arg\max_{d' \in \mathfrak{F}_t} \mathbb{E}[r_{\text{dial}}] = \arg\min_{d' \in \mathfrak{F}_t} \left\{ D_{\text{KL},2}(p \,\|\, q_{d'}) + C(d') \right\}. \tag{27}$$

Validity condition: fidelity makes compression meaningful. Theorems 3 and 1 compare codes for the same canonical S; reconstruction and grounding scores are operational checks of this premise, not certificates of semantic equivalence. The expected-KL result additionally requires verification states not used to construct or select the candidate code; separate decoder calls on the construction state do not provide that independence. Moreover, Eq. 1 includes fidelity terms that may vary among feasible candidates, so Corollary 1 characterizes the proper-code objective rather than an unconditional equivalence for the full weighted reward.

The same principle gives a criterion for acquiring evidence. Let action a reveal task-visible evidence E_a, let Y denote a future grounded scientific state, let q_a^- and q_a^+ be predictive codes before and after observing E_a, and let $c(a) \geq 0$ be acquisition and update cost expressed in bit-equivalent units. Define

$$r_{\text{future}}(a; E_a, Y) = \log_2 \frac{q_a^+(Y \mid E_a, \mathcal{K})}{q_a^-(Y \mid \mathcal{K})} - c(a). \tag{28}$$

Theorem 4 (Predictive-information value of scientific actions)**.** *Assume $q_a^-(Y \mid \mathcal{K}) = p(Y \mid \mathcal{K}, a)$ and $q_a^+(Y \mid E_a, \mathcal{K}) = p(Y \mid E_a, \mathcal{K}, a)$. Then*

$$\mathbb{E}[r_{\text{future}}(a; E_a, Y) \mid \mathcal{K}, a] = I_2(Y; E_a \mid \mathcal{K}, a) - c(a), \tag{29}$$

where I_2 is conditional mutual information in bits.

Rule 2: useful evidence buys future predictability. Under the stated assumptions, an observation has positive expected value when its predictive information exceeds its bit-equivalent acquisition and update cost (Lindley, 1956; Houthooft et al., 2016). The induced ranking maximizes information *minus* cost, not information per unit cost. This criterion extends the analysis to evidence acquisition; the reported experiments do not estimate this mutual information or evaluate a separate information-gain acquisition module. Appendix D additionally shows how same-state scoring can reward memorization and gives a Fano-style bound relating reconstruction error to retained information. External experiments, hidden tests, or expert review remain necessary to establish scientific truth.

D Proofs for Grounded Dialect Compression as Intrinsic Reward

This appendix proves the results in Section 4 and states the assumptions under which the implemented compression score has the claimed predictive interpretation. All logarithms in this section are base two, so entropy, divergence, information, and description length are measured in bits. For readability, we condition on a fixed visible context $\mathcal{K} = k$ inside each proof; the conditional statements follow by averaging over $\mathcal{K}$.

D.1 Formal Setup and Correspondence to the Implemented Reward

Let S be the canonical grounded scientific state, where surface realizations that preserve the same propositions, relations, assumptions, and required anchors are identified as one semantic equivalence class. A dialect state d specifies a public dictionary, an encoder–decoder contract, and a normalized coding distribution $q_d(x \mid k)$ that is positive on every state under consideration. Its ideal data codelength and charged representation cost are

$$\ell_d(x \mid k) = -\log_2 q_d(x \mid k), \qquad C(d) \geq 0. \tag{30}$$

The cost $C(d)$ includes every definition, typed argument schema, decoder template, and query-local extension needed to interpret the code. For two dialect states d and d', write

$$\Delta C(d, d') = C(d') - C(d) \tag{31}$$

and define r_{dial} by Eq. 3. The code may be implemented by arithmetic coding, in which case the realized integer length differs from the ideal $-\log_2 q_d$ length only by the coder's bounded overhead; the theorems use the ideal length to keep the identities exact (Delétang et al., 2024).

Proposition 2 (Correspondence to the implemented compression term)**.** *Fix a proposition sequence $P = (p_i)_{i=1}^n$ and candidate $c = (Z, \widehat{P}, \Delta\mathcal{D})$, and let*

$$N(P) = \sum_{i=1}^{n} L(p_i) + \epsilon. \tag{32}$$

Assume $w_c > 0$ and identify a verbose baseline state d_0 and candidate state d_c through

$$\begin{aligned} \ell_{d_0}(P \mid \mathcal{K}) &= \sum_{i=1}^{n} L(p_i), \\ \ell_{d_c}(P \mid \mathcal{K}) &= \sum_{i=1}^{n} L(z_i), \\ \Delta C(d_0, d_c) &= \alpha L(\Delta\mathcal{D}) + \frac{N(P) w_b}{w_c} B_{\text{dict}}(c). \end{aligned} \tag{33}$$

Then

$$w_c G_{\text{comp}}(c) - w_b B_{\text{dict}}(c) = \frac{w_c}{N(P)} r_{\text{dial}}(d_0 \to d_c; P, \mathcal{K}). \tag{34}$$

Consequently, for a fixed proposition batch, the implemented compression-and-dictionary score and net dialect codelength reduction induce the same candidate ranking. The KL interpretation additionally requires L to be coding-compatible, meaning that it equals $-\log_2 q$ up to a candidate-independent additive constant, as well as the code, verification, and fixed-cost assumptions of Theorem 1. Without coding compatibility, Eq. 34 remains an algebraic description-length identity rather than a probabilistic one.

Proof. By Eq. 17,

$$\begin{aligned} w_c G_{\text{comp}}(c) - w_b B_{\text{dict}}(c) &= \frac{w_c}{N(P)} \left[\sum_{i=1}^{n} L(p_i) - \sum_{i=1}^{n} L(z_i) - \alpha L(\Delta\mathcal{D}) \right] - w_b B_{\text{dict}}(c) \\ &= \frac{w_c}{N(P)} \left[\ell_{d_0}(P \mid \mathcal{K}) - \ell_{d_c}(P \mid \mathcal{K}) - \Delta C(d_0, d_c) \right], \end{aligned}$$

where the second equality substitutes Eq. 33 and brings the dictionary penalty inside the common factor. The bracketed expression is exactly $r_{\text{dial}}(d_0 \to d_c; P, \mathcal{K})$. Since $w_c / N(P) > 0$ is constant across candidates for the same batch, both objectives have the same ordering. □

The identification above is conditional: the two data lengths must encode the same canonical record using normalized distributions, with no omitted candidate-dependent normalizer. Every candidate-specific definition must be included in the stated cost; allowing $\alpha = 0$ without charging the omitted definitions elsewhere does not meet that condition. The variable reconstruction, grounding, and agreement terms in Eq. 1 are not part of Eq. 34. They leave the proper-code ranking unchanged only when their weighted sum is constant across the feasible candidates, or when an additional argument establishes equivalent ordering. The effective scheme costs must also be fixed before verification. If $N(P)$ or a penalty varies with the verification record, that cost must remain inside the expectation; a per-record ranking identity alone does not imply the fixed-cost objective in Theorem 1. Token-length implementations therefore instantiate the auditable-abstraction heuristic, not automatically the exact predictive-code objective.

D.2 Prediction–Compression Equivalence

Proof of Theorem 3. By the autoregressive factorization,

$$\begin{aligned}\ell_d(S \mid \mathcal{K}) &= -\log_2 \prod_{j=1}^{m} q_d(S_j \mid S_{<j}, \mathcal{K}) \\ &= -\sum_{j=1}^{m} \log_2 q_d(S_j \mid S_{<j}, \mathcal{K}),\end{aligned}$$

and the same identity holds for d'. Subtracting the two lengths gives

$$\begin{aligned}\ell_d(S \mid \mathcal{K}) - \ell_{d'}(S \mid \mathcal{K}) &= \sum_{j=1}^{m} [\log_2 q_{d'}(S_j \mid S_{<j}, \mathcal{K}) - \log_2 q_d(S_j \mid S_{<j}, \mathcal{K})] \\ &= \sum_{j=1}^{m} \log_2 \frac{q_{d'}(S_j \mid S_{<j}, \mathcal{K})}{q_d(S_j \mid S_{<j}, \mathcal{K})}.\end{aligned}$$

Subtracting $\Delta C(d, d')$ yields Eq. 26. Therefore a one-bit reduction in ideal dialect codelength is exactly a one-bit improvement in the cumulative autoregressive log score, before representation cost. □

D.3 Epistemic Alignment on Independent or Prequential Verification States

For distributions p and q on the same support, define the base-two Kullback–Leibler divergence

$$D_{\mathrm{KL},2}(p \,\|\, q) = \mathbb{E}_{S \sim p}\left[\log_2 \frac{p(S)}{q(S)}\right]. \tag{35}$$

A verification state is *independent* when it was not used to propose, fit, select, or edit d'. A prequential implementation satisfies the same requirement by scoring each state using only a dialect constructed from earlier states.

Proof of Theorem 1. Condition on the visible context and on all information used to select d', so that d' is fixed while $S^{\mathrm{ver}} \sim p$ remains unseen. Using Eq. 3,

$$\mathbb{E}_p[r_{\mathrm{dial}}] = \mathbb{E}_p\left[\log_2 q_{d'}(S^{\mathrm{ver}} \mid \mathcal{K}) - \log_2 q_d(S^{\mathrm{ver}} \mid \mathcal{K})\right] - \Delta C(d, d').$$

From Eq. 35,

$$\begin{aligned}D_{\mathrm{KL},2}(p \,\|\, q_d) - D_{\mathrm{KL},2}(p \,\|\, q_{d'}) &= \mathbb{E}_p\left[\log_2 \frac{p(S^{\mathrm{ver}} \mid \mathcal{K})}{q_d(S^{\mathrm{ver}} \mid \mathcal{K})} - \log_2 \frac{p(S^{\mathrm{ver}} \mid \mathcal{K})}{q_{d'}(S^{\mathrm{ver}} \mid \mathcal{K})}\right] \\ &= \mathbb{E}_p\left[\log_2 q_{d'}(S^{\mathrm{ver}} \mid \mathcal{K}) - \log_2 q_d(S^{\mathrm{ver}} \mid \mathcal{K})\right].\end{aligned}$$

Substitution proves Eq. 4. The expected reward is positive exactly when the KL reduction is larger than $\Delta C(d, d')$. When $C(d') = C(d)$, the current-state divergence is constant across candidates, so maximizing expected reward is equivalent to minimizing $D_{\mathrm{KL},2}(p \,\|\, q_{d'})$. □

Proof of Corollary 1. For fixed d, Theorem 1 gives

$$\mathbb{E}[r_{\text{dial}}] = D_{\text{KL},2}(p \,\|\, q_d) + C(d) - [D_{\text{KL},2}(p \,\|\, q_{d'}) + C(d')] .$$

The first two terms do not depend on d'. Taking an argmax over $d' \in \mathfrak{F}_t$ is therefore equivalent to taking an argmin of the bracketed expression, which is Eq. 27. If $q_{d'} = q_d$ on the support of p and $C(d') \geq C(d)$, then the KL reduction is zero and the expected reward is nonpositive. More generally, if no candidate in the family reduces KL divergence by more than its additional cost, none has positive expected reward. □

Proposition 3 (Why same-state scoring is insufficient)**.** *Let $\mathcal{S}$ contain $M \geq 2$ grounded states and let the baseline code be uniform, $q_d(x) = 1/M$. After observing a construction state $S = x$, suppose an unrestricted candidate may choose*

$$q_{d'}(x) = 1 - \eta, \qquad q_{d'}(x') = \frac{\eta}{M-1} \quad \textit{for } x' \neq x, \tag{36}$$

where $0 < \eta < 1$ and representation cost is ignored. The reward measured on the same construction state is

$$r_{\text{dial}}(d \to d'; x) = \log_2\big(M(1-\eta)\big), \tag{37}$$

which is positive whenever $\eta < 1 - 1/M$. However, for any data-generating distribution p assigning positive probability to at least one $x' \neq x$, $D_{\text{KL},2}(p \,\|\, q_{d'}) \to \infty$ as $\eta \downarrow 0$. Thus same-state shortening can reward memorization while making out-of-sample prediction arbitrarily worse; independent, cross-fitted, or prequential verification is necessary for the epistemic interpretation in Theorem 1.

Proof. The same-state reward is

$$\log_2 \frac{q_{d'}(x)}{q_d(x)} = \log_2 \frac{1-\eta}{1/M} = \log_2\big(M(1-\eta)\big).$$

It is positive under the stated inequality. For any $x' \neq x$ with $p(x') > 0$, the corresponding term in the KL divergence is

$$p(x') \log_2 \frac{p(x')}{\eta/(M-1)},$$

which diverges to $+\infty$ as $\eta \downarrow 0$. Therefore arbitrarily confident memorization of the construction state can improve same-state code length while destroying verification performance. □

D.4 Scientific Actions and Predictive Information

For a fixed action a, conditional mutual information in bits is

$$I_2(Y; E_a \mid \mathcal{K}, a) = \mathbb{E}\left[\log_2 \frac{p(Y, E_a \mid \mathcal{K}, a)}{p(Y \mid \mathcal{K}, a)p(E_a \mid \mathcal{K}, a)}\right]. \tag{38}$$

Proof of Theorem 4. By calibration, $q_a^-(Y \mid \mathcal{K}) = p(Y \mid \mathcal{K}, a)$ and $q_a^+(Y \mid E_a, \mathcal{K}) = p(Y \mid E_a, \mathcal{K}, a)$. Bayes' rule gives

$$\frac{p(Y, E_a \mid \mathcal{K}, a)}{p(Y \mid \mathcal{K}, a)p(E_a \mid \mathcal{K}, a)} = \frac{q_a^+(Y \mid E_a, \mathcal{K})}{q_a^-(Y \mid \mathcal{K})}.$$

Substituting this ratio into Eq. 38 yields

$$I_2(Y; E_a \mid \mathcal{K}, a) = \mathbb{E}\left[\log_2 \frac{q_a^+(Y \mid E_a, \mathcal{K})}{q_a^-(Y \mid \mathcal{K})} \,\middle|\, \mathcal{K}, a\right].$$

The logarithm is exactly the reduction in ideal codelength for Y after observing E_a. Subtracting the deterministic charged cost $c(a)$ proves Eq. 29. Since $c(a)$ and the conditional mutual information are action-dependent, maximizing expected future compression reward chooses an action maximizing $I_2(Y; E_a \mid \mathcal{K}, a) - c(a)$. □

Corollary 2 (Zero value for noninformative scientific actions)**.** *Under the assumptions of Theorem 4, if* $Y \perp E_a \mid (\mathcal{K}, a)$*, then*

$$\mathbb{E}[r_{\text{future}} \mid \mathcal{K}, a] = -c(a) \leq 0. \tag{39}$$

Thus an action whose evidence cannot improve prediction of a future grounded scientific state receives no positive expected compression reward after cost.

Proof. Conditional independence implies $I_2(Y; E_a \mid \mathcal{K}, a) = 0$. The result follows immediately from Eq. 29. □

D.5 Faithfulness and the Exclusion of Semantic Collapse

Let $h_2(u) = -u \log_2 u - (1-u)\log_2(1-u)$ denote binary entropy, with the usual convention $0 \log 0 = 0$. The reconstruction error in this subsection is defined over canonical grounded states rather than exact surface strings: a reconstruction is correct when it preserves the proposition semantics and every required anchor under the fixed equivalence rule.

Theorem 5 (Faithful shortening cannot collapse scientific distinctions)**.** *Fix visible context* $\mathcal{K}$ *and suppose the canonical grounded state* S *takes values in* $M \geq 2$ *semantic equivalence classes. Let* R *be an accepted dialect representation and let* $\widehat{S} = g(R, \mathcal{K})$ *be its independent reconstruction. If* $\Pr[\widehat{S} \neq S \mid \mathcal{K}] \leq \varepsilon$ *with* $0 \leq \varepsilon \leq 1 - 1/M$*, then*

$$I_2(S; R \mid \mathcal{K}) \geq H_2(S \mid \mathcal{K}) - h_2(\varepsilon) - \varepsilon \log_2(M-1). \tag{40}$$

Any prefix code for R *has expected length at least the right-hand side of Eq. 40. For exact reconstruction (*$\varepsilon = 0$*), a deterministic encoder is injective on the support of* S *and its expected length is at least* $H_2(S \mid \mathcal{K})$*. Thus accepted shortening can remove coding redundancy but cannot merge scientifically distinguishable grounded states without increasing the allowed reconstruction error.*

Proof of Theorem 5. Let $E = \mathbf{1}\{\widehat{S} \neq S\}$ be the reconstruction-error indicator. The conditional form of Fano's inequality (Scarlett & Cevher, 2019), together with monotonicity of $h_2(u) + u \log_2(M-1)$ on $[0, 1 - 1/M]$, gives

$$H_2(S \mid R, \mathcal{K}) \leq h_2(\varepsilon) + \varepsilon \log_2(M-1). \tag{41}$$

Therefore

$$\begin{aligned} I_2(S; R \mid \mathcal{K}) &= H_2(S \mid \mathcal{K}) - H_2(S \mid R, \mathcal{K}) \\ &\geq H_2(S \mid \mathcal{K}) - h_2(\varepsilon) - \varepsilon \log_2(M-1), \end{aligned}$$

which proves Eq. 40. For any prefix code of R conditioned on $\mathcal{K}$, the source-coding lower bound gives

$$\mathbb{E}[\ell(R \mid \mathcal{K})] \geq H_2(R \mid \mathcal{K}) \geq I_2(S; R \mid \mathcal{K}),$$

so its expected length is at least the right-hand side of Eq. 40.

When $\varepsilon = 0$, Eq. 41 gives $H_2(S \mid R, \mathcal{K}) = 0$. For a deterministic encoder, if two distinct grounded states in the support mapped to the same representation, the decoder would receive identical $(R, \mathcal{K})$ for both and could not reconstruct both exactly, a contradiction. Hence the encoder is injective on the support. Moreover, $I_2(S; R \mid \mathcal{K}) = H_2(S \mid \mathcal{K})$, so every prefix code for R has expected length at least $H_2(S \mid \mathcal{K})$. Accepted shortening can therefore reduce redundancy above the entropy floor, but exact shortening cannot merge scientifically distinct states. □

Remark 1 (What the gates must estimate)**.** *Theorem 5 is conditional on a bound for semantic reconstruction error. In the implementation,* S_{rec}*,* S_{ground}*,* S_{agree}*, and* V_v *are observable proxies for this condition. The theorem does not assert that an arbitrary judge score automatically equals the true error probability; rather, it specifies the property that the validation stack must control for the compression reward to have the stated information-preservation interpretation.*

D.6 Amortized Primitive Cost

Corollary 3 (Amortized promotion of a scientific primitive)**.** *Suppose a proposed primitive s incurs a one-time charged cost $C_s > 0$ and is evaluated on independent or prequential uses $i \in I_s$. Let*

$$g_i = \ell_{d_i}(S_i \mid \mathcal{K}_i) - \ell_{d_i+s}(S_i \mid \mathcal{K}_i) \tag{42}$$

be its verified codelength saving on use i before charging C_s. Its cumulative net intrinsic reward is positive if and only if

$$\sum_{i \in I_s} g_i > C_s. \tag{43}$$

Under the assumptions of Theorem 1, the expected version is

$$\sum_{i \in I_s} [D_{\mathrm{KL},2}(p_i \,\|\, q_{d_i}) - D_{\mathrm{KL},2}(p_i \,\|\, q_{d_i+s})] > C_s. \tag{44}$$

Thus a private one-off alias is not intrinsically valuable merely because its surface form is short; a primitive must repay its public definition and verification cost through repeated predictive savings.

Proof. The cumulative reward is

$$\sum_{i \in I_s} g_i - C_s,$$

whose sign is positive exactly under Eq. 43. Applying Theorem 1 independently to each use and summing the expected codelength reductions yields Eq. 44. □

D.7 Scientific Decision Utility Under a Predictive Code

Fix side information $\mathcal{K} = k$. Let $\mathcal{U}$ be a finite set of scientific decisions, including a possible joint action of a multi-agent system, and let $u : \mathcal{U} \times \mathcal{S} \to [0, 1]$ be a bounded utility. The utility is used only to state the decision-theoretic guarantee; it need not be observed by the intrinsic controller. For distributions p, q on scientific records, define

$$\begin{gathered} a_p \in \arg\max_{a \in \mathcal{U}} \mathbb{E}_p[u(a, S)], \qquad a_q \in \arg\max_{a \in \mathcal{U}} \mathbb{E}_q[u(a, S)], \\ \mathrm{Reg}_p(q) = \mathbb{E}_p[u(a_p, S) - u(a_q, S)]. \end{gathered} \tag{45}$$

Ties are broken by a fixed rule. The regret concerns the quality of a decision made using the predictive distribution, not the compression score itself.

Corollary 4 (A scientific decision-regret bound)**.** *Under the above definitions,*

$$0 \le \mathrm{Reg}_p(q) \le 2\,\mathrm{TV}(p, q) \le \sqrt{2 \ln(2)\, D_{\mathrm{KL},2}(p\|q)}, \tag{46}$$

where $\mathrm{TV}(p, q) = \frac{1}{2} \sum_s |p(s) - q(s)|$ is total variation distance. Consequently, among schemes with equal representation cost, maximizing expected verified compression reward minimizes this KL-based upper bound. For unequal costs, a candidate with $\mathbb{E}[r_{\mathrm{dial}}] + \Delta C > 0$ has a strictly smaller KL term than the baseline. This condition is implied by positive expected reward when $\Delta C \ge 0$.

Proof. Add and subtract the expectations under q:

$$\begin{aligned} \mathrm{Reg}_p(q) &= (\mathbb{E}_p - \mathbb{E}_q)u(a_p, S) + \left[\mathbb{E}_q u(a_p, S) - \mathbb{E}_q u(a_q, S)\right] + (\mathbb{E}_q - \mathbb{E}_p)u(a_q, S) \\ &\le 2\,\mathrm{TV}(p, q). \end{aligned}$$

The bracketed term is nonpositive by optimality of a_q, and each remaining term is at most total variation because $0 \le u \le 1$. For completeness, the KL bound follows by taking $A = \{s : p(s) \ge q(s)\}$, $r = p(A)$, and $b = q(A)$. The log-sum inequality gives $D_{\mathrm{KL},e}(p\|q) \ge r \ln(r/b) + (1 - r) \ln((1 - r)/(1 - b))$. As a function of r, the right-hand side is zero with zero derivative at $r = b$, and its second derivative is $1/[r(1 - r)] \ge 4$. It is therefore at least $2(r - b)^2 = 2\,\mathrm{TV}(p, q)^2$, with boundary cases obtained by continuity. Finally, $D_{\mathrm{KL},e} = \ln(2) D_{\mathrm{KL},2}$ proves Eq. 46. The comparison between representations follows from Theorem 1. □

Note that a smaller bound need not imply smaller realized decision regret for every utility. For example, let $p = (0.51, 0.49)$ on two outcomes, $q_0 = (0.9, 0.1)$, and $q_1 = (0.49, 0.51)$, and let a decision receive utility one exactly when it predicts the outcome. The KL divergence decreases from approximately 0.706 bits to 0.00115 bits, yet q_0 chooses the Bayes-optimal first outcome while q_1 chooses the second and incurs regret 0.02. Likewise, a fixed black-box solver need not implement the utility-maximizing rule under its implicit predictive code. The theorem establishes log-score alignment and a decision-regret certificate under the stated assumptions, not unconditional improvement in every downstream metric.

D.8 Representation Sufficiency and Semantic Minimality

The results here concern finite proposition-and-anchor records, which can include values, units, directed relations, evidential status, and source pointers. They apply to symbolic encodings with a public dictionary, whether the solver is one model or a communicating group of agents. No new empirical claim is needed for these representation statements.

Proof of Theorem 2. First construct an exact encoding. Serialize every source proposition and its typed anchors using public field names, unambiguous delimiters, and explicit escaping of literal text. Preserve relation endpoints and roles, rather than serializing only an unordered collection of values. This finite residual-text representation is always expressible under the theorem's assumption. A decoder that reads the fields and unescapes the literals reconstructs S exactly. Replacing any subrecord by a primitive instance is permissible when the primitive's public expansion reconstructs exactly that subrecord. Thus the existence of the residual channel makes the representation class sufficient even if the reusable dictionary contains no applicable primitive. It does not imply that this fallback representation is short.

Now fix $\mathcal{K} = k$ and let $R = E(S, k)$ be deterministic with exact recovery $S = g(R, k)$. Then S and R determine one another on their conditional support. They generate the same information, so conditioning on either gives the same distribution of every downstream variable Y. This proves Eq. 5. For a randomized encoder, the same conclusion holds when its randomization is independent of Y conditional on (S, k) and recovery remains exact.

Let π be any decision rule based on (S, k). Define $\pi_R(R, k) = \pi(g(R, k), k)$, coupling any policy randomization identically. The decisions agree whenever the reconstruction is correct. For a utility in $[0, 1]$, the pointwise utility difference is at most one on the reconstruction-error event and is zero elsewhere. Hence

$$\mathbb{E}\big[u(\pi(S, k), Y) - u(\pi_R(R, k), Y) \mid k\big] \le \Pr[g(R, k) \neq S \mid k] \le \varepsilon. \tag{47}$$

With exact reconstruction the two rules have identical expected utility. The approximate statement is an emulation guarantee, not a bound on an arbitrary solver that ignores or misinterprets the decoded state. □

From the declared record to the full task. Let W denote the full available source material and suppose $S = \kappa(W, k)$ is its declared semantic record. Equation 5 preserves what is in S. To preserve all information in W relevant to a downstream outcome Y, one additionally needs $Y \perp W \mid (S, k)$. Under this sufficiency assumption, $p(Y \mid W, k) = p(Y \mid S, k) = p(Y \mid R, k)$. No choice of decoder can recover an omitted scientific commitment that was absent from both S and the side information. This is why anchor completeness and faithful abstraction are separate requirements.

Proof of Proposition 1. Fix k and define an equivalence relation on source descriptions by

$$w \sim_k w' \quad \Longleftrightarrow \quad \kappa(w, k) = \kappa(w', k). \tag{48}$$

The canonical record is the corresponding equivalence-class label. Suppose two different classes had the same representation r. The decoder would receive the same pair (r, k) for distinct required records and could not reconstruct both, contradicting exact recovery. Thus every admissible encoding must distinguish these classes. Conversely, encoding the class label itself distinguishes no additional surface forms and is sufficient for all declared commitments. It is minimal in this sense, up to an invertible relabeling of the classes.

For any conditional prefix code with length $L(r \mid k)$, the source-coding inequality gives

$$\mathbb{E}[L(R) \mid k] \geq H_2(R \mid k) \geq I_2(S; R \mid k) = H_2(S \mid k), \tag{49}$$

where exact recovery gives the final equality. One can verify the first inequality directly from Kraft's inequality: with $Z_k = \sum_r 2^{-L(r|k)} \leq 1$ and $q(r \mid k) = 2^{-L(r|k)}/Z_k$, nonnegativity of KL gives $\mathbb{E}[L(R) \mid k] = H_2(R \mid k) + D_{\mathrm{KL},2}(p_R \| q) - \log_2 Z_k \geq H_2(R \mid k)$. If the conditional state distribution is known, lengths $\lceil -\log_2 p(s \mid k) \rceil$ admit a prefix code and achieve expected length below $H_2(S \mid k) + 1$ bits; coding independent blocks approaches the entropy rate. This establishes an information-theoretic floor, not token-optimality of the learned dictionary or of a particular LLM tokenizer. □

Two limits of the minimality statement. First, a task that asks only for one decision may admit a coarser statistic than the full scientific record. Our minimality claim is for exact recovery of every declared commitment, not for every possible task-specific utility. Second, all bounds are conditional on the permitted side information. If that side information already determines the entire record, the conditional entropy is zero and the lower bound is vacuous. A decoder's success with rich evidence therefore does not by itself quantify information carried uniquely by the symbolic expression. This distinction is separate from whether the expression is grounded in that evidence.

Corollary 5 (Lossless abstraction under a representation budget)**.** *Consider a finite set of source records with two exact public encodings, E_0 and E_1. Suppose their fully charged representation lengths satisfy $L(E_1(s,k)) \leq L(E_0(s,k))$ for every relevant s. Under a fixed representation budget $B \in \mathbb{N}$ bits, every source-state policy whose memory is feasible using E_0 remains representationally feasible using E_1 by composing the source-state policy with E_1's decoder. Thus the best attainable expected utility cannot decrease, provided the policy class permits that composition and decoding computation is feasible. It increases strictly when the compressed encoding makes a previously infeasible, higher-utility source-state policy feasible.*

Proof. Any memory record that fits under E_0 has length at most B; the assumed pointwise inequality implies that its E_1 encoding also fits. Exact decoding reconstructs the same record, allowing the same downstream policy and utility. Hence the old feasible policy set is contained, up to equivalent recoding, in the new one. The nondecrease follows by taking a supremum of expected utility over these sets. Strict increase follows when the new feasible set contains a policy with utility above the old supremum. □

This corollary provides a resource-based reason that lossless abstraction can help a memory-limited MAS. It does not describe a measured token-budget guarantee for the reported implementation, which retains original context, can add decoded meanings, and incurs audit cost. Nor does maximizing token savings alone choose the most useful policy when different candidates preserve different scientific content.

D.9 Scope of the Guarantees

The two parts of the analysis make complementary statements. Under proper public codes, complete cost accounting, and independent or prequential verification, expected net compression progress measures predictive log-score improvement after cost. Under exact public decoding, the proposed state format can preserve every declared commitment and emulate any source-state decision rule. Qualified minimality concerns the semantic distinctions required by exact recovery, while the budget result concerns representational feasibility, not the behavior of a fixed LLM. The task-level conclusions require that the source record itself retains the information relevant to the downstream task. Approximate audits, tokenizer lengths, and one-shot candidate selection need not satisfy these ideal conditions. The evidence-acquisition identity additionally assumes calibrated before-and-after codes and is not an evaluated module. Neither part establishes scientific truth, causal validity, or universal improvement of a fixed solver.

E METRIC DEFINITIONS

E.1 TOKEN F1

Let y be a generated answer, let $y^\star$ be its target answer, and let $T(\cdot)$ return a normalized token multiset. Let $\epsilon > 0$ be a numerical stabilizer. Token precision, recall, and F1 are

$$P_{\text{tok}} = \frac{|T(y) \cap T(y^\star)|}{|T(y)| + \epsilon}, \qquad R_{\text{tok}} = \frac{|T(y) \cap T(y^\star)|}{|T(y^\star)| + \epsilon}, \qquad F1_{\text{tok}} = \frac{2P_{\text{tok}}R_{\text{tok}}}{P_{\text{tok}} + R_{\text{tok}} + \epsilon}. \tag{50}$$

E.2 ROUGE-L F1

ROUGE-L uses the longest common subsequence between candidate and reference text (Lin, 2004). Let $\text{LCS}(y, y^\star)$ be the longest-common-subsequence length, and let $|y|$ and $|y^\star|$ denote the token lengths used by the evaluator.

$$P_{\text{LCS}} = \frac{\text{LCS}(y, y^\star)}{|y| + \epsilon}, \qquad R_{\text{LCS}} = \frac{\text{LCS}(y, y^\star)}{|y^\star| + \epsilon}, \qquad F1_{\text{ROUGE-L}} = \frac{2P_{\text{LCS}}R_{\text{LCS}}}{P_{\text{LCS}} + R_{\text{LCS}} + \epsilon}. \tag{51}$$

E.3 FINAL-ANSWER AND TRACE ANCHOR COVERAGE

Let $\mathcal{A}_\tau = \bigcup_{p_i \in P_\tau} g_{\text{anc}}(p_i, \mathcal{K}_\tau; \Omega)$ be the nonempty task-level union of required anchors, and let o be the output field being audited. Let $\text{Recover}_{\text{out}}(o, \mathcal{K}_\tau)$ denote the anchors recovered from that output under the fixed evaluation rules. Anchor coverage is

$$C_{\text{anchor}}(o, \tau) = \frac{1}{|\mathcal{A}_\tau|} \sum_{a \in \mathcal{A}_\tau} \mathbf{1}\{a \in \text{Recover}_{\text{out}}(o, \mathcal{K}_\tau)\}. \tag{52}$$

Final-answer anchor coverage sets o to the final answer. Trace-anchor coverage sets o to the concatenation of the answer, dialect expression, decoded state, evidence audit, and skeptic report. Structured anchors are matched by normalized exact matching when available. Natural-language anchors use the fixed extractor and matching rules from the experiment scripts.

E.4 ADAPTED DISCOVERYBENCH OFFICIAL-CODE SCORE

The DiscoveryBench primary result uses the adapted official-code field `official_final_score`. The score is labeled adapted official-code because the evaluator wrapper follows the benchmark's row structure and scoring logic outside the hosted leaderboard environment. All methods in Table 2 are evaluated on the same 191 frozen held-out task IDs with the same fixed evaluator backbone.

E.5 FACETED DISCOVERYBENCH DIAGNOSTIC

For mechanism analysis, a separate faceted diagnostic scores context, variables, relation, evidence use, and workflow plausibility on a $[0, 1]$ scale. Let $q_{\tau,k} \in [0, 1]$ be the score for facet $k \in \{1, \ldots, 5\}$ and let $r_\tau \in [0, 1]$ be a validity factor penalizing empty, non-responsive, or malformed answers. The per-task diagnostic score is

$$J_\tau = r_\tau \cdot \frac{1}{5} \sum_{k=1}^{5} q_{\tau,k}, \tag{53}$$

and the aggregate judge total is the mean of J_τ over the manifest. Facet means are aggregated separately, so the arithmetic mean of aggregate facets need not exactly equal the aggregate judge total. The diagnostic is method-blind and frozen across rows, but it is not the primary adapted-official score.

E.6 PRESCIENCE CONTRIBUTION-GENERATION METRICS

The PreScience contribution-generation block reports token F1, ROUGE-L F1, grounding-anchor coverage, novelty, diversity, and cost. Token F1 and ROUGE-L measure recovery of reference contribution content, while grounding measures recovery of task-visible scholarly anchors. The main text emphasizes token F1 and grounding because they directly separate contribution-content recovery from evidence fidelity.

E.7 SCICODE OBSERVABLE AND AVAILABLE-TEST METRICS

The SciCode evaluator reports an observable score, an available-test pass rate, observable execution rate, and parse rate. The SciCode block uses observable and available-test metrics because hidden-test targets are outside this evaluator contract. The available-test pass rate is the fraction of generated solutions that pass the checks exposed to the harness. The parse rate measures whether generated solutions can be parsed into executable Python artifacts.

E.8 PIEVO COMPATIBLE-CORE METRICS

The molecular bioactivity (MBO), nanomaterial optical-property (NHO), superconductor critical-temperature (SPO), and transition-metal-complex (TMC) rows use a dependency-light compatible-core reproduction of the principle-evolution interface. For each task, solution quality (SQ) is the normalized final best score and AUOC is the area under the cumulative-best curve over the fixed objective budget. Both are reported as percentages with one decimal place. The compatible-core runner uses the same deterministic surrogate, 20 paired seeds, 24 objective evaluations, and eight candidate proposals per selection step across methods. It does not assign LLM backbones or API costs to these rows. These values are internally comparable within the compatible-core harness and are not combined numerically with public results produced by a different runtime, agent stack, or score normalization.

E.9 EQUATION-DISCOVERY DIAGNOSTICS

For equation-discovery settings, symbolic accuracy should measure exact or equivalence-checked expression-family recovery after variable normalization. For m held-out observations, let y_i be the target value, $\hat{y}_i$ the predicted value, $\bar{y} = m^{-1} \sum_{i=1}^{m} y_i$ the target mean, and $\epsilon > 0$ a numerical stabilizer. The coefficient of determination is

$$R^2 = 1 - \frac{\sum_{i=1}^{m} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{m} (y_i - \bar{y})^2 + \epsilon}. \tag{54}$$

Equation-discovery evaluation is outside the reported benchmark suite, so the manuscript reports no equation-discovery score.

F Comparator and Configuration Details

LLM-facing comparators. The study spans the Qwen, DeepSeek, and GLM model families (Yang et al., 2025; Liu et al., 2024; Glm et al., 2024). Direct generation is a single-pass task prompt. Chain-of-thought (CoT) requests an explicit reasoning trace (Wei et al., 2022), and self-consistency aggregates multiple sampled traces (Wang et al., 2023). Prompt Compression follows context-shortening methods such as LLMLingua (Jiang et al., 2023) but adds no independent reconstruction or grounding gate. ConceptBank is a study-specific reusable-concept-memory baseline motivated by learned-library approaches (Ellis et al., 2020; Grand et al., 2024). Retrieval Only augments the prompt with scholarly evidence in the style of retrieval-augmented generation (Lewis et al., 2020). Reviewer Rerank is a study-specific candidate-selection baseline that uses an LLM reviewer, following the broader LLM-as-a-judge paradigm (Zheng et al., 2023). Multi-Agent Forecasting coordinates proposer and critic roles, as in multi-agent deliberation (Du et al., 2024). Self-Refine and Reflexion perform iterative critique and revision (Madaan et al., 2023; Shinn et al., 2023), and Frontier Direct is a stronger instruction-only task prompt.

SciCode comparators. ReAct interleaves reasoning with tool-oriented actions (Yao et al., 2023b); Self-Debug revises code from execution feedback (Chen et al., 2024); Code Rerank selects among program candidates using generated or observable tests (Chen et al., 2023); and execution-guided, unit-test, and Reflexion repair use the same visible verifier feedback under their reported call budgets. The full SciCode table in Appendix I also reports Frontier Direct and lighter SciDialect coding-adapter ablations so that execution quality can be compared with parse validity, token use, and repair cost.

Optimization comparators. The compatible-core study includes random search (RS) (Bergstra & Bengio, 2012), a genetic algorithm (GA) (Holland, 1992), differential evolution (DE) (Storn & Price, 1997), a vanilla multi-agent-system (MAS) proposer–critic policy without principle evolution, and PiEvo (Pu et al., 2026). Every method uses the same deterministic surrogate, objective-evaluation budget, proposal count, and paired seeds. The ablations in Appendix K change only how dialect features, validity gates, and anomaly routing enter the compatible-core search process.

Primary configurations. Each main table reports one pre-specified SciDialect configuration per benchmark so that the primary comparison tests the shared intrinsic-reward mechanism rather than selecting among many adapter variants after observing held-out results. Development-set adapter selection for DiscoveryBench is documented in Appendix G; PreScience cost matching is documented in Appendix H; and SciCode and optimization component variants are reported in Appendices I and K.

G DiscoveryBench Split and Selection Protocol

The DiscoveryBench generator study contains 239 public tasks. Forty-eight tasks are used exclusively for choosing among dialect renderers and fixing the Prompt Compression comparator. The remaining 191 task IDs form the frozen held-out split. All nine methods are evaluated on those IDs with three generation seeds, producing 5,157 evaluator rows and no evaluator errors. The evaluator backbone and evaluator prompt are fixed across generator methods. The selected compact SciDialect renderer is a one-call configuration that retains only the proposition, evidence, and relation fields needed by the adapted official-code scorer; alternative renderers are retained as separate ablation rows rather than combined after held-out evaluation.

The held-out result is interpreted through two comparator roles. Prompt Compression is the pre-specified confirmatory comparator because it directly tests whether grounding and reconstruction add value beyond shortening the context. Direct is the strongest baseline observed after held-out scoring and is reported as a descriptive robustness comparison. The two roles are kept separate in Section 6.2.

Adapter	Final score	Recall	Tokens	CF cost
SciDialect (compact)	**15.8**	36.3	**2171**	**0.0034**
SciDialect (hypothesis-rich)	14.8	34.0	2845	0.0045
SciDialect (minimal)	14.2	**37.0**	2817	0.0044

Table G.6: DiscoveryBench adapter ablation on the frozen held-out split. The compact renderer retains only proposition, evidence, and relation fields; richer variants expose additional symbolic structure to the final generator.

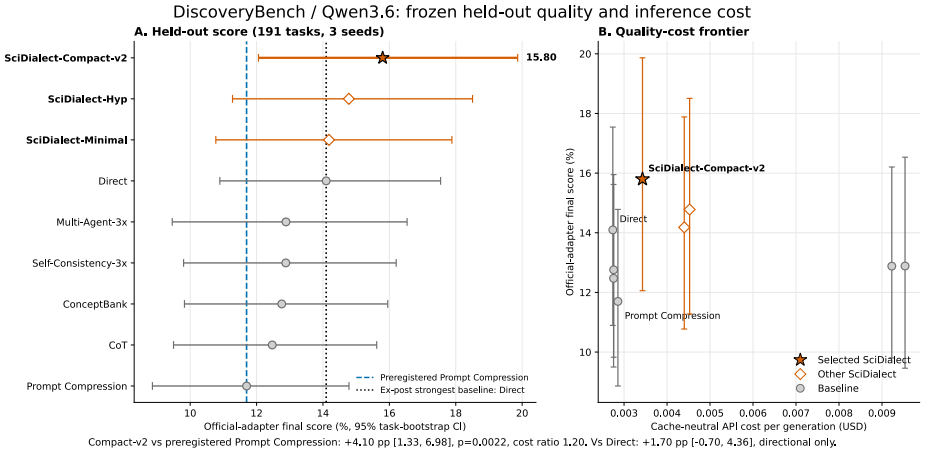


Figure G.1: DiscoveryBench score intervals and quality–cost frontier. The compact dialect is compared with Direct and Prompt Compression under the frozen evaluator.

H PRESCIENCE COST AND EFFICIENCY ANALYSIS

Table H.7 reports the counterfactual cache-free (CF) API cost that is omitted from the main PreScience table. The values use the fixed pricing configuration described in Section 5.1 and are reported per generation.

Method	Qwen3.5	DeepSeek-V3	GLM-4
Direct	0.0014	0.0027	0.0027
CoT	0.0014	0.0028	0.0028
Self-consistency	0.0022	0.0031	0.0034
Prompt compression	0.0015	0.0029	0.0029
Multi-agent forecasting	0.0022	0.0031	0.0033
Retrieval only	0.0022	0.0032	0.0033
Reviewer rerank	0.0021	0.0030	0.0033
Self-refine	0.0042	0.0059	0.0063
Reflexion	0.0042	0.0059	0.0061
Frontier direct	0.0021	0.0030	0.0033
SCIDIALECT	0.0015	0.0028	0.0029

Table H.7: PreScience CF cost per generation. Dashes indicate that the four-call reranker was used only for the paired GLM analysis.

The four-call GLM reviewer-rerank comparator obtains token F1 15.7, grounding 21.7, 3,977 tokens, and CF cost 0.0131. Against this comparator, SCIDIALECT improves token F1 by 3.9 percentage points and grounding by 16.8 percentage points while using approximately 22% of the CF cost. The corresponding ratios against Qwen CoT and DeepSeek Retrieval Only are approximately 1.07 and 0.88.

Figure H.2 places all matched methods in a common quality–cost plane. The three SCIDIALECT points occupy the upper region of their respective backbone clouds, while the four-call GLM reranker illustrates the additional cost of candidate generation followed by reviewer selection.

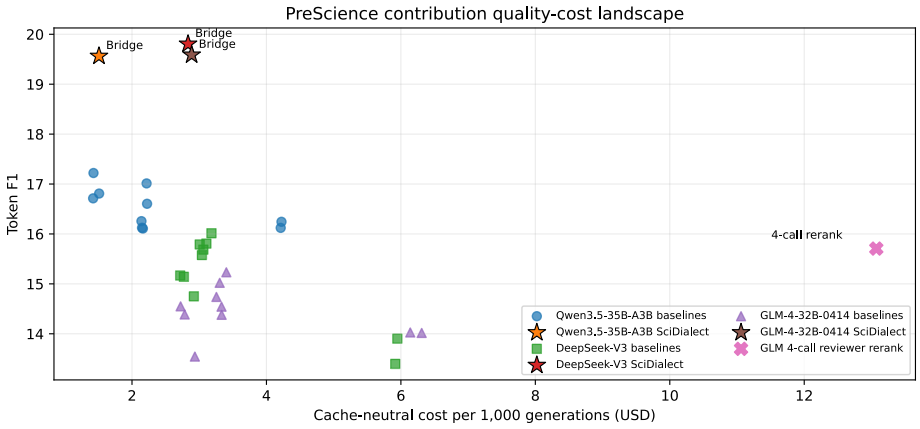


Figure H.2: PreScience token F1 versus CF cost across the matched baselines and three LLM families. Cost is shown per 1,000 generations so that single-call and multi-call methods appear on a readable scale.

I SciCode Component Ablations and Cost Frontier

Table I.8 expands the primary comparison in Table 3 to the full adapter and repair set. The table separates execution quality from parse validity and reports cost alongside quality because repair depth changes test-time compute substantially.

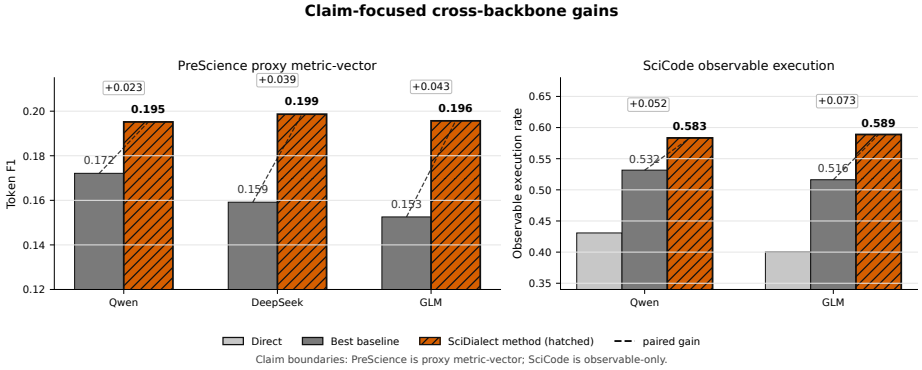


Figure I.3: Claim-focused gains under matched evaluator contracts. The PreScience panel reports token F1 across three LLM families; the SciCode panel reports observable execution on Qwen and GLM. Hatched bars denote SciDialect, gray bars denote Direct and the strongest paired comparator, and labels report absolute gains over the paired comparator. Metrics are benchmark-specific and are not compared across panels.

Backbone	Method	Execution	Parse	Tokens	CF cost
Qwen3.5-35B-A3B	Direct	40.0	78.0	4799	0.0073
	Frontier direct	46.0	92.0	27719	0.0376
	ReAct/tool coding	47.5	94.0	28649	0.0400
	Self-debug	49.5	98.0	27765	0.0396
	Code rerank	45.0	92.0	28625	0.0399
	Execution-guided repair	44.8	88.0	28642	0.0401
	Unit-test repair	45.8	92.0	28024	0.0381
	Reflexion repair	41.3	88.0	29177	0.0412
	SciDialect (base)	41.8	88.0	23772	0.0339
	SciDialect (visible repair)	53.5	90.0	25868	0.0362
	SciDialect (NumPy robust)	46.1	88.0	26124	0.0366
	SciDialect (contract repair)	**55.2**	98.0	31695	0.0438
GLM-4-32B-0414	Direct	41.1	88.0	4021	0.0110
	Frontier direct	46.0	86.0	23861	0.0605
	ReAct/tool coding	44.8	82.0	24147	0.0619
	Self-debug	47.3	90.0	22802	0.0597
	Code rerank	52.8	86.0	24609	0.0640
	Execution-guided repair	39.5	84.0	24916	0.0654
	Unit-test repair	41.3	84.0	24133	0.0620
	Reflexion repair	42.3	82.0	24466	0.0632
	SciDialect (base)	47.2	90.0	19957	0.0524
	SciDialect (visible repair)	49.0	90.0	21479	0.0546
	SciDialect (NumPy robust)	43.5	82.0	21586	0.0549
	SciDialect (contract repair)	**56.0**	90.0	34964	0.0689

Table I.8: SciCode observable-test results on 50 held-out problems per backbone. Execution and parse are percentages; tokens and CF cost are per problem.

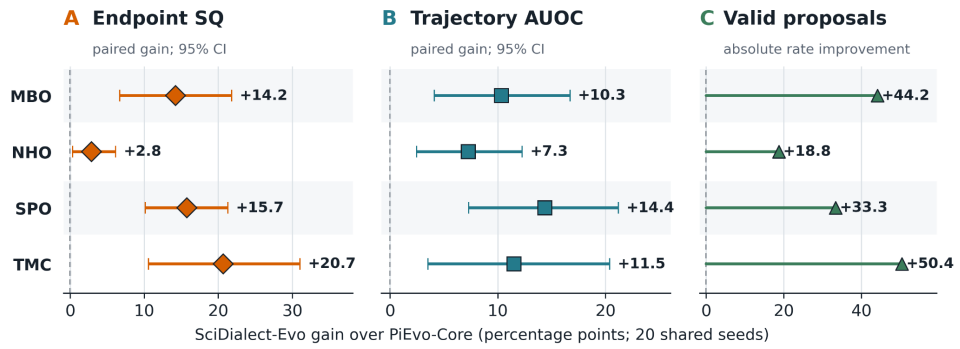


Figure K.4: Paired equal-budget gains of SCIDIALECT over PIEVO across 20 shared seeds. Markers report endpoint SQ and trajectory AUOC differences; annotations give the corresponding validity improvement.

On Qwen, the SCIDIALECT contract-repair adapter has the highest mean execution rate, 55.2 versus 49.5 for Self-Debug, with a paired difference of 5.7 percentage points and a 95% interval of $[0.3, 13.0]$. The lighter SCIDIALECT visible-repair adapter reaches 53.5 while using 25,868 tokens and cost 0.0362 per problem, compared with 27,765 tokens and cost 0.0396 for Self-Debug. On GLM, the contract-repair adapter reaches 56.0 versus 52.8 for Code Rerank; the paired difference is 3.2 percentage points with interval $[1.5, 10.5]$ and a 1.5 cost ratio. These rows show that contract-anchored repair can improve the mean observable execution rate.

J DISCOVERYBENCH TASK EXAMPLES

Table J.9 lists real final-manifest task examples from the DiscoveryBench evaluation. The examples are included to make the benchmark concrete and to clarify what anchors mean in practice.

Task identifier	Domain	Task input summary	Target output summary	Anchor types
`discoverybench_real_test_archaeology_34_0`	Humanities	Time-series analysis over archaeology and pollen-openness variables, with dates measured before the Common Era (BCE).	Identify a steep decline between 1600–1700 BCE and a later peak around 1400 BCE.	Time window, BCE dates, growth-rate variable, pollen-openness variable, peak direction, workflow.
`discoverybench_real_test_nls_ses_5_1`	Sociology	Regression and interaction analysis over National Longitudinal Survey (NLS) socioeconomic-status and academic-ability data.	State that the advantage of higher socioeconomic status (SES) is less pronounced at higher academic ability because the SES–ability interaction is negative.	SES variable, ability variable, academic outcome, interaction sign, regression relation.
`discoverybench_real_test_meta_regression_raw_1_0`	Metascience	Meta-regression over original and replication-study power variables.	Compare observed power in original studies with planned power in replication studies for Experimental Economics.	Domain filter, original-study power, replication planned power, aggregation, numeric means.
`discoverybench_real_test_worldbank_education_gdp_1_0`	Economics	Econometric analysis of education expenditure and gross domestic product (GDP) per capita in developing-country groups.	State a positive relationship between education expenditure and per-capita GDP.	Education-expenditure variable, GDP variable, country group, direction, econometric workflow.

Table J.9: Representative DiscoveryBench final-manifest examples.

K COMPATIBLE-CORE ABLATIONS, POLICY CONTEXT, AND AUDITABILITY

Table K.10 further reports the component variants omitted from the main comparison. These values come from a separate small-scale development evaluation, not the primary twenty-seed comparison in Table 4. They are compared only within this table and are not substituted for the primary means or pooled with them. The variants change feature, gate, and anomaly configurations, with associated policy differences; they are exploratory integration comparisons rather than fixed-candidate estimates of individual component effects.

	MBO		NHO		SPO		TMC	
Variant	SQ	AUOC	SQ	AUOC	SQ	AUOC	SQ	AUOC
SciDialect (full integration)	**81.8**	**61.4**	**99.6**	**90.0**	**94.9**	74.9	64.8	**47.8**
Features only	76.6	58.4	99.2	85.5	93.5	**75.5**	58.0	44.9
Gate only	70.4	53.8	98.7	88.5	94.7	73.4	**65.7**	46.6
Anomaly only	76.6	59.5	97.9	85.9	90.4	71.7	63.6	46.2

Table K.10: Compatible-core SciDialect component ablation. No single component dominates across tasks; the full integration provides the strongest aggregate balance of endpoint and trajectory quality.

Table K.11 jointly reports paired quality gains, proposal validity, and compatible-core runtime. The quality advantage is accompanied by complete removal of invalid or constraint-violating proposals at nearly unchanged wall time; every row uses 24 objective evaluations and 192 candidate proposals.

	Paired gain [95% CI]		Valid (%)		Violations (%)		Wall time (s)	
Task	ΔSQ	ΔAUOC	PiEvo	SciDialect	PiEvo	SciDialect	Core	SciDialect
MBO	+8.3 [2.7, 15.8]	+10.3 [4.1, 16.7]	55.8	100.0	44.2	0.0	0.357	0.373
NHO	+2.8 [0.3, 6.1]	+7.3 [2.5, 12.3]	81.2	100.0	18.8	0.0	0.376	0.390
SPO	+15.7 [10.1, 21.3]	+14.4 [7.3, 21.2]	66.7	100.0	33.3	0.0	0.353	0.394
TMC	+20.7 [10.5, 31.0]	+11.5 [3.5, 20.4]	49.6	100.0	50.4	0.0	0.332	0.375

Table K.11: Paired SciDialect minus PiEvo quality differences, proposal validity, constraint-violation rates, and wall time in small-scale development set. Quality differences are percentage points; validity and violation rates use a 0–100 scale.

Table K.12 reports behaviorally matched compatible-core policy abstractions for baselines for scientific-agents. They operate on the same deterministic surrogate and objective budget as Table 4 and provide controlled search-policy context within the compatible-core environment.

	MBO		NHO		SPO		TMC	
Policy abstraction	SQ	AUOC	SQ	AUOC	SQ	AUOC	SQ	AUOC
ReAct-core	71.1	51.6	99.8	89.6	92.7	69.6	56.6	43.8
AI Scientist v1-core	78.5	61.0	99.5	89.6	95.8	76.8	65.9	47.3
AI Scientist v2-core	92.4	67.2	100.0	89.7	98.0	79.7	73.5	49.7
AI Researcher-core	89.9	62.0	99.9	88.3	97.3	76.4	80.4	50.7

Table K.12: Behaviorally matched scientific-agent policy abstractions in the compatible-core optimizer. Each row instantiates the named system's high-level search policy under the shared deterministic compatible-core protocol; the comparison concerns policy behavior rather than end-to-end language-model execution.