The Essay as a Pipeline

External Memory, Asymmetric Models, and Inspectable Repair in Local Language-Model Writing

Flyxion

August 2026

Abstract

The ordinary interface to a large language model encourages a misleading picture of machine-assisted writing. A prompt is supplied, a text is returned, and the resulting essay is evaluated as though it were the product of a single cognitive act. This paper develops and examines a different architecture in which writing is decomposed into a sequence of materially persistent transformations. A topic is first converted into an argumentative outline, the outline into a draft, the draft into an independent critical review, and the combination of draft and review into a proposed revision. The implementation considered here uses local Granite 4.1 models through Ollama, assigning the smaller 3B model primarily to generative operations and the larger 8B model to criticism. The significance of the arrangement lies less in the particular models than in the separation of cognitive roles. Generation and evaluation are treated as distinct operations, intermediate representations are externalized into ordinary files, and the entire apparatus can be inspected through the filesystem and Git history rather than disappearing into an opaque conversational context.

This architecture suggests that language-model writing is better understood as a trajectory through a space of textual artifacts than as a mapping from prompt to answer. An outline constrains a draft without determining it; a review diagnoses properties of the draft without directly rewriting it; and a revision attempts to repair the diagnosed defects while preserving material that survived criticism. Because each state persists independently, disagreement between diagnosis and repair becomes observable. The experiments examined here demonstrate why this distinction matters. The larger reviewing model can identify weak definitions, unsupported transitions, hidden assumptions, and conclusions stronger than their argumentative warrant even when the subsequent revision fails to incorporate those criticisms adequately. Such a case is not merely a failed generation. It is evidence that diagnostic competence and repair competence are separable properties of a model-mediated writing system.

The paper therefore treats the essay pipeline as a small experimental cognitive architecture. It develops a formal account of staged generation, externalized memory, asymmetric allocation of computational capacity, critical diagnosis, conservative revision, and

provenance-preserving textual evolution. The central claim is that the epistemic value of such a system does not consist in guaranteeing better prose at every stage. Its value consists in converting an otherwise transient act of generation into an inspectable process whose intermediate commitments, criticisms, failures, and repairs remain available for comparison. A writing pipeline becomes scientifically interesting when it preserves enough of its own history for the difference between what a system noticed and what it successfully changed to become an object of analysis.

1 Introduction: From Generated Text to Textual Process

A language model presented through a conversational interface appears to perform writing as a nearly instantaneous transformation. The user supplies an instruction, the model produces a sequence of tokens, and the completed sequence is presented as the answer. Even when the model internally performs operations that resemble planning, reconsideration, or evaluation, the ordinary artifact exposed to the user is usually the terminal text. This interface naturally encourages the representation of machine writing as a function from an input prompt to an output document. If p denotes a prompt and e the resulting essay, the interaction appears to have the simple form

$$F(p) = e.$$

Such a representation is convenient, but it suppresses precisely those distinctions that become important when language models are used as components of a repeatable intellectual workflow. It does not tell us whether the system first constructed an argumentative plan, whether it distinguished generation from criticism, whether a defect was recognized before being repaired, whether a later change preserved or destroyed an earlier successful passage, or whether the final text is better because of a particular intermediate operation. The function F records only the endpoints. It hides the trajectory.

The experimental repository considered in this paper begins from the opposite assumption. Writing is treated not as a single generation but as an ordered sequence of transformations whose intermediate products deserve to exist as independent objects. The implementation is intentionally modest. It consists principally of shell scripts, textual prompt templates, a small Python substitution utility, local language models served through Ollama, and ordinary directories containing generated Markdown files. This simplicity is methodologically useful. There is no elaborate orchestration framework whose abstractions obscure the movement of information. The state of the writing process can be inspected with standard Unix tools. A topic exists as a text file. An outline exists as a Markdown file. A draft exists separately from the review written about it. The revised essay exists separately from both. The pipeline therefore makes the history of the essay materially visible.

The initial model allocation is deliberately asymmetric. The Granite 4.1 3B model serves primarily as the inexpensive generative model, while Granite 4.1 8B is assigned the more demanding role of criticism, evaluation, and revision guidance. This arrangement should not be interpreted as a claim that model size maps monotonically onto a single scalar quantity called intelligence. The more interesting hypothesis is architectural. If computational resources are heterogeneous, then it may be unnecessary to allocate the most capable available model uniformly to every transformation. Planning and first-pass generation may tolerate a relatively inexpensive model, while critical inspection may benefit disproportionately from additional capacity. A pipeline can therefore allocate models according to functional role rather than treating one model as an indivisible author.

The resulting process has four principal stages. Given a topic, the first stage asks the generative model to construct an argumentative outline. The prompt explicitly requires a central argument, a conceptual problem, a thesis, logically dependent development, a serious objection or limitation, and consequences. The second stage supplies both topic and outline to the same generative model and requests a substantial analytical essay. The third stage changes both task and model. The larger model receives the draft and is explicitly instructed not to rewrite it. Its function is diagnostic: it searches for conceptual gaps, unsupported transitions, ambiguities, repetition, weak definitions, hidden assumptions, and conclusions whose strength exceeds their argumentative warrant. The fourth stage returns to the generative model, supplying both the original draft and the independent review. It is instructed to preserve claims and passages that survived criticism, to change material only where doing so improves the argument, and to prioritize substantive conceptual repair over stylistic modification.

This separation is important because criticism and rewriting are not the same operation. If the reviewing model were simply asked to produce a better essay, its output would erase the distinction between recognizing a defect and constructing a successful repair. A fluent replacement could conceal whether the original defect had actually been identified. Conversely, an explicit review can be excellent even when the later revision is poor. By storing the review as its own artifact, the architecture permits these possibilities to be distinguished empirically. The pipeline can therefore fail in more informative ways than a one-shot generator. It can produce a weak outline, a weak realization of a good outline, a strong diagnosis of a weak draft, an inadequate repair of a correctly diagnosed defect, or even a revision that damages material the reviewer did not object to. Each failure occurs at a different transformation and consequently suggests a different intervention.

The experimental outputs already make this distinction concrete. In the essay concerning continuation and trajectories, the larger reviewer identifies a serious conceptual problem: the draft moves too easily from the existence of a trajectory to an assumption of continuity, even though trajectories may contain jumps or other discontinuities. It also questions whether

the proposed framework is genuinely universal across disciplines and whether claims about improved prediction have been adequately supported. These are not merely corrections of diction. They are objections to the inferential structure of the argument. The reviewer is therefore performing something recognizably different from the initial generator. It is evaluating relations among claims rather than merely extending a topic into prose.

The admissibility experiment exposes the complementary problem. The reviewer identifies vague definitions, missing measurable criteria, abrupt transitions, hidden assumptions, and an overextended conclusion. Yet the existence of a good review does not logically entail that the revision operator will successfully implement it. Indeed, comparison of the preserved artifacts can reveal cases in which the final essay remains extremely close to the draft despite substantive criticism. This observation is central to the present analysis. A pipeline with persistent intermediate states does not merely attempt to improve writing. It generates evidence about its own transformations. A discrepancy between review and revision becomes an observable experimental result rather than disappearing inside an undifferentiated request to "make the essay better."

The filesystem consequently plays a role analogous to external memory. This claim should not be inflated into the assertion that a directory is itself a cognitive agent. The point is narrower and more operational. A language model invocation is transient. Once its output has been consumed by another invocation, the internal circumstances of generation are not ordinarily available to subsequent stages unless they are reconstructed in context. By writing the output to a file, the pipeline converts a transient generation into a persistent representation. Later operations can read that representation exactly as it was produced. Human investigators can inspect it independently. External programs can compare it, hash it, version it, search it, or subject it to additional analysis. The file is therefore not merely output in the colloquial sense. It is a state variable in a larger computational process.

Git extends this principle from state persistence to historical persistence. A filesystem records what exists now; version control can record how the apparatus and its artifacts came to exist. The repository was intentionally developed as a sequence of commits so that the growth of the experimental machinery itself becomes part of the record. This gives the system two temporal dimensions. Within an individual essay run, there is the trajectory from topic through outline, draft, review, and revision. Across development of the repository, there is a second trajectory describing changes to scripts, prompts, tests, and experimental procedures. The writing process has provenance, and the apparatus that produced the writing has provenance as well.

The purpose of this paper is therefore not to claim that a four-stage shell script constitutes an autonomous author. Nor is it to claim that the resulting essays are necessarily superior

to texts generated directly by a larger model. The stronger and more defensible claim is methodological. Once language-model writing is decomposed into explicit stages, persisted as artifacts, and subjected to asymmetric evaluation, properties that are invisible in one-shot generation become measurable. We can ask whether the outline constrains the draft, whether the reviewer detects defects that the drafting model misses, whether revision responds to the review, whether accepted passages remain stable, whether successive repairs converge, and whether additional computational effort is more valuable when allocated to generation or diagnosis. The unit of analysis ceases to be the final essay alone. It becomes the transformation history that produced the essay.

This shift from product to process also changes what should count as success. A conventional writing benchmark might assign a scalar score to the final document. Such a score can still be useful, but it collapses a multidimensional process into one terminal judgment. In the pipeline architecture, success can instead be decomposed. An outline may be structurally adequate even when its draft realization is poor. A reviewer may correctly identify a defect even when no available revision operation can repair it. A revision may improve argumentative warrant while reducing stylistic elegance. A final essay may receive a higher holistic score while nevertheless containing a newly introduced conceptual error. The architecture makes these possibilities explicit because it preserves the boundaries across which they occur.

The deeper motivation is thus inspectability rather than automation for its own sake. The pipeline does not attempt to remove the human observer from writing. It creates additional places at which observation can occur. Every intermediate artifact is a diagnostic surface. Every transformation can in principle be compared with its input. Every review can be compared with the revision it was supposed to motivate. The system is useful precisely because its failures need not be compressed into the statement that "the model wrote a bad essay." A bad final result can be traced backward through a chain of transformations until the relevant failure becomes more precisely describable. In this respect, the architecture treats writing less like the emission of a document and more like the evolution, diagnosis, and repair of a representational object.

1.1 Formalization: Writing as an Artifact Trajectory

Let $\mathcal{X}$ denote the space of textual artifacts and let $t \in \mathcal{X}$ be a topic representation. A one-shot writing system can be represented by a map

$$F : \mathcal{X} \to \mathcal{X},$$

with

$$e = F(t).$$

The only externally represented states are therefore t and e.

The experimental pipeline instead introduces distinct operators

$$\mathcal{O}, \mathcal{D}, \mathcal{R}, \mathcal{V},$$

where

$$\mathcal{O} : \mathcal{X} \to \mathcal{X},$$

$$\mathcal{D} : \mathcal{X} \times \mathcal{X} \to \mathcal{X},$$

$$\mathcal{R} : \mathcal{X} \to \mathcal{X},$$

and

$$\mathcal{V} : \mathcal{X} \times \mathcal{X} \to \mathcal{X}.$$

For a topic t, define

$$o = \mathcal{O}(t),$$

$$d = \mathcal{D}(t, o),$$

$$r = \mathcal{R}(d),$$

and

$$e = \mathcal{V}(d, r).$$

The complete essay trajectory is therefore

$$\tau(t) = (t, o, d, r, e).$$

Notice that r is not itself an essay state in the same semantic sense as d and e. It is a diagnostic representation whose referent is d. Consequently the pipeline is not merely a Markov chain over homogeneous documents. It is more accurately represented as a typed transformation graph,

$$t \xrightarrow{\mathcal{O}} o, \qquad (t, o) \xrightarrow{\mathcal{D}} d, \qquad d \xrightarrow{\mathcal{R}} r, \qquad (d, r) \xrightarrow{\mathcal{V}} e.$$

The distinction matters because the review operator is intentionally prevented from collapsing into the revision operator.

Let $P : \mathcal{X} \to \mathcal{A}$ be a persistence operation mapping a transient model output to a stable external artifact. The implemented process is then more precisely expressed as

$$o = P(\mathcal{O}(t)),$$

$$d = P(\mathcal{D}(t, o)),$$

$$r = P(\mathcal{R}(d)),$$

$$e = P(\mathcal{V}(d, r)).$$

Persistence changes the epistemic structure of the experiment even when it does not change the textual content of any output. Without P, an intermediate result may be consumed and discarded. With P, each result remains independently addressable.

Definition 1.1 (Inspectable trajectory)**.** *A writing trajectory*

$$\tau = (x_0, x_1, \ldots, x_n)$$

is inspectable *when every x_i required to explain a subsequent transformation is externally recoverable without regenerating x_i.*

The implemented essay pipeline satisfies this criterion because topic, outline, draft, review, and revised essay are separately persisted.

Proposition 1.2 (Persistence strictly increases retrospective distinguishability)**.** *Suppose two writing processes produce the same terminal essay e but have different intermediate trajectories,*

$$\tau_1 = (t, o_1, d_1, r_1, e)$$

and

$$\tau_2 = (t, o_2, d_2, r_2, e),$$

with $\tau_1 \neq \tau_2$. If only t and e are retained, the two processes are observationally indistinguishable with respect to their intermediate histories. If every intermediate state is persisted, they are distinguishable whenever at least one corresponding intermediate artifact differs.

Proof. Under endpoint-only observation, define the observation operator

$$E(\tau) = (x_0, x_n).$$

Then

$$E(\tau_1) = (t, e) = E(\tau_2),$$

so no observation available through E distinguishes the two trajectories.

Now define the persistent observation operator

$$P_\tau(\tau) = (x_0, x_1, \ldots, x_n).$$

Because $\tau_1 \neq \tau_2$, there exists some index k such that

$$x_k^{(1)} \neq x_k^{(2)}.$$

Therefore

$$P_\tau(\tau_1) \neq P_\tau(\tau_2).$$

Hence persistence strictly enlarges the class of process distinctions available to retrospective inspection. □ □

This elementary result has an important methodological consequence. Two systems can produce indistinguishable final essays while reaching them through radically different processes. One may have generated a strong draft immediately and received a nearly empty review. Another may have generated a weak draft, received an excellent diagnosis, and successfully repaired it. Endpoint evaluation assigns these processes the same observable result even though they demonstrate different competencies.

We can similarly separate diagnosis from repair. Let

$$\Delta(d)$$

denote the set of substantive defects present in draft d, and let

$$C(r, d) \subseteq \Delta(d)$$

denote the defects correctly identified by review r. Define diagnostic recall as

$$\delta(r, d) = \frac{|C(r, d)|}{|\Delta(d)|},$$

whenever $|\Delta(d)| > 0$.

Let

$$\rho(d, r, e)$$

denote the subset of correctly diagnosed defects that are actually repaired in e. Conditional repair effectiveness may then be written

$$\eta(e \mid d, r) = \frac{|\rho(d, r, e)|}{|C(r, d)|},$$

whenever $|C(r, d)| > 0$.

A high value of δ therefore does not imply a high value of η.

Theorem 1.3 (Diagnosis–repair non-equivalence)**.** *There exists no general implication*

$$\delta(r, d) \to \eta(e \mid d, r)$$

under the staged architecture unless additional constraints are imposed on the revision operator.

Proof. Consider a draft d containing a nonempty defect set

$$\Delta(d) = \{q_1, \ldots, q_m\}.$$

Suppose the reviewer identifies every defect correctly, so

$$C(r, d) = \Delta(d)$$

and therefore

$$\delta(r, d) = 1.$$

Now choose a revision operator satisfying

$$\mathcal{V}(d, r) = d.$$

No diagnosed defect is repaired, so

$$\rho(d, r, e) = \varnothing$$

and hence

$$\eta(e \mid d, r) = 0.$$

Thus perfect diagnosis is compatible with zero repair effectiveness. Consequently diagnostic competence and repair competence cannot be identified without further assumptions about $\mathcal{V}$. □ □

The theorem formalizes the central empirical advantage of retaining the review separately from the final essay. If only d and e were visible, failure to improve might be attributed to failure of recognition. Once r is persisted, we can determine whether the system failed because it did not see the problem or because it saw the problem and failed to act upon it.

Finally, let the model allocation function be

$$\mu : \{\mathcal{O}, \mathcal{D}, \mathcal{R}, \mathcal{V}\} \to \{\mathcal{M}_s, \mathcal{M}_\ell\},$$

where $\mathcal{M}_s$ is the smaller model and $\mathcal{M}_\ell$ the larger model. The current architecture uses

$$\mu(\mathcal{O}) = \mathcal{M}_s, \qquad \mu(\mathcal{D}) = \mathcal{M}_s,$$

$$\mu(\mathcal{R}) = \mathcal{M}_\ell, \qquad \mu(\mathcal{V}) = \mathcal{M}_s.$$

If the computational cost of invoking model m is $c(m)$, with

$$c(\mathcal{M}_s) < c(\mathcal{M}_\ell),$$

then the cost of one pipeline execution is approximately

$$C_{\text{pipe}} = 3c(\mathcal{M}_s) + c(\mathcal{M}_\ell),$$

ignoring differences in prompt and output lengths. Uniform use of the larger model would instead cost

$$C_\ell = 4c(\mathcal{M}_\ell).$$

The asymmetric allocation therefore yields nominal savings

$$\Delta C = C_\ell - C_{\text{pipe}} = 3\big(c(\mathcal{M}_\ell) - c(\mathcal{M}_s)\big).$$

The important question is not merely whether $\Delta C > 0$, which follows immediately from the cost assumption, but whether the lost generative capability is compensated by concentrating the stronger model at the diagnostic boundary. Let $g_i(m)$ denote the marginal quality contribution obtained by assigning model m to stage i. The asymmetric architecture is preferable to uniform large-model allocation under a cost-sensitive objective whenever

$$\sum_i g_i(\mu(i)) - \lambda C_{\text{pipe}} \geq \sum_i g_i(\mathcal{M}_\ell) - \lambda C_\ell,$$

for a chosen computational cost coefficient $\lambda > 0$.

The experiments described in the following sections do not yet establish this inequality empirically. They establish something logically prior: the architecture makes its terms separately observable. Once generation, diagnosis, revision, persistence, and cost have been decomposed, comparative experiments can be designed to determine where additional model capacity actually produces the greatest marginal improvement. The shell pipeline is therefore not the conclusion of the experiment. It is the apparatus that makes the experiment possible.

2 Architecture: A Typed Pipeline of Persistent Transformations

The implementation is deliberately small enough that its architecture can be understood without appealing to an orchestration framework. At the center is a shell program, `essay-pipeline.sh`, whose responsibility is not to write an essay itself but to coordinate a sequence of transformations. The program receives a topic as a command-line argument, converts that topic into a filesystem-safe identifier, establishes a dedicated output directory, writes the original topic to `topic.txt`, and then invokes the language models repeatedly under different prompt conditions. Each invocation produces an artifact that becomes an explicit input to a later invocation. The shell therefore performs a function analogous to a scheduler: it determines which operation occurs next, which model performs it, which prior artifacts are available to it, and where the resulting representation is persisted.

This division of labor is significant because the shell contains almost none of the semantic content of the essay. It does not know what constitutes a good thesis, what counts as an unsupported transition, or how an objection should be answered. Those judgments are delegated to prompt-conditioned language models. Conversely, the models do not control the global procedure. The drafting model cannot decide to skip review, overwrite the original draft, silently replace the topic, or redirect the review into another stage. The surrounding program fixes the topology of the process. Model outputs vary, but the relations among stages remain externally imposed. The architecture therefore separates procedural control from semantic transformation.

The shared shell library, `scripts/common.sh`, makes this distinction explicit. It establishes two configurable model roles, with `granite4.1:3b` assigned by default to `DRAFT_MODEL` and `granite4.1:8b` assigned to `REVIEW_MODEL`. Before an experiment begins, the environment checker verifies both the existence of Ollama and the availability of the required models. Model invocation itself is reduced to a small primitive in which a prompt is written to standard input and consumed by `ollama run`. The surrounding scripts therefore treat a language model as a transform that can be called from an ordinary Unix process rather than as an application environment within which the rest of the experiment must live.

This inversion is methodologically useful. The model is embedded inside the experiment rather than the experiment being embedded inside the model interface. Filesystem operations, shell control flow, model selection, prompt construction, testing, logging, and version control remain available independently of any particular inference call. Replacing Granite with another local model family would alter an important experimental variable but would not require replacing the architecture itself. Likewise, the 3B and 8B assignments can be reversed, equalized, or varied stage by stage. The model is therefore one component of a larger writing machine.

Prompt construction is similarly externalized. Rather than embedding long instructions directly inside shell functions, the system stores the outline, drafting, review, and revision instructions as independent text files. A small Python utility, `render-prompt.py`, reads one of these templates and substitutes the contents of named files for placeholders such as `{{TOPIC}}`, `{{OUTLINE}}`, `{{ESSAY}}`, and `{{REVIEW}}`. This apparently mundane design choice establishes a useful representational boundary. A prompt is not simply a string constructed invisibly inside an invocation. It is assembled from persistent components whose provenance can be inspected separately. The instructions defining the cognitive role of a stage are stored independently from the artifacts upon which that stage operates.

The first stage begins with the least developed representation in the system: a topic. The topic is not supplied directly to a general request to "write an essay." Instead it is inserted into an outlining prompt that imposes structural expectations before prose generation begins. The model is asked to identify a conceptual problem, establish a central thesis, organize the essay into logically dependent sections, consider a serious objection or limitation, and derive consequences. The instruction that the essay should develop one central argument rather than a collection of loosely associated observations is especially important. It attempts to move structural decisions upstream, before the drafting model has accumulated enough prose that changing the argument becomes expensive.

The resulting `outline.md` is therefore not merely a convenience for the human reader. It is an intermediate representation. It compresses a proposed argumentative trajectory into a form that is more structured than the topic but less committed than the finished essay. This intermediate state constrains the next transformation while leaving considerable lexical and explanatory freedom. The outline can consequently be evaluated independently. If the final essay lacks argumentative coherence, one can inspect whether the incoherence was already present in the outline or emerged during realization. The persistence of the outline creates a diagnostic boundary between planning and drafting.

The second stage combines the original topic with the generated outline. This is an important detail because the draft is not generated from the outline alone. The topic remains independently available, reducing the possibility that distortions introduced during outlining become the only representation of the original task. The drafting prompt asks for a substantial analytical essay, instructs the model to treat the outline as argumentative structure rather than text to be mechanically repeated, requires important concepts to be defined when necessary, and distinguishes claims that follow from the argument from those that are merely suggestive. The draft is written to `draft.md`, producing a second persistent boundary.

The transition from outline to draft is qualitatively different from the transition from topic to outline. Outlining is primarily an operation over relations among prospective claims. Drafting

must instantiate those relations in language. It must choose definitions, examples, transitions, qualifications, and explanatory depth. Consequently a sound outline does not guarantee a sound draft. A model may possess enough planning competence to propose a defensible sequence of claims while lacking the local discipline necessary to realize those claims without introducing equivocation or overstatement. Because the two products are separated, such a discrepancy can be observed rather than inferred retrospectively from the final text.

The third stage introduces the architectural asymmetry. The draft is passed to the larger model under a prompt that explicitly changes its role from producer to critic. The reviewer is instructed not to rewrite the essay. Instead it must identify conceptual gaps, unsupported transitions, ambiguities, accidental repetition, weak definitions, hidden assumptions, and cases in which conclusions exceed what the preceding argument warrants. It must distinguish substantive defects from stylistic preferences and identify the revisions with the highest expected argumentative value. The resulting `review.md` is not a replacement document. It is a representation of defects and proposed interventions.

Preventing the reviewer from rewriting is more than a stylistic preference. If diagnosis and replacement occurred in the same inference, the experiment would lose access to the mapping between detected defect and proposed repair. A rewritten paragraph does not necessarily reveal why the model changed it. It may improve a passage without correctly diagnosing its weakness, or change a passage that was already adequate merely because another formulation was available. The independent review forces criticism into a representation that can itself be inspected. The experiment can ask whether the critic noticed a particular problem before asking whether a later model solved it.

The fourth stage recombines the draft with its review. The smaller model is returned to the generative role, but it is now operating under constraints supplied by the larger model's diagnosis. The revision prompt contains a particularly consequential instruction: material that survived criticism should be preserved, and change should occur only where it improves the argument. Revision is therefore represented not as unrestricted regeneration but as conservative repair. The ideal revision is not maximally different from the draft. It is minimally different subject to correction of substantive defects.

This principle distinguishes revision from replacement. If the system simply asked the 3B model to write the essay again after reading the critique, there would be no architectural preference for retaining successful structure. The resulting text might repair some problems while unnecessarily destroying passages that were already adequate. By explicitly instructing preservation, the pipeline introduces an asymmetry between accepted and diagnosed material. The review defines a tentative repair surface: portions of the draft implicated by criticism are candidates for intervention, whereas material outside that surface should remain stable unless

modification is required by dependencies elsewhere in the argument.

The four persisted files therefore correspond to different representational functions. The topic specifies the problem. The outline proposes an argumentative organization. The draft realizes that organization in prose. The review diagnoses the realization. The final essay records an attempted repair. Their coexistence is more important than any particular filename. A pipeline that overwrote `draft.md` during revision would be computationally capable of producing the same final essay, but experimentally poorer because the pre-repair state would disappear. Preservation transforms otherwise disposable intermediate products into evidence.

The repository contains additional scripts that reinforce this experimental interpretation. `test-models.sh` supplies the same small prompt independently to both Granite models, making direct behavioral comparison possible. `compare-models.sh` accepts an arbitrary prompt and persists the responses of both models separately. `review-essay.sh` exposes the review stage independently of the complete pipeline, allowing an existing essay to be subjected to the 8B critic without first regenerating it. `granite-essay.sh` exposes the outline-and-draft portion independently. The full pipeline is therefore not a monolithic program but a composition of operations that can also be tested in isolation.

The batch script extends the experiment along another dimension. Rather than modifying the architecture, it holds the pipeline fixed while changing the conceptual topic. Essays concerning repair and difference, admissibility and optimization, continuation and trajectories, and technological capability and diffusion are processed by the same sequence of transformations. This provides the beginnings of a repeated-measures experimental design. The architecture is held approximately constant while the semantic problem varies. The resulting outputs cannot yet support strong statistical conclusions, but they make it possible to ask whether recurrent behaviors belong to a particular topic or to the pipeline more generally.

This is especially important because a single successful essay would provide weak evidence about the architecture. Language-model outputs are stochastic, topic-sensitive, and strongly conditioned by the particular affordances of a prompt. A pipeline can appear effective because one topic happens to align well with a model's learned representations. Repeating the same procedure across theoretically distinct prompts begins to separate architectural properties from accidental success. If the 8B reviewer repeatedly identifies hidden assumptions that the 3B drafter repeatedly overlooks, that recurrence becomes more informative than any isolated instance. If revisions repeatedly fail to incorporate certain classes of criticism, that too becomes an architectural observation.

The entire system can thus be understood as a deliberately low-level experimental apparatus. Bash supplies control flow. The filesystem supplies persistent state. Prompt templates supply local task definitions. Python performs transparent textual substitution. Ollama supplies

model inference. Granite models provide heterogeneous semantic transforms. Git supplies historical provenance. None of these components alone constitutes the writing pipeline. The pipeline exists in the relations among them.

This relational perspective is important for understanding why a small collection of shell scripts can be more analytically useful than a much more sophisticated conversational interface. A conversational system may possess greater capabilities while exposing fewer boundaries. The shell pipeline has comparatively little hidden orchestration. Its weakness as an integrated product is simultaneously its strength as an experiment. Inputs and outputs cross visible interfaces. Intermediate states have filenames. Model roles are named. Prompt templates can be diffed. Failed stages terminate visibly. Outputs can be compared using ordinary textual tools. The architecture does not make cognition transparent in the strong sense of revealing the internal computation of the neural network, but it makes the *composition of cognitive operations* transparent.

This distinction between internal opacity and external inspectability should be maintained carefully. Nothing in the pipeline reveals why Granite selects a particular token, which latent features support a judgment, or what internal representation corresponds to an argumentative defect. The models remain black boxes at the level of their learned computation. What the architecture makes visible is the causal organization surrounding those black boxes. We know which artifact was supplied to which model under which role-defining prompt and which artifact was subsequently produced. This is weaker than mechanistic interpretability but stronger than observing only a final answer.

The resulting methodological position is therefore neither that language models are transparent nor that their opacity prevents structured experimentation. An opaque transform can participate in a transparent experimental protocol. Indeed, much of empirical science operates under analogous conditions: a system need not be completely understood internally before carefully controlled interventions and observations become informative. Here the intervention consists of changing model assignment, prompt structure, input artifact, or pipeline topology, while the observations consist of persistent textual states and their differences.

The architecture also establishes a natural boundary between orchestration errors and semantic errors. If Ollama is unavailable, a required model is absent, a file cannot be read, or a prompt template fails to render, the shell environment can detect a procedural failure before it is confused with poor writing. The scripts use strict shell behavior and explicit environment checks precisely for this reason. A semantic experiment is meaningful only after the apparatus has established that the intended computation actually occurred. This is a modest but essential form of experimental hygiene.

The same principle applies to reproducibility. Exact textual reproduction is not guaranteed

merely because the scripts and prompts are preserved; stochastic inference, model versions, runtime configuration, and sampling behavior may still alter outputs. Nevertheless, preservation of the apparatus dramatically narrows the space of unexplained variation. A future investigator can recover the prompt templates, the model identifiers, the transformation order, and the intermediate products from a particular run. Reproducibility should therefore be understood here in layers. The pipeline is procedurally reproducible even when token-level generation is not perfectly deterministic, and individual historical outputs remain directly inspectable even when they cannot be regenerated exactly.

The architecture's central achievement is consequently not automation but decomposition. A task ordinarily expressed as "write and improve an essay" has been factored into transformations whose inputs, outputs, and responsibilities differ. Once those boundaries exist, questions that were previously vague acquire operational form. Whether planning helps drafting can be studied by removing or replacing the outline. Whether a stronger model is disproportionately useful for criticism can be studied by swapping model assignments. Whether reviews produce repairs can be studied by comparing diagnosed defects against changes in the final essay. Whether conservative revision preserves successful material can be studied through textual similarity restricted to undiagnosed regions. Architecture creates the variables that subsequent experiments can manipulate.

2.1 Formalization: The Pipeline as a Typed Directed Computation Graph

Let the pipeline be represented by a directed acyclic graph

$$G = (V, E),$$

where vertices correspond to persistent artifacts or computational operators and edges represent information dependencies. Partition the vertices into artifact vertices

$$V_A = \{T, O, D, R, F\}$$

and operator vertices

$$V_\Omega = \{\Omega_o, \Omega_d, \Omega_r, \Omega_v\},$$

where T is the topic, O the outline, D the draft, R the review, and F the final essay.

The operator types are

$$\Omega_o : T \to O,$$

$$\Omega_d : T \times O \to D,$$

$$\Omega_r : D \to R,$$

and

$$\Omega_v : D \times R \to F.$$

The dependency relation is therefore

$$E = \{(T, \Omega_o), (\Omega_o, O), (T, \Omega_d), (O, \Omega_d), (\Omega_d, D), (D, \Omega_r), (\Omega_r, R), (D, \Omega_v), (R, \Omega_v), (\Omega_v, F)\}.$$

This representation makes explicit a property that a linear notation can obscure: the draft has two outgoing semantic roles. It is simultaneously the object of review and one of the inputs to revision. The review does not replace the draft as the revision input. Instead,

$$F = \Omega_v(D, R).$$

This preserves the original object being repaired alongside the diagnosis of that object.

Let

$$\mathcal{T}, \mathcal{O}, \mathcal{D}, \mathcal{R}, \mathcal{F}$$

denote the semantic types of topics, outlines, drafts, reviews, and final essays respectively. Although all are serialized as text, their computational types differ. Thus

$$T \in \mathcal{T}, \qquad O \in \mathcal{O}, \qquad D \in \mathcal{D}, \qquad R \in \mathcal{R}, \qquad F \in \mathcal{F}.$$

The distinction between serialization type and semantic type is crucial. At the operating-system level,

$$\mathcal{T} \cong \mathcal{O} \cong \mathcal{D} \cong \mathcal{R} \cong \mathcal{F} \cong \Sigma^*,$$

where Σ^* is the set of finite strings over an alphabet Σ. Semantically, however,

$$\mathcal{T} \neq \mathcal{O} \neq \mathcal{D} \neq \mathcal{R} \neq \mathcal{F}.$$

The prompt templates enforce these distinctions by assigning different admissibility conditions to outputs at each stage.

Let

$$A_i(x) \in \{0, 1\}$$

denote whether artifact x satisfies the role constraints associated with stage i. For example, an admissible review must diagnose rather than simply replace the essay. Thus an artifact can be

a perfectly valid string while failing its semantic type:

$$x \in \Sigma^* \quad \not\Rightarrow \quad A_r(x) = 1.$$

This gives the pipeline a typed interpretation despite its implementation in dynamically composed text files.

Definition 2.1 (Stage contract)**.** *A stage contract is a tuple*

$$K_i = (I_i, M_i, P_i, O_i, A_i),$$

where I_i is the required input type, M_i the assigned model, P_i the prompt transformation, O_i the expected output type, and A_i an admissibility predicate over outputs.

For the review stage,

$$K_r = (\mathcal{D}, M_{8B}, P_r, \mathcal{R}, A_r).$$

For the drafting stage,

$$K_d = (\mathcal{T} \times \mathcal{O}, M_{3B}, P_d, \mathcal{D}, A_d).$$

The distinction between stage contracts permits failures to be localized. If the review artifact violates A_r, the review stage has failed even if the final essay later happens to be good. Conversely, a review can satisfy A_r while the revision stage violates A_v.

Proposition 2.2 (Failure localization)**.** *Suppose each pipeline stage persists its output before that output is consumed by subsequent stages. Then any first stage k whose output violates its admissibility predicate A_k can be distinguished from failures introduced strictly after k.*

Proof. Let

$$x_0 \xrightarrow{\Omega_1} x_1 \xrightarrow{\Omega_2} \cdots \xrightarrow{\Omega_n} x_n$$

be a persisted sequence of stage outputs, and suppose

$$A_i(x_i) = 1$$

for every $i < k$, while

$$A_k(x_k) = 0.$$

Because x_k is persisted independently, it can be evaluated without reference to any later x_j, $j > k$. Therefore the violation exists prior to all transformations $\Omega_{k+1}, \ldots, \Omega_n$. Later stages may preserve, repair, or compound the defect, but they cannot be its point of introduction. Hence the earliest observable contract violation is localizable to Ω_k. □ □

The result is simple but important. In a one-shot architecture

$$T \xrightarrow{F} E,$$

all failures are observationally assigned to F. In a staged architecture, the failure space is partitioned.

Let

$$\mathcal{E} = \mathcal{E}_o \cup \mathcal{E}_d \cup \mathcal{E}_r \cup \mathcal{E}_v,$$

where the subsets correspond respectively to outlining, drafting, review, and revision errors. The architecture therefore increases the resolution of error attribution.

The model assignment can now be formalized as

$$\mu(\Omega_o) = M_s, \qquad \mu(\Omega_d) = M_s,$$

$$\mu(\Omega_r) = M_\ell, \qquad \mu(\Omega_v) = M_s,$$

where M_s and M_ℓ denote the smaller and larger Granite models.

Let the expected competence of model m on operator class i be

$$q(m, i),$$

and its computational cost be

$$c(m, i).$$

For an assignment μ, define total expected utility

$$U(\mu) = \sum_i w_i q(\mu(i), i) - \lambda \sum_i c(\mu(i), i),$$

where w_i represents the importance of competence at stage i and λ expresses the cost sensitivity of the experiment.

Uniform assignment of the larger model is optimal only if

$$U(\mu_\ell) \geq U(\mu)$$

for every heterogeneous assignment μ. There is no structural reason this must hold. In particular, if the marginal competence advantage

$$\Delta q_i = q(M_\ell, i) - q(M_s, i)$$

varies substantially by stage, then allocating the larger model only where

$$\frac{w_i \Delta q_i}{c(M_\ell, i) - c(M_s, i)}$$

is greatest can dominate uniform allocation under a fixed resource budget.

Proposition 2.3 (Role-sensitive allocation)**.** *Suppose the additional cost of using the larger model is positive at every stage,*

$$\Delta c_i = c(M_\ell, i) - c(M_s, i) > 0,$$

and a budget permits the larger model to be used at exactly one stage. Then the utility-maximizing assignment selects

$$i^* = \arg\max_i \left(w_i \Delta q_i - \lambda \Delta c_i \right).$$

Proof. Let U_s denote utility when the smaller model is used at every stage. Replacing M_s by M_ℓ only at stage i changes utility by

$$\Delta U_i = w_i \left[q(M_\ell, i) - q(M_s, i) \right] - \lambda \left[c(M_\ell, i) - c(M_s, i) \right].$$

Thus

$$\Delta U_i = w_i \Delta q_i - \lambda \Delta c_i.$$

Because exactly one replacement is permitted, maximizing total utility is equivalent to choosing the stage with maximal ΔU_i. Therefore

$$i^* = \arg\max_i \left(w_i \Delta q_i - \lambda \Delta c_i \right). \quad \square$$

□

The current assignment of the larger model to review can therefore be interpreted as an experimental hypothesis:

$$i^* = r.$$

It asserts, provisionally, that additional model capacity has unusually high marginal value when applied to diagnosis. The present experiments motivate this hypothesis but do not establish it. Establishing it would require controlled permutations of model assignments across otherwise equivalent runs.

The architecture also admits a formal account of conservative revision. Let a draft D be decomposed into atomic textual or argumentative units

$$D = \{d_1, d_2, \ldots, d_n\}.$$

Let the review induce a diagnostic indicator

$$\chi_R(d_i) = \begin{cases} 1, & \text{if } d_i \text{ is implicated by the review,} \\ 0, & \text{otherwise.} \end{cases}$$

Let

$$\delta(d_i, f_i)$$

measure the magnitude of change between draft unit d_i and corresponding final unit f_i. A conservative repair objective can then be written

$$J(F \mid D, R) = L(F) + \alpha \sum_{i=1}^{n} \big(1 - \chi_R(d_i)\big)\delta(d_i, f_i),$$

where $L(F)$ measures unresolved argumentative loss and $\alpha > 0$ penalizes unnecessary modification of material not implicated by criticism.

The revision problem becomes

$$F^* = \arg\min_F J(F \mid D, R).$$

This objective does not demand minimal textual change absolutely. A change to an undiagnosed passage can still be justified if it sufficiently reduces $L(F)$, for example because repairing one argument requires adjusting a dependent transition elsewhere. It does, however, encode the intuition expressed by the revision prompt: successful material should not be rewritten merely because rewriting is possible.

Finally, the architecture can be generalized beyond four stages. Let

$$X_0 = T$$

and let each subsequent state be generated by

$$X_{k+1} = P\left(\Omega_k(\text{parents}(X_{k+1}))\right),$$

where P persists the result. A pipeline is then a finite typed graph of model-mediated transformations rather than a fixed sequence of four calls.

The present system is one particularly simple instance:

$$T \to O \to D \to R \to F,$$

with additional dependency edges from T to D and from D to F. Its simplicity is advantageous

because every edge has an intelligible interpretation. Before adding additional agents, iterative loops, retrieval systems, scoring functions, or automatic convergence criteria, the four-stage architecture permits the foundational questions to be studied directly: what is gained by planning before generation, what is gained by separating criticism from production, what is gained by assigning heterogeneous models to heterogeneous roles, and what is gained by refusing to discard the states through which a text passes?

These questions concern the organization of computation rather than any particular model. The shell scripts make them concrete, but the architecture they instantiate is more general: intelligence supplied by a language model becomes experimentally tractable when it is placed inside a persistent, typed, and externally controlled process.

3 Asymmetric Cognition: Separating Generation from Criticism

The most consequential architectural decision in the experiment is not the use of two models but the refusal to treat writing competence as a single undifferentiated capability. A conventional comparison between language models tends to ask which model produces the better answer when both receive approximately the same task. Such comparisons are useful when models are competing substitutes, but they are less informative when models can instead become specialized components of a larger process. The present pipeline begins from the hypothesis that the operations required to produce an essay are heterogeneous enough that the model best suited to one stage need not be the model most economically suited to every stage. Generation, structural planning, diagnosis, and repair place different demands upon a language model, even though all four operations ultimately manipulate text.

This distinction becomes particularly important when local models differ substantially in computational cost. The Granite 4.1 3B model occupies approximately 2.1 GB in the local installation, whereas the Granite 4.1 8B model occupies approximately 5.3 GB. These storage figures are not direct measurements of inference cost, nor do they by themselves establish any precise ratio of computational expense, but they indicate the practical asymmetry that motivated the experiment. The smaller model is sufficiently inexpensive to invoke repeatedly for generative work. The larger model represents a more substantial local resource and can therefore be reserved for a stage at which additional capability is hypothesized to have greater marginal value. The pipeline assigns the 3B model to outlining, drafting, and revision, while concentrating the 8B model at the critical boundary between draft and revision.

The motivation for this arrangement is analogous to a familiar division of intellectual labor. Producing candidate material and determining whether candidate material is adequate are not identical problems. Generation benefits from fluency, associative reach, continuity, and

the ability to elaborate incomplete structures. Criticism instead requires resistance to the momentum of the existing text. A critic must ask whether a conclusion actually follows, whether a definition changes meaning during the argument, whether an example establishes what it is being asked to establish, and whether an apparently smooth transition conceals an unsupported inference. The generative operation asks, in effect, "what can follow from here?" The critical operation asks "what is licensed to follow from here?" These questions overlap, but they are not equivalent.

The distinction is particularly salient for autoregressive language models because fluent continuation is their native computational mode. Once a draft establishes terminology, assumptions, and rhetorical direction, subsequent generation is conditioned by that existing trajectory. This conditioning is useful for coherence, but it can also stabilize errors. A questionable premise introduced early in a draft may become increasingly difficult to notice because later passages are generated under a context in which that premise has already been normalized. A critic receiving the completed draft under an explicitly adversarial prompt occupies a different functional position. Its task is not to continue the trajectory but to evaluate the trajectory as an object.

This is one reason the review prompt explicitly prohibits rewriting. The prohibition creates a form of cognitive friction. The larger model cannot discharge criticism merely by producing prose that feels better. It must externalize its objections. When it identifies an unsupported transition, it must represent that transition as a problem rather than silently smoothing it. When it notices that a conclusion is stronger than the argument warrants, it must state the discrepancy. The review artifact consequently records something closer to a diagnostic map of the draft.

The distinction between continuation and evaluation can be seen in the actual experimental outputs. In the essay concerning continuation as a property of trajectories rather than states, the draft develops a broad conceptual argument across physics, biology, computation, and related domains. The reviewing model does not merely object to stylistic choices. It notices that the essay risks treating the existence of a trajectory as though it entailed continuity, even though trajectories may contain discontinuities, jumps, phase changes, or discrete transitions. This criticism targets a structural ambiguity in the central concept. The reviewer also challenges the breadth of the cross-domain claim and questions whether the asserted predictive consequences have actually been established. The critical operation is therefore testing the inferential warrant of the generated conceptual structure rather than simply proposing alternative sentences.

The experiment concerning admissibility and optimization provides a second example. There the reviewer asks for more precise definitions and measurable criteria, objects to transitions that move too rapidly between conceptual domains, identifies assumptions that have not been

defended, and recommends narrowing claims whose scope exceeds the evidence supplied in the draft. Again, the relevant competence is not merely linguistic polish. The reviewer is attempting to discriminate between a rhetorically coherent argument and an adequately warranted one.

These observations motivate a broader interpretation of model heterogeneity. A larger model need not be understood simply as a more expensive replacement for a smaller model. It can function as a scarce cognitive resource allocated to a particular transformation. The architecture thereby resembles heterogeneous computing, where different processors are assigned operations suited to their respective cost and performance characteristics. A system containing a CPU and a GPU does not normally ask which device is universally "better." It asks which computations benefit from which device. Likewise, a heterogeneous language-model pipeline can ask where additional representational capacity has the highest marginal value.

The analogy should not be taken too literally. The 3B and 8B models do not possess sharply disjoint computational specialties comparable to conventional hardware accelerators. Both can outline, draft, criticize, and revise. The specialization is imposed by the architecture rather than intrinsic to the models. This is precisely what makes the experiment interesting. Role differentiation can arise from orchestration even when the underlying components are general-purpose. The system creates functional specialization by assigning different prompts, inputs, and positions in the transformation graph.

Such imposed specialization has another advantage: it permits controlled role reversal. If the larger model were simply called whenever "difficult reasoning" seemed necessary, the criterion would remain informal. Once roles are explicit, the assignments can be permuted. One can run a $3B \to 3B \to 8B \to 3B$ pipeline, an $8B \to 8B \to 3B \to 8B$ pipeline, a uniform 3B pipeline, and a uniform 8B pipeline while preserving the same prompt topology. Differences among outputs can then be attributed more carefully to model allocation rather than to a completely different workflow.

The present experiment does not yet perform this full factorial comparison. Its current asymmetry is a hypothesis embodied in software. The hypothesis is that the marginal benefit of additional model capacity is unusually large at the diagnostic stage. This claim is plausible because diagnosis requires the model to resist the framing supplied by the draft, search for inconsistencies across relatively distant passages, distinguish substantive objections from stylistic alternatives, and rank possible interventions. But plausibility is not evidence. The value of implementing the hypothesis as an explicit assignment is that it becomes falsifiable through later permutations.

There is also a deeper reason to distinguish generative and critical competence. A model can produce a proposition without possessing the same reliability in determining whether that

proposition should survive. In human intellectual practice, these capacities are routinely separated through drafting, editing, peer review, replication, and adversarial criticism. Scientific institutions do not generally assume that the person who generates a hypothesis is thereby the uniquely qualified judge of its adequacy. The separation is not an insult to the generator; it is a recognition that creation and evaluation are subject to different biases.

Language-model systems often collapse these institutional distinctions into a single inference. A prompt may request an essay and then append instructions such as "check your reasoning carefully" or "make sure the argument is rigorous." Such instructions can improve output, but they preserve the identity of generator and evaluator within one transient operation. The resulting answer provides little evidence about whether self-criticism occurred, which defects were noticed, or whether an apparent correction resulted from diagnosis rather than alternative generation. The staged pipeline reconstructs, in miniature, an institutional separation that ordinary prompting tends to erase.

The use of a different model for criticism strengthens this separation further. If the same model instance drafts and reviews under different prompts, role differentiation still exists, but the two stages share the same learned parameters and potentially similar systematic blind spots. Introducing a second model size does not guarantee independence, particularly because both models belong to the same family, but it creates at least a modest source of heterogeneity. The reviewer is not simply the same computational configuration asked to reconsider its own immediately preceding continuation.

This point should be stated cautiously. The two Granite models are not independent observers in a statistical sense. Their training histories, architectural lineage, and learned representations are likely strongly related. Agreement between them cannot be interpreted as independent confirmation, and disagreement cannot be treated as though two unrelated experts had reached different conclusions. The experiment creates functional heterogeneity, not epistemic independence. Nevertheless, functional heterogeneity is sufficient to investigate whether different allocations of model capacity and task framing produce systematically different behaviors.

The review stage can therefore be understood as a diagnostic boundary. Before review, the system possesses a candidate argumentative object. After review, it possesses both that object and an explicit representation of suspected defects. The review does not itself determine which criticisms are correct. A human investigator may reject the reviewer's objection, discover an error the reviewer missed, or judge that a proposed revision would make the essay worse. What changes at the boundary is that potential defects become represented as first-class artifacts.

This creates an important distinction among error, detection, and warrant for repair. A draft may contain an actual defect that the reviewer fails to detect. A reviewer may diagnose

something that is not actually defective. A correctly identified defect may admit several possible repairs. A proposed repair may remove the local problem while damaging the global argument. The pipeline should therefore not be represented as a simple sequence in which review mechanically reveals truth and revision mechanically implements it. Review produces candidate diagnoses. Revision acts upon those diagnoses under uncertainty.

The conservative revision prompt implicitly recognizes this uncertainty by instructing the drafting model to preserve claims and passages that survived criticism. The instruction establishes a null intervention: leave the existing text unchanged. This matters because revision systems often possess a built-in bias toward modification. If a model is asked to "improve" a document, producing no change can appear unresponsive even when the document is already adequate. By making preservation explicit, the pipeline treats non-intervention as a legitimate outcome. A change must earn its place by repairing something.

This principle is closely related to the broader distinction between optimization and repair. An optimizer given a scalar objective may continue changing an artifact as long as some measured score can be increased. A repair system begins instead from an existing structure whose successful properties carry evidential weight. Its problem is not to replace that structure with the globally best imaginable alternative, but to restore or increase admissibility while preserving distinctions and relations that remain functional. In textual revision, this means that an effective paragraph should not be rewritten merely because another paragraph can be generated. Existing success is part of the constraint set.

The pipeline therefore contains two asymmetric relations. Computationally, the larger model is concentrated at review while the smaller model performs three generative transformations. Epistemically, criticism is permitted to identify candidate failures, but revision is instructed not to treat the entire draft as disposable. These asymmetries reinforce one another. The stronger model spends its capacity locating where intervention may be warranted; the cheaper model performs the intervention under a preservation constraint.

There is no guarantee that this division will succeed. Indeed, the preserved experimental artifacts reveal cases where it does not. A reviewer can produce a detailed and persuasive diagnosis while the revision remains close to the original draft. Such an outcome is particularly informative because it falsifies a naive assumption that once a language model has been told what is wrong, correction follows automatically. Information about a defect is not equivalent to the ability to transform an artifact so that the defect disappears without introducing another.

This distinction has consequences beyond essay writing. Many proposed agentic systems assume that iterative criticism will produce monotonic improvement: generate, critique, revise, repeat. But if diagnostic competence and repair competence are independent variables, iteration alone does not guarantee convergence. A system can enter a regime in which the same

defects are repeatedly identified but inadequately repaired, or in which successive repairs oscillate between competing formulations. Without persistent intermediate artifacts, such behavior may be mistaken for ordinary stochastic variation. With persistence, it becomes a trajectory that can be measured.

The present architecture should therefore be regarded as a minimal experiment in asymmetric cognition. It asks whether a small model can cheaply generate candidate structures, whether a larger model can identify defects that the smaller generator did not prevent, and whether those diagnoses can be converted back into successful repairs by the smaller model. The interesting object is not any one answer. It is the coupling among competencies.

3.1 Formalization: Generation, Diagnosis, and Repair as Distinct Competence Functions

Let X denote a textual state and let

$$\mathcal{E}(X)$$

denote the set of substantive defects in X relative to a specified argumentative standard. Because no automatic oracle for argumentative correctness is assumed, $\mathcal{E}$ should be understood as an ideal evaluative relation used for formal analysis rather than as a quantity directly available to the pipeline.

For a model M, define generative competence on task t as

$$G(M,t) = \mathbb{E}\left[Q\big(M(t)\big)\right],$$

where Q is an appropriate quality functional.

Define diagnostic competence as

$$D(M,X) = \mathbb{E}\left[\frac{|\widehat{\mathcal{E}}_M(X) \cap \mathcal{E}(X)|}{|\mathcal{E}(X)|}\right],$$

where

$$\widehat{\mathcal{E}}_M(X)$$

is the defect set proposed by model M.

Diagnostic precision can separately be represented as

$$P(M,X) = \mathbb{E}\left[\frac{|\widehat{\mathcal{E}}_M(X) \cap \mathcal{E}(X)|}{|\widehat{\mathcal{E}}_M(X)|}\right].$$

These quantities distinguish failure to notice genuine defects from the generation of spurious criticism.

Now define a repair operator

$$\rho_M(X, \widehat{\mathcal{E}}) = X',$$

and let successful defect removal be

$$S(X, X') = \mathcal{E}(X) \setminus \mathcal{E}(X').$$

The repair recall of M, conditional on a diagnosis, can then be expressed as

$$R(M, X, \widehat{\mathcal{E}}) = \frac{|S(X, X') \cap \widehat{\mathcal{E}}|}{|\mathcal{E}(X) \cap \widehat{\mathcal{E}}|}.$$

This quantity measures how many correctly diagnosed defects actually disappear after revision. It does not yet penalize newly introduced defects. Define

$$N(X, X') = \mathcal{E}(X') \setminus \mathcal{E}(X)$$

as the set of novel defects introduced by repair. A net repair functional may then be written

$$\mathcal{R}_{\text{net}} = \alpha|S(X, X')| - \beta|N(X, X')| - \gamma C(X, X'),$$

where $C(X, X')$ measures unnecessary structural change and

$$\alpha, \beta, \gamma > 0.$$

The separation of these terms establishes that successful revision requires more than defect removal. A revision that eliminates one problem while introducing three equally serious problems may have negative net repair value.

Proposition 3.1 (Generation does not imply diagnosis)**.** *There exists no general implication*

$$G(M, t) \textit{ high} \quad \Longrightarrow \quad D(M, X) \textit{ high}.$$

Proof. Consider a model M whose generative distribution strongly favors fluent realizations of patterns associated with high-quality essays but whose evaluator, when conditioned on an already generated artifact, fails to discriminate a particular class of invalid inferences. Let t be drawn from a task distribution on which that invalid inference rarely becomes necessary

during generation. Then M can achieve high expected quality

$$G(M, t)$$

while possessing arbitrarily poor recall for that defect class when it occurs in an externally supplied X. Hence generative success over t does not entail diagnostic success over arbitrary X. The competencies are related empirically but not logically identical. □ □

The converse also fails.

Proposition 3.2 (Diagnosis does not imply generation)**.** *A model may have high diagnostic competence over a class of essays while having lower generative competence for constructing essays in that class from sparse topics.*

Proof. Let the diagnostic task provide a complete candidate essay X, thereby supplying propositions, definitions, transitions, and argumentative dependencies explicitly in context. Let the generative task provide only a topic t. Diagnosis requires discrimination among relations already represented in X, whereas generation requires construction of those relations from a much less constrained state. There is no logical requirement that competence at the former transformation determine competence at the latter. Therefore models can be ordered differently under D and G. □ □

These results justify treating model capability as a vector rather than a scalar. For model M, define

$$\mathbf{c}(M) = \begin{pmatrix} c_o(M) \\ c_d(M) \\ c_r(M) \\ c_v(M) \end{pmatrix},$$

where the components represent competence in outlining, drafting, reviewing, and revising.

The simplistic scalar model assumes

$$c_o(M) \approx c_d(M) \approx c_r(M) \approx c_v(M)$$

up to a common monotone factor representing general capability. The pipeline instead permits

$$c_i(M_a) > c_i(M_b)$$

while simultaneously

$$c_j(M_a) \leq c_j(M_b)$$

for distinct tasks $i \neq j$, particularly after computational cost is incorporated.

Let the effective competence per unit cost be

$$\epsilon_i(M) = \frac{c_i(M)}{\kappa_i(M)},$$

where $\kappa_i(M) > 0$ is the cost of assigning M to stage i. A smaller model can therefore be the rational assignment to drafting even if

$$c_d(M_\ell) > c_d(M_s),$$

provided

$$\epsilon_d(M_s) > \epsilon_d(M_\ell).$$

The heterogeneous allocation problem is then

$$\max_{\mu} \sum_i w_i c_i(\mu(i))$$

subject to

$$\sum_i \kappa_i(\mu(i)) \leq B,$$

where B is the available computational budget.

The current architecture instantiates the assignment

$$\mu = (M_s, M_s, M_\ell, M_s).$$

The hypothesis that review is the most valuable location for the larger model can be represented as

$$\frac{c_r(M_\ell) - c_r(M_s)}{\kappa_r(M_\ell) - \kappa_r(M_s)} > \frac{c_i(M_\ell) - c_i(M_s)}{\kappa_i(M_\ell) - \kappa_i(M_s)}$$

for

$$i \in \{o, d, v\}.$$

This inequality is not assumed as a theorem about language models. It is the empirical conjecture encoded by the pipeline.

A second formal problem concerns whether iterative critique and revision necessarily converge. Let

$$X_{n+1} = \rho_M \left(X_n, \widehat{\mathcal{E}}_{M_r}(X_n) \right),$$

where M_r is the reviewer and M the reviser.

A common intuition is that repeated application yields

$$Q(X_{n+1}) \geq Q(X_n).$$

Nothing in the architecture guarantees this.

Theorem 3.3 (Non-monotonicity of unconstrained iterative revision)**.** *Without a repair admissibility condition, repeated critique–revision cycles do not guarantee monotonic improvement in textual quality.*

Proof. Suppose X_n contains a defect e_1, and the reviewer correctly identifies it. Let the revision operator produce X_{n+1} such that

$$e_1 \notin \mathcal{E}(X_{n+1}),$$

but introduce new defects

$$e_2, e_3 \in \mathcal{E}(X_{n+1})$$

with combined severity greater than that of e_1. For an additive defect loss

$$L(X) = \sum_{e \in \mathcal{E}(X)} w(e),$$

choose weights satisfying

$$w(e_2) + w(e_3) > w(e_1).$$

Then

$$L(X_{n+1}) > L(X_n),$$

and therefore, for quality inversely related to loss,

$$Q(X_{n+1}) < Q(X_n).$$

Thus successful repair of a diagnosed local defect is insufficient to guarantee global monotonic improvement. □ □

A stronger revision rule requires comparison with the null intervention. Let

$$\rho_0(X) = X$$

represent doing nothing. A proposed repair ρ is admissible only if

$$\mathbb{E}\left[L(\rho(X, R))\right] + \lambda C(\rho) < L(\rho_0(X)),$$

where $C(\rho)$ includes structural disruption, computational cost, and uncertainty associated with the intervention.

This gives the repair criterion

$$\Delta_\rho = L(X) - \mathbb{E}[L(\rho(X, R))] - \lambda C(\rho) > 0.$$

Definition 3.4 (Warranted textual repair)**.** *A textual intervention ρ is warranted relative to draft X and review R when its expected reduction in argumentative loss exceeds its expected intervention cost:*

$$\Delta_\rho > 0.$$

The definition clarifies why a review is not itself a command. The review supplies evidence relevant to estimating

$$\mathbb{E}[L(\rho(X, R))],$$

but it does not prove that any particular intervention satisfies the inequality. Diagnosis identifies a candidate repair surface; warrant requires an additional comparison among possible repairs and the null intervention.

Finally, define the complete asymmetric writing system as

$$\mathcal{W} = (G_s, C_\ell, R_s, P, H),$$

where G_s denotes small-model generation, C_ℓ large-model criticism, R_s small-model repair, P artifact persistence, and H historical preservation.

The system's epistemic output is not merely the terminal essay X_f. It is

$$\mathcal{O}_\mathcal{W} = \left(X_0, X_1, \ldots, X_f, C_1, \ldots, C_k, H\right).$$

The architecture therefore produces both an artifact and evidence concerning the transformations that generated it.

This distinction will become central in the next stage of the analysis. Once intermediate representations are persistent, the filesystem ceases to be merely a destination for model output. It becomes the external state space through which otherwise transient model invocations communicate. The next section therefore examines the role of files, directories, prompt templates, and version history as a deliberately constructed memory substrate for local language-model cognition.

4 The Filesystem as External Memory: Persistence, Addressability, and Provenance

The language models participating in the pipeline are invoked as transient processes. Each invocation receives a prompt, produces a textual response, and terminates. Considered in isolation, such an invocation has no durable relation to the invocation that precedes or follows it. The outlining model does not remain active while the draft is written; the drafting invocation does not somehow persist inside the reviewer; and the reviewer does not communicate with the reviser through an enduring shared internal state. Continuity is instead constructed outside the models. The filesystem carries information across otherwise discontinuous inference events.

This feature is easy to overlook because files are ordinarily regarded as implementation details. A generated outline must be stored somewhere, and a file appears to be merely the obvious place to put it. In the present architecture, however, persistence is doing considerably more conceptual work. The file converts an ephemeral model response into an independently addressable object. Once `outline.md` exists, the next model invocation need not reproduce the cognitive circumstances under which the outline was generated. It needs only the artifact. The pipeline thereby separates the production of a representation from the subsequent availability of that representation.

The distinction can be illustrated by comparing conversational context with filesystem persistence. In an ordinary conversation with a language model, an earlier response may remain available because it is included in the context supplied to a later inference. This creates a form of continuity, but the continuity is bound to the conversational substrate. The earlier representation exists as part of an accumulated interaction history, often surrounded by additional instructions, corrections, and dialogue. In the shell pipeline, the relevant state is extracted from this conversational mode and given an explicit address. The outline is not merely "something the model said earlier." It is a particular object at a particular path whose contents can be read without reconstructing the interaction that produced it.

Addressability changes what can be done with an intermediate representation. A file can be supplied to several different transformations. It can be compared with another file. It can be hashed to determine whether it changed. It can be searched, versioned, copied, annotated, or replaced independently of the model that created it. It can be examined by a human before the pipeline proceeds. It can be used as the input to an entirely different model family. It can remain available after every inference process has terminated. Persistence therefore enlarges the set of operations that can be performed on model output without requiring the original generator to participate.

The directory structure makes the temporal organization of each experiment visible. A sin-

gle topic receives its own output directory containing `topic.txt`, `outline.md`, `draft.md`, `review.md`, and `essay.md`. These names encode a developmental sequence, but they also preserve branching dependencies. The final essay does not replace the draft, and the review does not become part of the draft merely because it influences the revision. Each representation retains its identity. The resulting directory is therefore a compact record of an argumentative process.

This is particularly important when a later state fails to improve upon an earlier one. If revision overwrote the draft, the experiment would preserve only the result of attempted repair. Determining whether the repair helped would require recovering the previous state from some other source, if such a source existed at all. By retaining both `draft.md` and `essay.md`, the pipeline makes the comparison immediate. The review can then be positioned between them conceptually even though all three coexist physically. One can ask whether a defect mentioned in `review.md` is present in `draft.md`, whether it disappears from `essay.md`, and what collateral changes accompany its attempted removal.

The architecture thus transforms files into experimental observations. The distinction between an artifact and an observation is useful here. Every output is an artifact in the sense that it was produced by a computational process. It becomes an observation when the experiment treats its preserved state as evidence about that process. The draft is not only something from which to generate the final essay. It is evidence about what the drafting stage was capable of producing under a particular outline and topic. The review is evidence about what the reviewing stage detected. The final essay is evidence about what the revision stage did when supplied with both the draft and its diagnosis.

This interpretation also clarifies the role of the small Python prompt renderer. Its operation is almost trivial: it reads a template, reads named files, substitutes their contents for placeholders, and writes the resulting prompt. Yet this triviality is valuable because it keeps information transfer explicit. The review stage receives the contents of the draft because the renderer substitutes the draft file into the review template. The revision stage receives both draft and review because both are inserted into the revision template. There is no hidden conversational memory that must be assumed to explain how information moved from one stage to another. The dependencies can be reconstructed directly from the scripts.

The pipeline consequently possesses a form of memory without requiring any model to possess persistent memory. This statement requires precision. The filesystem does not remember in the phenomenological or cognitive sense associated with a human subject. It stores representations according to rules established by the surrounding program. Nevertheless, at the level of system architecture, storage performs the functional role required for information from an earlier computation to constrain a later one. If memory is operationally defined as the persistence of

information across temporally separated transformations, then the filesystem is unquestionably part of the system's memory mechanism.

This externalization has an important consequence for model size. A larger context window is one way to preserve more information across a complex task, but it is not the only way. The pipeline instead decomposes the task and selectively reconstructs context from persisted artifacts. The drafting model receives the topic and outline rather than an entire transcript of the outlining interaction. The reviewer receives the draft and its role instructions rather than the history of how the draft was produced. The reviser receives the draft and review. Context is therefore assembled according to causal relevance rather than accumulated indiscriminately.

Such selective reconstruction can be understood as a form of representational compression. The outline compresses the planning stage into an artifact intended to retain argumentative structure while discarding the transient token-by-token process that produced it. The review compresses evaluation into a representation of diagnosed weaknesses and recommended interventions. Neither artifact preserves everything that occurred inside the generating model, nor could it. What matters is whether it preserves enough structure for the next operation.

This introduces the possibility of memory failure at the architectural level. An outline may omit a distinction that was implicit in the topic. A review may fail to represent an important defect. A prompt renderer may faithfully transfer an artifact that is itself an inadequate summary of the preceding stage. External memory therefore does not solve the representation problem; it relocates it. The question becomes not merely whether information persists, but whether the representation selected for persistence is sufficient for admissible continuation of the writing process.

The topic file provides an elementary example of why redundant persistence can be useful. The drafting stage receives both topic and outline. If the outline were treated as a perfect sufficient statistic for the topic, the original topic could be discarded after outlining. The implementation refuses this assumption. By supplying both, it allows the draft to remain constrained by the original problem even if the outline has distorted or omitted part of it. The architecture therefore preserves an earlier state across more than one transition when that state remains causally relevant.

The same logic explains why the revision stage receives the original draft alongside the review. A review is not assumed to be a lossless representation of the draft. It contains claims about the draft, not the draft itself. Attempting revision from the review alone would force the reviser to reconstruct the object of criticism from a lossy diagnostic summary. By preserving both, the pipeline maintains a distinction between an object and metadata about that object.

This yields a general architectural principle: derived representations should not automatically

replace the representations from which they were derived. Replacement is justified only when the derived representation is known to preserve everything required by downstream operations. In the absence of such a guarantee, retaining both increases informational redundancy but decreases the risk that a premature compression destroys something subsequently needed.

The storage cost of this redundancy is negligible for textual essays. The epistemic benefit can be substantial. Contemporary computing environments often optimize aggressively for compact state, hidden caches, automatic replacement, and seamless user experience. Experimental work frequently benefits from the opposite tendency. Intermediate states should be retained precisely because one does not yet know which distinctions will matter. A file that appears redundant during successful execution can become decisive when diagnosing a failure.

The repository's `icepick.sh` procedure extends this principle one level further by concatenating the textual apparatus and its outputs into a single inspectable snapshot. Such a snapshot is not required for execution. The pipeline works without it. Its purpose is epistemic portability. Scripts, prompt templates, generated outlines, drafts, reviews, final essays, and explanatory documentation can be gathered into one textual object that can itself be inspected, archived, compared, or supplied to another analytical system. The experiment thereby creates a representation of its own representational environment.

There is an interesting recursion here. The essay pipeline externalizes model states into files. The repository organizes those files into an experimental history. The aggregation script then externalizes the repository's textual state into another artifact. Finally, the present essay is constructed by examining that aggregate artifact and interpreting the architecture it records. At each level, a process becomes available for higher-order inspection because some representation of the lower-order process persists.

This recursion should not be mistaken for perfect self-description. The aggregate file does not contain the internal activations of the Granite models, the complete runtime state of Ollama, the operating system scheduler, or every environmental variable. No finite experimental record contains its entire causal history. The relevant question is instead whether the record captures the variables necessary for the analysis being attempted. For studying the division between outlining, drafting, review, and revision, the textual artifacts and scripts provide a useful observational boundary.

Version control adds a further dimension. Filesystem persistence distinguishes simultaneous artifacts within a run; Git can distinguish successive states of the apparatus across time. A prompt template can change. A shell script can acquire another stage. Model assignments can be modified. A bug can be corrected. Without version history, later outputs may be compared even though they were generated by subtly different apparatuses. Git makes changes to the experimental instrument themselves explicit.

This matters because a software-mediated experiment has two classes of evolving object. The specimens evolve, and the apparatus evolves. In the present case, the specimens are essays and their intermediate states. The apparatus consists of scripts, prompts, model assignments, and conventions for persistence. If the apparatus changes between runs, the resulting essays are not strictly products of the same procedure. Version control provides a mechanism for identifying such changes and, when necessary, checking out the precise procedural state associated with an earlier experiment.

Git therefore functions as more than a publication mechanism. Indeed, nothing about the local experiment requires a remote repository. Its epistemically relevant feature is the ability to preserve a monotonic sequence of repository states. A commit establishes a named historical boundary after which a particular collection of files can be recovered. This makes procedural evolution inspectable in approximately the same way that preserving `draft.md` makes textual evolution inspectable.

The word "monotonic" requires qualification. Git history is not monotonic in the sense that file contents can only grow. Files can be deleted, rewritten, or reverted. What is monotonic is the accumulation of committed historical states along a chosen ancestry: once a commit has been created and retained, later commits can add new states without requiring the earlier state to disappear. The repository can therefore preserve destructive transformations while maintaining access to their predecessors. Historical information grows even when current files shrink.

This property is especially valuable for generative systems because generation is cheap enough to encourage replacement. A model can rewrite an entire essay in seconds. Without provenance, such cheap replacement makes it easy to lose track of which conceptual gains belonged to which stage. Versioned persistence imposes a countervailing discipline. The cost of generating alternatives remains low, but the history of alternatives need not be erased.

The architecture can consequently be described as a system of externalized continuation. No individual model invocation carries the whole process forward. Instead, each invocation leaves behind an artifact that alters what continuations are available to the next invocation. The outline changes the space of drafts that can plausibly follow. The draft changes the space of criticisms. The review changes the space of warranted revisions. The filesystem preserves these constraints between events.

This perspective differs from treating memory as a passive archive. An archive stores the past for possible later retrieval. The files in the pipeline are actively constitutive of future computation. The outline is read because the draft depends on it. The review is read because revision depends on it. Their persistence is therefore operational rather than merely documentary. The same artifact can simultaneously serve as state, evidence, and provenance.

The resulting architecture suggests a general principle for local language-model systems: when a task can be decomposed into transformations whose outputs have stable semantic roles, persistent external representations can substitute for a considerable amount of hidden orchestration. Instead of asking a model to remember everything, the system can decide what deserves to be remembered, serialize it, and reconstruct only the context required for the next operation. This does not eliminate the need for model context. It changes the relationship between context and memory. Context becomes a temporary working set assembled from a more durable external state.

The distinction is analogous to the relationship between processor registers and persistent storage, though again the analogy is functional rather than literal. A model's active context is immediately available during inference but disappears when the invocation ends. A file is slower and semantically inert by itself, but durable and addressable. The shell pipeline coordinates movement between these regimes. It loads selected persistent representations into transient model context, obtains a transformation, and writes the result back into persistent form.

Once described this way, the writing pipeline resembles a small state machine whose transition functions happen to be language models. Its continuity belongs to the architecture rather than to any individual inference. The apparent agent is therefore distributed across model, prompt, filesystem, and control program. This is not an argument that all of these components possess cognition independently. It is an argument about the appropriate boundary of analysis. If one wishes to explain why the final essay has the form it does, examining only the final 3B invocation is insufficient. The causal system includes the persisted outline generated earlier, the draft, the 8B review, the prompt templates that assign their roles, and the shell process that determines their order.

The external-memory interpretation also exposes a useful experimental possibility. Because each memory artifact can be independently modified, one can intervene on memory itself. A human can replace an outline while leaving the topic fixed. Two reviewers can evaluate the same frozen draft. The same review can be supplied to different revisers. A draft can be revised once with its review and once without it. An outline can be deliberately corrupted to measure how strongly the drafting stage depends upon it. These interventions would be difficult to interpret if the relevant states existed only implicitly inside a continuous conversation.

Persistent memory therefore converts hidden dependence into manipulable variables. This is one of the strongest motivations for the architecture. The system is not interesting merely because it can write essays locally. It is interesting because its cognitive decomposition can be perturbed experimentally.

4.1 Formalization: External Memory as a State-Preserving Channel

Let model invocations occur at discrete times

$$t_0, t_1, \ldots, t_n.$$

Suppose invocation I_k terminates after producing output x_k. Without an external persistence mechanism, assume its transient internal state z_k is unavailable to later invocation I_{k+1}:

$$z_k \notin \text{Inputs}(I_{k+1}).$$

Introduce a persistence operator

$$P : \mathcal{X} \to \mathcal{S},$$

where $\mathcal{X}$ is the space of generated textual representations and $\mathcal{S}$ is the space of externally addressable stored states. Let

$$s_k = P(x_k).$$

A retrieval operator

$$L : \mathcal{S} \to \mathcal{X}$$

loads the persisted representation into a later computational context. Ideally,

$$L(P(x)) = x.$$

For ordinary lossless text files this identity holds at the serialization level, modulo encoding and newline conventions. The next inference can therefore be written

$$x_{k+1} = M_{k+1}\left(p_{k+1}, L(s_{i_1}), \ldots, L(s_{i_m})\right),$$

where the indices $i_1, \ldots, i_m \leq k$ identify the prior artifacts selected as relevant context.

The system state before invocation $k+1$ is not the hidden state of the preceding model. It is the persistent set

$$S_k = \{s_0, s_1, \ldots, s_k\}.$$

A context-selection function

$$\Gamma_{k+1} : 2^{S_k} \to 2^{S_k}$$

determines which stored representations are loaded for the next stage. Thus

$$C_{k+1} = \Gamma_{k+1}(S_k).$$

For the drafting stage,

$$C_d = \{T, O\}.$$

For the review stage,

$$C_r = \{D\}.$$

For the revision stage,

$$C_v = \{D, R\}.$$

The architecture therefore does not use complete historical accumulation

$$C_{k+1} = S_k$$

by default. It uses selective context reconstruction.

Definition 4.1 (Externalized memory)**.** *A persistent store S constitutes externalized memory for a computational process when there exists at least one later transition whose output depends upon information retrieved from S after the process that originally produced that information has terminated.*

Under this definition, the pipeline's filesystem is externalized memory because, for example,

$$D = \Omega_d(T, O)$$

depends upon O after the outlining invocation that produced O has terminated.

Proposition 4.2 (Persistence permits discontinuous computational continuation)**.** *Let I_a and I_b be computational processes with no shared transient state. If I_a produces x, x is losslessly persisted as $s = P(x)$, and I_b receives $L(s)$, then information produced by I_a can constrain I_b despite the absence of shared transient state.*

Proof. By lossless persistence,

$$L(P(x)) = x.$$

Therefore I_b can be represented as

$$y = I_b(L(P(x)), u) = I_b(x, u),$$

where u denotes its other inputs. Since y is a function of x, information generated by I_a affects

the output of I_b. This dependence exists even though no transient state of I_a survives into I_b. Hence persistent external state is sufficient to establish computational continuation across discontinuous processes. □ □

The proposition explains why continuity of the writing process does not require continuity of model execution.

We can now distinguish accumulated context from selected context. Let the total size of historical state after n stages be

$$H_n = \sum_{i=0}^{n} |s_i|,$$

where $|s_i|$ denotes the serialized size of artifact s_i. A conversational architecture that reinserts the entire history at every stage may require context proportional to

$$H_n.$$

The external-memory architecture instead incurs active context

$$K_{n+1} = \sum_{s_i \in \Gamma_{n+1}(S_n)} |s_i|.$$

Whenever

$$\Gamma_{n+1}(S_n) \subsetneq S_n,$$

we have

$$K_{n+1} \leq H_n,$$

with strict inequality whenever at least one nonempty historical artifact is excluded.

This does not prove that selective context is semantically superior. Excluded information may matter. It establishes only that persistence allows context size to be decoupled from total historical state.

The central design problem becomes selection. Let

$$R(s_i, \Omega_j)$$

denote the relevance of stored artifact s_i to downstream operator Ω_j. An ideal context selector would choose

$$\Gamma_j^* = \arg\max_{\Gamma \subseteq S} \left[\sum_{s_i \in \Gamma} R(s_i, \Omega_j) - \lambda \sum_{s_i \in \Gamma} |s_i| \right].$$

The first term rewards relevant retained information; the second penalizes context cost.

The implemented pipeline uses hand-designed approximations to this optimization. The review receives the draft because the draft is assumed to be the principal object relevant to diagnosis. The revision receives both draft and review because both the object and its diagnosis are assumed relevant to repair.

The retention of the original topic during drafting can also be formalized. Let

$$O = f(T)$$

be the generated outline. If O were a sufficient statistic for T with respect to drafting, then

$$P(D \mid T, O) = P(D \mid O).$$

Under that condition, supplying T in addition to O would add no information relevant to D.

The architecture does not assume this sufficiency. Instead it permits

$$I(D; T \mid O) > 0,$$

where I denotes conditional mutual information. In ordinary language, the topic may retain information relevant to the draft that the outline failed to preserve. Keeping both is therefore a hedge against lossy intermediate representation.

An analogous relation holds for review and draft during revision. If the review R were a sufficient statistic of draft D for constructing the repaired essay F, then

$$P(F \mid D, R) = P(F \mid R).$$

The architecture again refuses this assumption and retains

$$F = \Omega_v(D, R).$$

Proposition 4.3 (Safe retention under uncertain sufficiency)**.** *Suppose an intermediate representation $Y = f(X)$ may be lossy with respect to a downstream task Z, and the storage cost of retaining X is negligible relative to the expected loss caused by unavailable task-relevant information. Then retaining both X and Y weakly dominates irreversible replacement of X by Y.*

Proof. If Y is sufficient for Z, retaining X produces no necessary informational gain, but by assumption its storage cost is negligible. If Y is not sufficient, there exists task-relevant information in X not recoverable from Y, and retaining X preserves access to that information. Therefore retention incurs negligible cost in the sufficient case and positive expected informa-

tional benefit in at least some insufficient cases. Under the stated cost assumption, irreversible replacement cannot have greater expected utility. □ □

For small textual artifacts, this result provides a rationale for apparently redundant storage.

We can also formalize provenance. Let

$$a_i$$

be an artifact and define its provenance record as

$$\pi(a_i) = (\text{parents}(a_i), \Omega_i, M_i, P_i, \theta_i)\,,$$

where Ω_i is the transformation class, M_i the assigned model, P_i the prompt specification, and θ_i any additional runtime parameters.

Complete provenance for terminal artifact F is then the transitive closure

$$\Pi(F) = \bigcup_{a \in \text{Ancestors}(F)} \pi(a).$$

The current repository does not yet persist every element of θ_i, so its provenance is incomplete in the strongest reproducibility sense. It nevertheless preserves substantially more causal structure than endpoint-only generation.

Version control introduces a second state index. Let

$$S^{(g)}$$

denote the repository state at Git commit g. An experimental artifact is then more precisely represented as

$$a_i^{(g)},$$

meaning stage artifact i generated under apparatus state g.

If two experiments are produced under commits g_1 and g_2, procedural equivalence requires at minimum that the relevant apparatus components be equal:

$$\mathcal{A}^{(g_1)} = \mathcal{A}^{(g_2)},$$

where $\mathcal{A}$ includes scripts, prompts, and model assignments.

When

$$\mathcal{A}^{(g_1)} \neq \mathcal{A}^{(g_2)},$$

observed output differences cannot be attributed solely to stochastic model behavior or topic differences without accounting for apparatus drift.

This gives Git an experimental rather than merely administrative role.

Theorem 4.4 (Two-dimensional provenance)**.** *A versioned staged pipeline admits two independent partial orders: an intra-run dependency order over artifacts and an inter-version ancestry order over apparatus states.*

Proof. Within a run, define

$$a_i \preceq_A a_j$$

when a_i is an ancestor dependency of a_j. Because the pipeline graph is acyclic, $\preceq_A$ is a partial order.

Across Git history, define

$$g_i \preceq_G g_j$$

when commit g_i is an ancestor of commit g_j. Along ordinary non-rewritten repository history this relation is also a partial order.

The two relations operate over different sets: $\preceq_A$ orders artifacts within an execution, while $\preceq_G$ orders states of the apparatus across development. Therefore an experimental observation can be indexed jointly as

$$(a_i, g_j),$$

locating it both within its generative trajectory and within the historical state of the instrument that generated it. □ □

This two-dimensional structure is one of the less obvious consequences of combining language models, ordinary files, shell orchestration, and version control. A final essay does not merely have content. It has ancestors. Its ancestors were generated by an apparatus that itself has ancestors.

The resulting representation can be written

$$F = F\left(T, O, D, R; \mathcal{A}^{(g)}\right).$$

The semicolon emphasizes that the first group consists of states within the writing trajectory, whereas $\mathcal{A}^{(g)}$ specifies the historical state of the machinery governing their transformations.

The architecture therefore replaces the impoverished equation

$$\text{prompt} \rightarrow \text{essay}$$

with a provenance-bearing construction:

$$(T \to O \to D \to R \to F)\Big|_{\mathcal{A}(g)}.$$

Once the process is represented in this form, failure can be investigated along either dimension. One may move backward through the artifact trajectory to locate where an argumentative defect first appeared, or backward through apparatus history to determine whether a change in prompts or orchestration altered the behavior of the pipeline. External memory thus provides more than persistence. It supplies coordinates for diagnosis.

The next question is what those coordinates reveal in practice. The preserved experiments show that the review stage can identify defects that remain incompletely repaired, making it possible to distinguish successful diagnosis from successful intervention. The following section therefore turns from architecture to the experimental outputs themselves and examines review–revision divergence as evidence about the limits of iterative language-model repair.

5 Diagnosis Without Repair: What the Experimental Outputs Reveal

The architecture becomes scientifically interesting only when its intermediate artifacts are allowed to disagree. If every review merely praised the draft, or if every revision perfectly implemented every criticism, the pipeline would provide little evidence about the independence of its stages. The preserved experimental outputs instead reveal a more complicated relationship. The larger reviewing model can articulate defects that remain visible in the subsequent revision, and the smaller revising model can preserve substantial portions of a draft even after receiving explicit instructions about their weaknesses. This divergence is not an incidental inconvenience. It provides evidence that diagnosis and repair constitute distinct transformations with distinct failure conditions.

The importance of this observation is easiest to see by considering the alternative architecture. Suppose a model receives an essay and the instruction to improve it. It returns a new essay. Comparison of the two documents may reveal that some passages changed, but the reason for those changes remains ambiguous. A passage may have been altered because the model detected a substantive problem, because a different phrasing happened to receive greater probability during regeneration, because surrounding material changed, or because the instruction to improve created a general pressure toward textual novelty. Conversely, an unchanged passage does not reveal whether the model judged it adequate or simply failed to notice its defect. The transformation is underdetermined by its endpoints.

An explicit review inserts another observable variable into this relation. If the review identifies

a problem in the draft and the final essay still contains that problem, then the failure can no longer be described simply as a failure to notice. At least at the level of the externalized review artifact, the system represented the defect before attempting revision. The problem lies somewhere between representation of the diagnosis and successful transformation of the text. The experiment has therefore increased the resolution with which failure can be described.

The essay concerning continuation and trajectories demonstrates the diagnostic capacity of the larger model particularly clearly. The generated draft advances the general thesis that continuation should be attributed to trajectories rather than isolated states. This is a natural extension of a broader process-oriented ontology, but it contains an important ambiguity. A trajectory is a temporally ordered history; it need not be continuous in the mathematical sense. Discrete dynamical systems, hybrid systems, event-driven computations, phase transitions, and jump processes can all possess trajectories containing discontinuities. The reviewer notices that the draft moves too readily between trajectory language and continuity language, thereby risking the inference that persistence through time requires smooth temporal evolution.

This objection is valuable because it attacks the central machinery of the argument rather than a peripheral example. If trajectory and continuity are conflated, then the essay risks excluding precisely the kinds of systems its framework ought to accommodate. A process ontology that recognizes only smooth trajectories would be poorly suited to computation, discrete state transitions, punctuated biological change, or many stochastic physical processes. The review therefore performs conceptual discrimination: it separates two notions that the draft had allowed to become partially fused.

The same review also questions the breadth of the essay's cross-domain generalization. Moving from physics to biology, computation, and other domains can reveal structural analogies, but structural analogy does not by itself establish a universal ontology. A concept that organizes several examples successfully may be heuristically powerful without being metaphysically fundamental. By objecting to the strength of the universalizing language, the reviewer is effectively demanding a distinction between demonstration and extrapolation. The draft has shown that trajectory-based descriptions are useful in several cases; it has not thereby proven that all persistence in every relevant domain must be understood in exactly that way.

A third criticism concerns prediction. The draft suggests that trajectory-centered representations may improve explanatory or predictive power, but the reviewer observes that this benefit has not been demonstrated with sufficient specificity. Here again the problem is one of warrant. A conceptual framework can reorganize explanation without necessarily improving prediction. If predictive improvement is claimed, some account must be given of what variables become observable, what competing representation performs worse, or what previously inaccessible inference becomes possible. The reviewer is therefore distinguishing a plausible consequence

from an established consequence.

These criticisms illustrate a general property of effective review. The critic does not need to reject the essay's thesis in order to improve its epistemic structure. Indeed, the strongest review often preserves the central idea while restricting the paths by which that idea can legitimately be extended. The reviewer's function is not necessarily opposition. It is constraint. A successful critical stage reduces the space of admissible continuations by ruling out unsupported transitions, ambiguous identifications, and conclusions whose strength exceeds their premises.

The admissibility experiment reveals an even more important phenomenon because the subsequent revision does not obviously track the sophistication of the review. The draft argues that admissibility offers a richer conceptual framework than optimization for understanding persistent systems. The reviewer responds by requesting more precise definitions, measurable or operational criteria, stronger transitions between conceptual domains, clarification of hidden assumptions, and a conclusion whose scope is more carefully matched to the preceding argument. These are substantive requests. They identify places where a conceptually attractive thesis has not yet been converted into a sufficiently discriminating analytical framework.

Yet comparison between the persisted draft and final essay shows substantial continuity. Passages criticized at the diagnostic stage can remain nearly unchanged after revision. The important point is not that the final essay is therefore worthless, nor that every criticism made by the 8B model should automatically have been obeyed. The important point is that the architecture lets us observe a difference between *having a representation of a criticism* and *successfully incorporating that criticism into a repaired artifact*.

This difference can arise for several reasons. The revising model may fail to understand the practical implications of the review. A criticism such as "provide measurable criteria" identifies a deficiency without necessarily specifying a valid criterion that can be inserted into the essay. The reviewer may therefore have diagnosed the absence of something without supplying the knowledge required to construct it. The reviser may recognize the instruction but lack enough domain information to perform the requested repair. Alternatively, the revision prompt's preservation constraint may compete with the pressure to make substantial changes. A smaller model may resolve that conflict conservatively, retaining text because it cannot determine which modifications would improve it safely.

There is also a more fundamental possibility: some criticisms may be easier to state than to satisfy. It is relatively inexpensive to say that a concept requires operationalization. It is much harder to invent an operationalization that is mathematically coherent, empirically meaningful, and faithful to the original concept. Likewise, identifying that an argument generalizes too quickly across domains is easier than reconstructing the argument so that each domain-specific bridge is independently warranted. Critical language can therefore create an

illusion of completed intellectual work when it has actually only named the work that remains.

This asymmetry resembles a familiar phenomenon in mathematics. Recognizing that a proof contains a gap is not equivalent to possessing a proof of the theorem. A reviewer may identify the exact line at which an inference fails and still be unable to supply a lemma that closes the gap. The diagnosis can be entirely correct while repair remains impossible under the current assumptions. Indeed, sometimes the correct consequence of discovering a gap is not to repair the proof but to weaken the theorem. Textual argument behaves similarly. A critic can establish that a conclusion is insufficiently warranted without thereby establishing which stronger premises are true.

The pipeline therefore benefits from treating criticism as information rather than authority. The 8B review is evidence about the draft, not a privileged ground truth. A criticism can be correct, partially correct, irrelevant, or mistaken. The revision stage should not mechanically satisfy every sentence in `review.md`; doing so would merely transfer epistemic authority from the drafting model to the reviewing model. What is required is a repair decision that evaluates whether the criticism identifies a real defect and whether a proposed intervention improves upon leaving the relevant passage unchanged.

This is where the preservation instruction acquires greater significance. "Preserve claims and passages that survived criticism" can be interpreted as an attempt to prevent indiscriminate rewriting, but survival itself must be defined carefully. A passage is not necessarily validated merely because the reviewer failed to mention it. Silence may reflect limited attention. Conversely, a passage is not necessarily defective merely because the reviewer criticized it. Review creates a diagnostic partition, but that partition is uncertain.

The architecture can therefore be understood as operating with three imperfect epistemic objects. The draft is a candidate argument. The review is a candidate diagnosis of that argument. The revision is a candidate repair conditioned on both. None has absolute authority over the others. Their relation is evidential. A human investigator gains information by comparing them precisely because disagreement remains possible.

This makes the preserved outputs more valuable than a simple ranking of final essays. Suppose the final essay receives only a modest improvement in some external quality score. Without intermediate artifacts, one might conclude that the larger reviewer contributed little. But if the review correctly identified several serious defects that the smaller reviser failed to address, the appropriate conclusion is different. The review stage may have succeeded while the repair stage became the bottleneck. The next architectural intervention should then target revision rather than review.

This suggests an immediate experimental permutation. Instead of using the 3B model for

revision, one could preserve the same 3B draft and 8B review while assigning the 8B model to revision. The comparison would isolate whether the observed review–revision gap results primarily from limited repair capacity in the smaller model. Conversely, one could use the 3B model for both draft and review while holding the final reviser fixed, testing whether the larger critic contributes diagnostic information unavailable from the smaller one. Because the existing artifacts are persistent, some of these experiments can reuse frozen drafts rather than regenerating the entire pipeline.

The ability to freeze an intermediate state is methodologically important. If the draft were regenerated every time the reviewer changed, then differences in final essays would combine variation in the draft with variation in review. By holding `draft.md` constant and changing only the reviewer, one can perform an intervention closer to a controlled experiment. Similarly, a single review can be supplied to multiple revisers. The filesystem thus permits counterfactual branching from a shared historical state.

The distinction between branching and sequential iteration deserves emphasis. A naive iterative system repeatedly transforms

$$D_0 \to R_0 \to D_1 \to R_1 \to D_2 \to \cdots$$

and observes only the resulting path. A branching experiment can instead hold D_0 fixed and construct several reviews

$$R_0^{(1)}, R_0^{(2)}, \ldots, R_0^{(k)},$$

then hold a selected review fixed and construct several revisions. Branching allows the contribution of individual transformations to be compared without conflating them with changes elsewhere in the trajectory.

This is one reason the current small collection of essays should be understood as exploratory rather than confirmatory. The experiments reveal phenomena worth measuring, particularly the apparent asymmetry between diagnostic sophistication and successful revision, but they do not yet establish effect sizes. The topics differ, the outputs are stochastic, and there is no independent annotation of all substantive defects. Nevertheless, exploratory experiments are valuable when they expose variables that a later controlled design should isolate. Here the principal newly visible variables are diagnostic recall, diagnostic precision, repair uptake, collateral damage, preservation, and net argumentative improvement.

The technological-diffusion essay adds another dimension to this analysis because its subject concerns the diffusion of competence itself. The pipeline is in one sense an experiment in the distribution of competence across computational components. The smaller model does not need to contain every capability required by the total process if another component

can externalize guidance that constrains its subsequent behavior. The review becomes a transferable representation of competence: the stronger model expends computation to identify weaknesses, then stores those judgments in a form the cheaper model can consume.

Whether this transfer succeeds is precisely what the review–revision divergence tests. If the 8B model can recognize a defect but the 3B model cannot use the resulting description to repair it, then competence has not been fully transmitted merely because information was transmitted. The distinction parallels a broader difference between receiving instructions and acquiring capability. A system may possess a textual description of what should be done without possessing the representational or computational resources required to do it.

This observation complicates simplistic accounts of model collaboration. Communication among models does not automatically pool their capabilities. An output channel can transmit propositions, recommendations, examples, or constraints, but it cannot necessarily transmit the latent competence that produced them. A strong critic may know that a proof needs a new lemma without being able to encode all of the mathematical search required to discover that lemma in a short review. A weaker reviser receives the diagnosis but not the critic's internal representational capacity.

The pipeline thus provides a small experimental setting for studying capability transmission through language. The review is a natural-language channel between models. Its effectiveness depends not only on the correctness of its content but on whether that content is sufficient to induce the downstream transformation. This suggests that reviews themselves could eventually be evaluated not only for diagnostic accuracy but for *repairability*: how effectively can another model convert a review into a better artifact?

A review optimized for human readability may not be optimal for machine-mediated repair. A general statement such as "the notion of admissibility needs operational criteria" may be intellectually correct but operationally weak. A more repairable review might identify the exact passage, state the missing distinction, propose a candidate formal criterion, identify dependencies elsewhere in the essay, and specify what should remain unchanged. The review could therefore evolve from commentary into a structured repair specification.

The current experiment intentionally does not go that far. Its natural-language review preserves flexibility and makes the outputs easy for a human investigator to read. But the observed gap between criticism and revision supplies a reason to explore richer representations later. The next generation of the pipeline might distinguish a prose review intended for human inspection from a machine-oriented repair plan intended to guide the reviser.

The central result of the present experiments is consequently methodological rather than evaluative. The pipeline has already succeeded at something even where revision fails: it

has made the failure decomposable. A weak final essay no longer forces the conclusion that "the model could not do it." One can ask which model, performing which role, on which representation, failed to perform which transformation. That is a substantial increase in explanatory resolution.

5.1 Formalization: Review–Revision Divergence and Repair Uptake

Let a draft D contain an ideal defect set

$$E(D) = \{e_1, e_2, \ldots, e_n\}.$$

Let a review R identify the proposed defect set

$$\widehat{E}(R) = \{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_m\}.$$

Define the correctly diagnosed set

$$C(D, R) = E(D) \cap \widehat{E}(R),$$

where intersection is understood semantically rather than as literal string identity.

The false-positive diagnostic set is

$$F^{+}(D, R) = \widehat{E}(R) \setminus E(D),$$

and the missed-defect set is

$$F^{-}(D, R) = E(D) \setminus \widehat{E}(R).$$

Diagnostic precision and recall are therefore

$$P_R = \frac{|C(D, R)|}{|C(D, R)| + |F^{+}(D, R)|}$$

and

$$R_R = \frac{|C(D, R)|}{|C(D, R)| + |F^{-}(D, R)|}.$$

Now let the revision operator produce

$$D' = \rho(D, R).$$

For each correctly diagnosed defect $e_i \in C(D, R)$, define a repair indicator

$$u_i = \begin{cases} 1, & e_i \notin E(D'), \\ 0, & e_i \in E(D'). \end{cases}$$

The review uptake rate is

$$U(D, R, D') = \frac{\sum_{e_i \in C(D,R)} u_i}{|C(D, R)|}.$$

A system can therefore satisfy

$$P_R \approx 1, \qquad R_R \approx 1,$$

while simultaneously having

$$U \approx 0.$$

This is the formal signature of diagnosis without repair.

Theorem 5.1 (Information availability is insufficient for successful repair)**.** *Let R contain a correct diagnosis of every defect in D. The availability of R as an input to revision is not sufficient to guarantee the existence of a revision operator ρ_M, realizable by a particular model M, such that*

$$E(\rho_M(D, R)) = \varnothing.$$

Proof. Let $e \in E(D)$ be a defect whose correction requires construction of some representation z outside the effective generative capability of M. Assume the review correctly states that e exists but does not itself contain a complete replacement representation z. Because M cannot construct z, every realizable output

$$D' = \rho_M(D, R)$$

either retains e, replaces it with another defect, or removes the affected claim entirely. If removal is not an admissible repair because the claim is necessary to the essay's thesis, then no defect-free D' is realizable by M under the available representation. Hence correct diagnostic information does not imply repairability by the downstream model. □ □

This theorem formalizes the difference between propositional transfer and capability transfer. The review can transmit the proposition

"passage p is defective"

without transmitting the competence required to generate a valid replacement for p.

Let

$$K_M$$

denote the set of transformations realizable by model M under the relevant context and inference configuration. Let

$$\mathcal{A}(D, R)$$

denote the set of admissible repaired essays relative to draft and review. Successful repair is possible only if

$$K_M(D, R) \cap \mathcal{A}(D, R) \neq \varnothing.$$

A stronger review can shrink uncertainty about

$$\mathcal{A}(D, R),$$

but it does not necessarily enlarge

$$K_M(D, R).$$

This distinction is fundamental. Better information about the target region does not necessarily expand the reviser's reachable region.

The relationship can be represented geometrically. Let $\mathcal{X}$ be the space of possible essays. The draft occupies point

$$D \in \mathcal{X}.$$

The reviewer induces an estimated admissible region

$$\widehat{\mathcal{A}}_R \subseteq \mathcal{X}.$$

The revising model has a reachable set

$$\mathcal{K}_M(D, R) \subseteq \mathcal{X}.$$

Successful review-guided repair requires

$$\mathcal{K}_M(D, R) \cap \widehat{\mathcal{A}}_R \neq \varnothing,$$

and genuine successful repair additionally requires that the selected point lie within the true admissible region

$$\mathcal{A} \subseteq \mathcal{X}.$$

Thus the actual requirement is

$$D' \in \mathcal{K}_M(D, R) \cap \widehat{\mathcal{A}}_R \cap \mathcal{A}.$$

A perfect reviewer may improve the approximation

$$\widehat{\mathcal{A}}_R \approx \mathcal{A}$$

without changing the fact that

$$\mathcal{K}_M(D, R) \cap \mathcal{A} = \varnothing.$$

This gives a precise interpretation to the possibility that the 8B model can diagnose a problem that the 3B reviser cannot adequately repair.

The same formalism permits a distinction between textual change and repair. Let

$$d_{\text{text}}(D, D')$$

measure textual distance between draft and revision. Large distance does not imply large repair:

$$d_{\text{text}}(D, D') \gg 0 \Rightarrow U(D, R, D') \gg 0.$$

Likewise, small textual distance does not imply failure if the relevant defect can be corrected locally:

$$d_{\text{text}}(D, D') \approx 0 \Rightarrow U(D, R, D') \approx 0.$$

The correct quantity concerns defect-sensitive change.

Let

$$\Lambda_R \subseteq D$$

denote the textual support implicated by review R, and let

$$\Lambda_{\bar{R}}$$

denote material not implicated by the review. Define

$$\Delta_R = d(D|_{\Lambda_R}, D'|_{\Lambda_R})$$

and

$$\Delta_{\bar{R}} = d(D|_{\Lambda_{\bar{R}}}, D'|_{\Lambda_{\bar{R}}}).$$

A crude conservative-repair efficiency can then be defined as

$$\eta_{\text{repair}} = \frac{\text{weighted defects removed}}{\epsilon + \Delta_R + \lambda \Delta_{\bar{R}}},$$

where

$$\lambda > 1$$

penalizes unnecessary modification outside the diagnosed repair surface and $\epsilon > 0$ prevents division by zero.

This quantity rewards interventions that remove substantive defects with limited collateral rewriting.

We can now state a preservation result.

Proposition 5.2 (Minimal repair under perfect localization)**.** *Assume that the review perfectly identifies the support of every defect in D, that defects can be repaired without modifying text outside that support, and that textual modification has positive cost. Then any optimal repair under a loss-plus-change objective leaves all text outside the diagnosed support unchanged.*

Proof. Let the objective be

$$J(D') = L(D') + \lambda d(D, D'), \qquad \lambda > 0.$$

By assumption, there exists a repair D^* eliminating all targeted defects while satisfying

$$D^*|_{\Lambda_R} = D|_{\Lambda_R}.$$

Consider another defect-equivalent repair $\widetilde{D}$ that also modifies some material outside Λ_R. Because both repairs have identical defect loss,

$$L(D^*) = L(\widetilde{D}),$$

while

$$d(D, D^*) < d(D, \widetilde{D}).$$

Therefore

$$J(D^*) < J(\widetilde{D}).$$

Hence an optimal repair does not modify undiagnosed material when all defects can be corrected without doing so. □ □

Real essays violate the proposition's assumptions because argumentative dependencies are

nonlocal. Repairing a definition can require changing every later passage that uses it. The proposition nevertheless provides a useful baseline: collateral rewriting requires justification.

The experimental design can now be extended through controlled branching. Freeze a draft D. Let two reviewers produce

$$R_s = C_{3B}(D)$$

and

$$R_\ell = C_{8B}(D).$$

Hold the reviser fixed:

$$D'_s = \rho_{3B}(D, R_s),$$

$$D'_\ell = \rho_{3B}(D, R_\ell).$$

Then the contrast

$$Q(D'_\ell) - Q(D'_s)$$

estimates the downstream value of the stronger review subject to the fixed reviser's ability to use it.

Conversely, hold the 8B review fixed and vary the reviser:

$$D'_3 = \rho_{3B}(D, R_\ell),$$

$$D'_8 = \rho_{8B}(D, R_\ell).$$

The contrast

$$Q(D'_8) - Q(D'_3)$$

estimates the marginal value of stronger repair capacity conditional on an identical diagnosis.

A complete 2×2 review–revision experiment would produce

$$D'_{ij} = \rho_{M_j}\left(D, C_{M_i}(D)\right),$$

where

$$i, j \in \{3B, 8B\}.$$

This yields four conditions:

$$D'_{3,3}, \qquad D'_{3,8}, \qquad D'_{8,3}, \qquad D'_{8,8}.$$

The main effect of reviewer capacity, the main effect of reviser capacity, and their interaction can then be investigated while the original draft remains fixed.

The interaction term is especially interesting:

$$I = \left[Q(D'_{8,8}) - Q(D'_{8,3})\right] - \left[Q(D'_{3,8}) - Q(D'_{3,3})\right].$$

If

$$I > 0,$$

then the stronger reviser benefits disproportionately from the stronger review, suggesting that some diagnostic information produced by the 8B critic is usable only when sufficient repair capacity is available downstream.

This possibility captures the central lesson of the exploratory outputs. Cognitive components cannot be evaluated solely by the quality of the information they emit. Their value depends on whether downstream components can transform that information into admissible continuation.

The essay pipeline therefore exposes a distinction that one-shot generation conceals:

$$\text{recognition} \neq \text{representation} \neq \text{repair} \neq \text{verified improvement}.$$

A model may recognize a defect internally but fail to express it. A review may represent the defect accurately but fail to specify a usable intervention. A reviser may perform the requested intervention but introduce greater collateral damage. A final essay may appear smoother while remaining conceptually weaker.

Persistent staging turns these possibilities into separate experimental hypotheses. This is the principal result of the first runs. Their importance lies not in proving that the 8B model is a universally better critic or that the 3B model is an optimal writer, but in demonstrating that a sufficiently decomposed writing process allows the location of competence and failure to become an object of investigation.

6 Planning Before Prose: The Outline as an Intermediate Representation

The first transformation in the pipeline deserves separate attention because it introduces structure before substantial prose exists. Given only a topic, the system does not immediately ask the drafting model to produce an essay. It first asks for an outline containing a conceptual problem, a central thesis, logically dependent sections, a serious objection or limitation, and the consequences of the proposed argument. This design reflects a hypothesis about writing

that is independent of language models: some errors are cheaper to prevent at the level of structure than to repair after they have been realized across several pages of prose.

An essay is not merely a sequence of individually acceptable sentences. Its argumentative identity depends upon relations distributed across the document. A definition introduced near the beginning may constrain a conclusion several sections later. An example may be relevant only because an earlier claim established the dimension along which the example should be interpreted. An objection may force qualification of the thesis rather than merely occupy an isolated paragraph. Consequently, local fluency is insufficient for global coherence. A model capable of producing plausible paragraphs can still generate an essay whose sections do not accumulate toward a common conclusion.

One-shot generation asks the model to solve structural and linguistic problems simultaneously. At each token, it must continue the local sentence while implicitly maintaining a representation of the global argumentative trajectory. Modern language models can often do this surprisingly well, but the arrangement provides little external control over the relation between local realization and global plan. Once several paragraphs have been generated, the existing prose itself becomes part of the conditioning context. The model is then constrained not only by the original topic but by its own accumulated commitments.

The outline introduces a lower-cost representational state before these commitments become expensive. A thesis can be changed in an outline by replacing a sentence. A missing objection can be inserted as a structural requirement before dozens of paragraphs depend upon the unqualified thesis. Two sections whose functions overlap can be merged before both have been elaborated. A concept that appears necessary in the conclusion can be moved earlier so that it is available when needed. Planning therefore shifts some forms of repair into a representation where their propagation cost is lower.

The outline should not, however, be understood as a miniature essay. The drafting prompt explicitly instructs the model to treat it as argumentative structure rather than text to be repeated mechanically. This distinction is essential. If the outline already determined every important sentence, drafting would become little more than expansion. Instead, the outline specifies relations among prospective regions of the essay while leaving the linguistic and explanatory realization underdetermined. It is an intermediate representation because it contains more structure than the topic and less commitment than the draft.

This position between topic and prose gives the outline a dual function. It is both a plan for generation and a diagnostic surface. Before the draft exists, the outline can be inspected for missing argumentative dependencies. After the draft exists, it can be used to determine whether the drafting model preserved the intended structure. A weak final essay may therefore be traced to at least two distinct sources. The plan may itself have been defective, or a defensible

plan may have been poorly realized.

The distinction matters because these failures demand different repairs. If the outline contains three sections that merely restate the thesis in different vocabulary, improving individual paragraphs will not solve the structural redundancy. The plan must change. Conversely, if the outline establishes a clear sequence but the draft introduces an unsupported inference while realizing one section, rebuilding the entire outline may be unnecessary. The repair should occur closer to the point at which the defect was introduced.

The experimental prompts make this structural role unusually explicit. The outlining model is instructed to construct one central argument rather than a list of loosely connected observations. This addresses a common failure mode of generated essays: thematic association masquerading as argumentative development. A text may contain several relevant facts about a topic while lacking any reason that the facts must appear in their particular order. Such an essay possesses topical coherence but weak inferential structure.

A stronger conception of organization requires that sections transform the argumentative state. After a definition is established, later claims can rely upon it. After a distinction is introduced, later examples can be classified by it. After an objection is considered, the thesis may become narrower but better warranted. Each section should therefore change what can legitimately be asserted in the next section. The outline is useful precisely because these transitions can be represented before they are obscured by rhetorical detail.

This suggests that the natural unit of an outline is not a heading but a transformation. A heading such as "Applications to Biology" says where the essay will speak but not what argumentative work the section performs. A stronger outline would specify that the biological case tests whether a criterion developed earlier remains meaningful when the system contains adaptive regulation, reproduction, and environmental coupling. The difference is between a taxonomy of topics and a dependency graph of claims.

The generated outline is still produced by a language model and therefore cannot be assumed correct. Indeed, moving planning upstream merely relocates some opportunities for error. The model may produce a beautifully ordered outline around a false distinction. It may invent an objection that is easy to answer while ignoring a more serious one. It may establish a thesis whose apparent coherence results from ambiguity in a central term. Structural planning reduces some forms of accidental drift, but it can also stabilize early mistakes by propagating them deliberately through the draft.

For this reason the original topic remains available during drafting. The architecture does not allow the outline to become the sole authoritative representation of the task. This redundancy creates a weak form of error correction. If the outline drifts away from the topic, the drafting

model still has access to the original formulation. The topic and outline become two partially overlapping constraints rather than a simple chain in which each new representation replaces its predecessor.

A more developed architecture could introduce review at the planning stage itself. The present pipeline uses the 3B model to produce an outline and proceeds directly to drafting. Yet if structural mistakes are indeed cheaper to repair before prose generation, then the marginal value of review might be especially high at this point. An 8B outline critic could evaluate whether the thesis is actually discriminating, whether each section has a necessary function, whether the objection threatens the strongest version of the thesis, and whether the proposed conclusion follows from the planned development. Such a modification would create a planning-repair loop before the more expensive drafting operation begins.

The current design avoids this additional complexity because its purpose is exploratory. The first experiments establish a minimal decomposition in which planning exists as an independent stage. Once this boundary has proved useful, more elaborate interventions become experimentally meaningful. Without a persisted outline, there would be no stable object upon which an outline reviewer could operate.

The outline also provides an opportunity to distinguish semantic expansion from argumentative expansion. During drafting, word count necessarily increases dramatically. But the increase in textual volume should not be confused with an increase in argumentative content. A model can expand a short outline into a long essay by paraphrasing each point repeatedly, adding generic examples, and supplying rhetorical transitions without introducing new inferential structure. Such a draft is longer without being conceptually deeper.

A useful drafting transformation should instead preserve the dependency structure of the outline while enriching the representations carried along its edges. If the outline states that optimization presupposes a stable objective while admissibility can accommodate changing constraints, the draft should do more than restate this contrast in several ways. It should explain what counts as a stable objective, show why persistence may alter the relevant constraints, specify how admissibility differs formally, and examine cases in which the distinction matters. Prose should unpack the inferential content of the plan.

This gives the outline a role analogous to an interface contract. It does not specify every implementation detail, but it establishes expectations that the implementation should satisfy. The draft can then be evaluated not merely for whether it sounds coherent but for whether it realizes the promised argumentative operations. If an outline promises an objection and the draft supplies only a weak rhetorical concession, that discrepancy is observable.

The relationship between outline and draft also reveals why intermediate representations can

improve interpretability without revealing neural internals. We cannot inspect the latent plan, if any, that the 3B model internally forms during generation. We can, however, require it to externalize a plan before drafting and then compare the draft with that external representation. The outline is not guaranteed to be identical to whatever internal structure guides generation, but it creates a public commitment against which later behavior can be assessed.

This notion of public commitment is important. Once a thesis and sequence have been written to `outline.md`, they become constraints that survive beyond the inference that generated them. The drafting model may still depart from them, but departure is now measurable. Without the outline, structural drift is difficult to distinguish from an alternative plan that was supposedly intended all along.

The same logic appears in scientific and engineering practice. Experimental protocols, preregistrations, software specifications, proof sketches, architectural diagrams, and research proposals all externalize intended structure before final realization. Their value is not that reality must obey the plan exactly. Their value is that deviations become legible. A plan converts some implicit expectations into inspectable commitments.

In the essay pipeline, this commitment is deliberately lightweight. The outline remains natural language rather than a formal schema. This preserves the generative flexibility of the drafting model, but it also limits automatic verification. A program cannot straightforwardly determine whether a paragraph genuinely satisfies a planned argumentative function merely by matching headings. Human or model-based semantic evaluation remains necessary. The representation is structured enough to guide reasoning but not structured enough to make conformance mechanically decidable.

This intermediate position is arguably appropriate for exploratory conceptual writing. Excessive formalization at the planning stage could force arguments into categories chosen before their content is understood. Too little formalization collapses planning back into unconstrained generation. The outline provides a provisional scaffold whose authority is real but revisable.

The batch experiments further demonstrate why such a scaffold matters. The topics concern distinct conceptual relations: repair and difference, admissibility and optimization, continuation and trajectories, and technological capability and diffusion. A generic essay generator could respond to all four with similarly shaped expository prose. Requiring an argumentative outline creates pressure to identify the particular inferential problem posed by each relation. The resulting essays can then be compared not only by style but by how their structures differ in response to different theoretical demands.

Planning is therefore not simply a productivity technique within the experiment. It is a method for making global structure persistent enough to become an object of analysis. The

outline converts the future essay from an unconstrained space of possible continuations into a constrained but still open region. Drafting then explores that region. Review evaluates the resulting trajectory. Revision attempts to return departures from admissibility toward a better-supported path.

6.1 Formalization: Outlining as Constraint Formation

Let $\mathcal{E}$ denote the space of possible essays for topic T. Before planning, the topic induces a broad feasible set

$$\mathcal{E}(T) = \{E : E \text{ is semantically responsive to } T\}.$$

An outline O introduces additional constraints

$$C_O = \{c_1, c_2, \ldots, c_m\}.$$

The outline-constrained essay space is

$$\mathcal{E}(T, O) = \{E \in \mathcal{E}(T) : E \models c_i \text{ for relevant } c_i \in C_O\}.$$

Ideally,

$$\mathcal{E}(T, O) \subset \mathcal{E}(T).$$

The purpose of outlining is therefore not to identify a unique essay but to reduce the admissible search region.

Let

$$|\mathcal{E}(T)|$$

represent, abstractly, the effective number of materially distinct argumentative realizations available under topic T. Define the structural reduction induced by outline O as

$$\Delta_O = \log |\mathcal{E}(T)| - \log |\mathcal{E}(T, O)|.$$

A useful outline should satisfy

$$\Delta_O > 0$$

without collapsing the feasible region to an overdetermined or defective trajectory.

The quality of the outline therefore cannot be measured by constraint strength alone. Let

$$A(E)$$

denote argumentative admissibility. Define the admissible subset

$$\mathcal{E}_A(T) = \{E \in \mathcal{E}(T) : A(E) = 1\}.$$

A good outline should preferentially remove inadmissible possibilities while preserving admissible ones. Define its filtering precision as

$$\phi(O) = \frac{|\mathcal{E}(T,O) \cap \mathcal{E}_A(T)|}{|\mathcal{E}(T,O)|}.$$

An outline that imposes many arbitrary constraints may produce a large Δ_O while reducing $\phi(O)$. Structural specificity is therefore not equivalent to structural quality.

Definition 6.1 (Admissibility-preserving outline)**.** *An outline O is admissibility-preserving relative to topic T if*

$$\mathcal{E}(T,O) \cap \mathcal{E}_A(T) \neq \varnothing.$$

This is a minimal requirement. An outline that excludes every admissible realization has constrained the problem too strongly or in the wrong direction.

The outline can also be represented as a directed dependency graph

$$G_O = (V_O, E_O),$$

where vertices correspond to argumentative commitments and

$$(v_i, v_j) \in E_O$$

means that the argumentative role of v_j depends upon v_i.

For example, a simplified structure may contain

$$v_1 = \text{definition},$$

$$v_2 = \text{distinction},$$

$$v_3 = \text{application},$$

$$v_4 = \text{objection},$$

$$v_5 = \text{qualified conclusion},$$

with dependencies

$$v_1 \to v_2,$$

$$v_2 \to v_3,$$

$$v_2 \to v_4,$$

and

$$(v_3, v_4) \to v_5.$$

An essay that merely contains prose corresponding to every vertex does not necessarily realize the graph. It must preserve the dependency relations.

Let the realized draft graph be

$$G_D = (V_D, E_D).$$

Define a structure-preservation mapping

$$f : V_O \to 2^{V_D},$$

where $f(v_i)$ identifies the draft passages intended to realize outline node v_i.

A dependency

$$(v_i, v_j) \in E_O$$

is preserved when the realization of v_j actually depends argumentatively upon the realization of v_i. Let

$$\chi_{ij} = \begin{cases} 1, & \text{dependency preserved,} \\ 0, & \text{otherwise.} \end{cases}$$

Then structural fidelity can be approximated as

$$F_{\text{struct}} = \frac{1}{|E_O|} \sum_{(v_i, v_j) \in E_O} \chi_{ij}.$$

This separates structural realization from surface similarity. A draft can use completely different wording from the outline while achieving

$$F_{\text{struct}} \approx 1.$$

Conversely, it can repeat every heading and key phrase while having low structural fidelity if the inferential relations are absent.

The advantage of early structural repair can now be formalized through propagation cost. Suppose defect e is introduced at stage k and affects a set of downstream commitments

$$\mathrm{Desc}(e).$$

Let the cost of repairing e after n dependent realizations have been generated be

$$C_r(e, n) = c_0(e) + \sum_{j=1}^{n} c_j(e),$$

where $c_0(e)$ is the cost of correcting the originating representation and $c_j(e) \geq 0$ is the cost of updating downstream dependency j.

Then

$$C_r(e, n+1) \geq C_r(e, n).$$

Proposition 6.2 (Nondecreasing downstream repair cost)**.** *If every downstream commitment dependent upon a defective representation requires nonnegative effort either to verify or modify after that representation changes, then expected repair cost is nondecreasing with the number of realized dependencies.*

Proof. By definition,

$$C_r(e, n+1) = C_r(e, n) + c_{n+1}(e).$$

Since

$$c_{n+1}(e) \geq 0,$$

it follows that

$$C_r(e, n+1) \geq C_r(e, n). \quad \square$$

$\square$

The proposition gives a formal reason to inspect structural defects before full prose realization. It does not claim that every outline defect is easy to detect early. It states that, conditional on detecting the same defect, repairing it before dependent prose accumulates weakly reduces the amount of downstream material that must be reconsidered.

The relationship between topic and outline can also be treated information-theoretically. Let T be the original topic and O its outline. If outlining is a lossy transformation,

$$H(T \mid O) > 0,$$

where H denotes conditional entropy. Some information present in the topic is not recoverable from the outline alone.

Drafting from outline only would produce

$$D \sim P(D \mid O).$$

The implemented architecture instead permits

$$D \sim P(D \mid T, O).$$

The information contributed by retaining the topic is

$$I(D; T \mid O) = H(D \mid O) - H(D \mid T, O).$$

Whenever

$$I(D; T \mid O) > 0,$$

the topic contains information relevant to drafting that is not captured by the outline. Retaining both therefore prevents the outline from becoming an irreversible bottleneck.

The same framework allows structural drift to be defined. Let

$$\Psi(O)$$

be a representation of the principal argumentative commitments encoded by the outline, and

$$\Psi(D)$$

the commitments realized by the draft. Define drift

$$\delta_{OD} = d_{\Psi}(\Psi(O), \Psi(D)),$$

where d_{Ψ} is a semantic distance measure.

Some drift is desirable because drafting necessarily elaborates and may legitimately refine the plan. Therefore the objective is not

$$\delta_{OD} = 0.$$

Instead, let

$$\delta^*$$

represent a region of admissible elaboration. Structural failure occurs when

$$\delta_{OD} > \delta^*$$

without corresponding evidence that the outline itself required revision.

This distinction suggests two possible repair directions:

$$O \to O'$$

when the plan is defective, or

$$D \to D'$$

when realization departed from a defensible plan.

Theorem 6.3 (Planning separates structural and realization error under identifiable contracts)**.** *Assume that an outline O has an independently evaluable structural contract A_O, and a draft D has a realization contract $A_D(D \mid O)$. Then failure of the final draft can, in principle, be decomposed into planning failure, realization failure, or both.*

Proof. There are four possible truth assignments:

$$(A_O, A_D) \in \{(1,1), (1,0), (0,1), (0,0)\}.$$

If

$$(A_O, A_D) = (1,0),$$

the plan satisfies its structural contract while the draft fails to realize it, yielding realization failure.

If

$$(A_O, A_D) = (0,1),$$

the draft faithfully realizes a structurally defective plan, yielding planning failure.

If

$$(A_O, A_D) = (0,0),$$

both failures occur.

Only

$$(A_O, A_D) = (1,1)$$

satisfies both conditions. Therefore explicit intermediate planning permits the two failure classes to be represented separately. □ □

Without O, the variable A_O is not externally available and this decomposition cannot be performed directly.

Finally, the potential value of reviewing before drafting can be derived. Let

$$p_e$$

be the probability that an outline contains a structural defect, let

$$p_d$$

be the probability that an outline reviewer detects it, and let

$$C_o$$

be the expected cost of repairing the defect at outline time. Let

$$C_f$$

be the expected cost after full drafting, with

$$C_f > C_o.$$

Ignoring review cost momentarily, expected avoided repair cost is

$$V = p_e p_d (C_f - C_o).$$

If the computational and temporal cost of outline review is

$$C_{\text{review}},$$

then adding the review stage has positive expected value whenever

$$p_e p_d (C_f - C_o) > C_{\text{review}}.$$

This inequality provides a concrete criterion for extending the current architecture. The additional stage is not justified merely because more review sounds desirable. It is justified when the expected value of detecting structural errors before their downstream realization exceeds the cost of performing the review.

The present pipeline has not yet measured these quantities. Its contribution is to make

them definable. By materializing the outline as an independent representation, it creates the experimental surface upon which structural diagnosis, early repair, fidelity measurement, and model-allocation experiments can subsequently operate.

The outline is therefore neither disposable scaffolding nor a miniature final product. It is a deliberately incomplete representation positioned where global commitments can be manipulated before they become deeply entangled with local prose. In a staged language-model architecture, that incompleteness is a feature: it preserves a region in which the argument can still change cheaply.

7 Prompt Templates as Experimental Protocols

If the filesystem provides the persistent state of the essay-writing system, the prompt templates determine the local laws under which that state is transformed. They occupy an intermediate position between ordinary prose instructions and executable program logic. A shell function can determine that the review stage follows the drafting stage, but it cannot by itself determine what intellectual operation "review" means. The model receives a string of text and generates another string. The prompt establishes the semantic contract that makes one invocation an outlining operation, another a drafting operation, another a critical evaluation, and another a revision. In this sense, prompts are not merely requests issued to a language model. Within a staged experimental architecture, they are components of the experimental apparatus.

This interpretation differs from the common treatment of prompting as a craft of finding particularly effective wording. Prompt engineering is often discussed as though the objective were to discover a formulation that causes a model to produce the best possible answer. The present pipeline has a different requirement. A useful prompt must certainly elicit competent behavior, but it must also define a sufficiently stable transformation that the resulting stage can be compared across topics, models, and runs. The question is therefore not only whether a prompt "works." It is whether it establishes an intelligible operation whose effects can be distinguished from those of neighboring operations.

The separation of the instructions into `outline.txt`, `draft.txt`, `review.txt`, and `revise.txt` is important for this reason. If the instructions were embedded throughout the shell script, procedural control and semantic control would be mixed. Altering a sentence in the review instructions would require editing the orchestration code, and a later investigator would have to inspect executable shell logic to determine which intellectual expectations changed. External templates make the semantic protocol independently versionable. A Git diff can show that the reviewer was newly instructed to distinguish substantive defects from stylistic preferences without implying that the execution order, model assignment, or filesystem layout

also changed.

The prompt renderer reinforces this separation by treating the templates as parameterized documents. Placeholders such as `{{TOPIC}}`, `{{OUTLINE}}`, `{{ESSAY}}`, and `{{REVIEW}}` distinguish stable instructions from variable experimental material. The template defines what operation should be performed; the substituted files define the object upon which the operation is performed. This is analogous to the distinction between a function and its arguments. The same review protocol can be applied to several drafts without rewriting the instructions, and the same draft can be reviewed under alternative protocols without changing the draft itself.

The importance of this distinction becomes apparent when considering reproducibility. Suppose two reviews differ. If they were generated from different drafts, different models, and different prompts, the cause of the difference is heavily confounded. If the draft and model are held fixed while only `review.txt` changes, the experiment isolates the effect of the review protocol more effectively. Conversely, if the prompt is held fixed while the reviewer changes from the 3B to the 8B model, differences can be interpreted as evidence about model behavior under a common role specification. Externalized prompts therefore create another family of experimentally manipulable variables.

The outlining prompt demonstrates how a protocol can establish expectations without determining content. It asks for a conceptual problem and central thesis, but it does not prescribe what that thesis must be. It requests logically dependent sections, but it does not determine their number or titles in advance. It requests an objection or limitation, but it does not tell the model which objection to select. The prompt therefore defines a class of admissible outlines rather than a single expected output.

The drafting prompt performs a different kind of constraint. It supplies both topic and outline, asks for a substantial analytical essay, and warns against treating the outline as prose to be mechanically expanded. This last instruction is especially important because it distinguishes structural dependence from lexical dependence. The draft should inherit the argumentative organization of the outline while generating the explanatory material required to make that organization intelligible. A successful transformation is therefore neither independent generation nor literal expansion.

The review prompt imposes the sharpest role boundary. The model is asked to identify conceptual gaps, unsupported transitions, ambiguous definitions, hidden assumptions, unnecessary repetition, and conclusions that exceed their warrant. At the same time, it is told not to rewrite the essay. This negative instruction is as important as the positive instructions because it removes an otherwise easy continuation path. Given a weak paragraph, a generative model can often produce a better paragraph more readily than it can articulate a precise theory of why the original paragraph is weak. Prohibiting rewriting forces the output toward diagnosis.

Negative constraints of this sort are methodologically useful because model behavior is underdetermined by broad task labels. "Review this essay" could mean copyediting, summarization, stylistic commentary, grading, fact checking, adversarial criticism, or wholesale rewriting. A stage name in the shell script cannot resolve these possibilities. The prompt must define what counts as a legitimate response. The semantic type of `review.md` is therefore established partly through exclusions.

The revision prompt introduces another negative constraint: material that survived criticism should be preserved. Again, this is not merely a preference about prose style. Without such an instruction, "revise according to this review" can easily become a request for regeneration conditioned on the review. The preservation requirement changes the implied objective from unconstrained improvement to repair. The reviser is asked to treat the draft as an artifact possessing already successful structure, not merely as raw material from which another essay can be generated.

This distinction reveals that prompt design can encode an intervention philosophy. A prompt that says "rewrite this essay to make it better" defines a very different transformation from one that says, in effect, "identify the warranted repair surface, correct the diagnosed failures, and preserve everything else unless dependencies require further change." Both may produce fluent essays. Only the latter attempts to preserve causal continuity between the draft and revision.

The prompts also establish different epistemic attitudes toward the input. The drafting model treats the outline as guidance. The reviewing model treats the draft as an object whose claims must earn their warrant. The revising model treats the review as diagnostic evidence rather than as replacement prose. Thus identical textual material could elicit different behavior depending upon the protocol in which it is embedded. A paragraph placed inside the drafting context is something to continue; the same paragraph inside the review context is something to interrogate.

This role conditioning can be described as a change in the model's effective relation to the artifact. Although the underlying model parameters remain fixed during inference, the prompt modifies the distribution of continuations. The architecture exploits this property deliberately. It does not require separate neural systems for planning, generation, criticism, and revision. It creates functional differentiation by placing general-purpose models under different textual contracts.

The experiment consequently contains two kinds of specialization. Model specialization is coarse and architectural: the smaller model is assigned primarily generative work and the larger model critical work. Prompt specialization is finer: even the same model can be made to perform distinct transformations by changing its role definition and supplied artifacts. These

two mechanisms interact. A model's observed competence at review is not simply a property of its parameters; it is competence under a particular review protocol.

This point complicates claims such as "the 8B model is a better critic." What has actually been observed is that the 8B model, under the current `review.txt` instructions and when supplied with the generated drafts, produces criticism that appears capable of identifying certain substantive defects. A different review prompt could alter that result substantially. A prompt focused on grammar would not test the same capability. A prompt demanding adversarial rejection might increase false positives. A prompt requesting numerical scores might compress useful diagnostic information into poorly calibrated scalar judgments. The protocol and the model are therefore inseparable components of the observed behavior.

For this reason prompt revisions should be treated with the same caution as code revisions. Improving a prompt during an experiment can improve output while simultaneously invalidating comparisons with earlier runs. If the purpose is product quality, such adaptation is desirable. If the purpose is experimental comparison, it constitutes apparatus drift. The repository should therefore preserve prompt versions and associate outputs with the prompt state that produced them.

This need not make prompt development rigid. On the contrary, versioning makes exploratory prompt modification safer because earlier protocols remain recoverable. One can create a new review protocol, run it against frozen drafts, and compare its diagnoses with those generated by the previous protocol. If the new version appears superior, that superiority can be examined rather than merely assumed because the latest output feels better.

The natural-language form of the prompts also makes them accessible to direct human criticism. An experimental protocol written in code may conceal its assumptions behind implementation details. A prompt states many of its assumptions explicitly. If the reviewer is told to prioritize unsupported transitions over stylistic preferences, the experiment has declared a theory of what matters in review. If the reviser is told to preserve material not implicated by criticism, the experiment has declared a theory of repair. These assumptions can themselves be debated.

There remains, however, a gap between the written protocol and the operation actually performed by the model. A shell function follows its programmed branch conditions with comparatively strict semantics. A language model interprets instructions probabilistically. It may partially ignore a prohibition, reinterpret an ambiguous phrase, or satisfy one requirement at the expense of another. The prompt is therefore not executable law in the strong sense. It is better understood as a normative protocol imposed upon a stochastic transformer.

This difference has experimental consequences. Compliance must itself become measurable. If a review prompt says not to rewrite the essay but the output contains long replacement

passages, the review may be semantically useful while violating its stage contract. If the revision prompt says to preserve uncriticized material but the final essay is almost entirely regenerated, the system may have produced a good essay while failing the intended repair experiment. Endpoint quality and protocol compliance are separate variables.

The current pipeline relies largely on human inspection to detect such deviations. A future version could introduce lightweight automatic checks. Review outputs could be tested for excessive textual overlap with a hypothetical rewritten essay, revisions could be compared with drafts to estimate the magnitude and localization of changes, and prompt hashes could be stored with every generated artifact. Such checks would not determine semantic correctness, but they would improve confidence that the intended experimental transformation actually occurred.

The prompt templates can therefore be understood as a boundary layer between deterministic orchestration and stochastic cognition. Outside the boundary, Bash determines that a particular file is read and another is written. Inside the boundary, a language model interprets natural-language constraints and generates a representation whose exact content cannot be specified in advance. The prompt mediates between these regimes. It converts procedural intent into textual conditions that a model can act upon.

This hybrid structure is one of the defining characteristics of language-model programming. Traditional programming specifies transformations primarily through executable rules. Pure conversational prompting specifies them primarily through natural language. The present system combines both: code determines topology, while prompts determine semantic operators. The result is neither conventional software nor unconstrained conversation. It is a computational graph whose nodes contain probabilistic natural-language programs.

The experimental value of this arrangement lies in its modularity. A semantic operator can be altered without changing the graph. The graph can be altered without rewriting every semantic operator. A model can be replaced without rewriting the protocol. An artifact can be frozen while both model and prompt are varied around it. These separations create the conditions for systematic investigation.

7.1 Formalization: Prompts as Parameterized Stochastic Operators

Let a language model M define a conditional distribution over output strings

$$P_M(y \mid x),$$

where x is the complete inference context.

Let a prompt template be a function

$$p_i : \mathcal{A}_i \to \Sigma^*,$$

where $\mathcal{A}_i$ is the product space of artifacts required by stage i, and Σ^* is the space of finite prompt strings.

For the review stage,

$$p_r : \mathcal{D} \to \Sigma^*,$$

while revision has

$$p_v : \mathcal{D} \times \mathcal{R} \to \Sigma^*.$$

The effective stage operator is therefore not simply the model M, but the composition

$$\Omega_{M,p_i}(a) \sim P_M\big(y \mid p_i(a)\big).$$

Thus changing the prompt while holding the model fixed changes the operator:

$$p_i \neq p_i' \quad \Longrightarrow \quad \Omega_{M,p_i} \not\equiv \Omega_{M,p_i'}$$

in general.

Likewise, changing the model while holding the protocol fixed gives

$$M_a \neq M_b \quad \Longrightarrow \quad \Omega_{M_a,p_i} \not\equiv \Omega_{M_b,p_i}.$$

Observed stage behavior is consequently a joint function

$$B_i = B(M, p_i, a, \theta),$$

where a is the input artifact and θ represents inference parameters and environmental conditions.

This immediately yields an experimental requirement.

Proposition 7.1 (Prompt control requirement)**.** *A comparison intended to estimate the effect of changing model M at a particular stage must hold the stage prompt fixed, unless prompt–model interaction is itself part of the quantity being studied.*

Proof. Suppose outputs

$$y_a \sim P_{M_a}(y \mid p_a(x))$$

and

$$y_b \sim P_{M_b}(y \mid p_b(x))$$

are compared with both

$$M_a \neq M_b$$

and

$$p_a \neq p_b.$$

Then an observed difference

$$\Delta Q = Q(y_b) - Q(y_a)$$

may arise from the model difference, the prompt difference, or an interaction between them. Without additional conditions these effects are not identifiable. Holding

$$p_a = p_b = p$$

removes prompt variation from the direct comparison, leaving model assignment as the manipulated variable, modulo stochastic and environmental variation. □ □

Similarly, prompt experiments should hold the model and artifact fixed.

Let

$$A_i(y) \in \{0, 1\}$$

denote whether output y complies with the semantic contract of stage i. The compliance probability is

$$C(M, p_i, a) = \Pr\left[A_i(Y) = 1 \mid Y \sim P_M(\cdot \mid p_i(a))\right].$$

Prompt quality cannot therefore be reduced to output quality. Let

$$Q(y)$$

measure the substantive quality of an output and define

$$U(p_i) = \mathbb{E}[Q(Y)] + \lambda \mathbb{E}[A_i(Y)],$$

with

$$\lambda > 0.$$

A protocol that occasionally produces excellent outputs by violating its assigned role may have high Q but low experimental utility.

For review, let the contract contain constraints

$$K_r = \{k_{\text{diagnose}}, k_{\text{no rewrite}}, k_{\text{substantive}}, k_{\text{prioritize}}\}.$$

Define

$$A_r(R) = \prod_{k \in K_r} a_k(R),$$

where

$$a_k(R) \in \{0, 1\}$$

indicates satisfaction of individual constraint k. Strict compliance requires

$$A_r(R) = 1.$$

A softer compliance score can be written

$$\widetilde{A}_r(R) = \sum_{k \in K_r} w_k a_k(R), \qquad \sum_k w_k = 1.$$

This allows partial adherence to be distinguished from complete role failure.

The negative instruction against rewriting can be modeled as a constraint on output type. Let

$$\mathcal{R}$$

be the space of diagnostic reviews and

$$\mathcal{W}$$

the space of replacement essays. The desired output distribution should concentrate probability on

$$\mathcal{R}$$

rather than $\mathcal{W}$.

For prompt p_r, define leakage

$$L_r = \Pr[Y \in \mathcal{W} \mid p_r(D)].$$

An effective negative constraint should reduce

$$L_r.$$

The prompt-design problem can therefore be represented as

$$p_r^* = \arg\max_p \left[\mathbb{E} Q_r(Y) - \lambda L_r(p) \right],$$

where Q_r evaluates diagnostic quality rather than essay quality.

This distinction is essential because the objectively best replacement essay would still be a failed review output if the experimental stage requires diagnosis.

Prompt versioning can also be formalized. Let

$$p_i^{(g)}$$

denote stage prompt i at repository state g. The stage operator becomes

$$\Omega_i^{(g)} = \Omega_{M_i^{(g)}, p_i^{(g)}}.$$

Two runs r_1 and r_2 are prompt-equivalent at stage i when

$$p_i^{(g_1)} = p_i^{(g_2)}.$$

In implementation, exact equality can be tested by a cryptographic digest. Let

$$h_i = H(p_i)$$

for hash function H. Then

$$h_i^{(1)} = h_i^{(2)}$$

provides strong evidence that the serialized prompt templates are identical.

The complete run specification could therefore include

$$\Theta = (M_o, M_d, M_r, M_v, h_o, h_d, h_r, h_v, \theta).$$

Persisting Θ with each run would make the experimental configuration considerably more explicit.

We can now formalize the distinction between a prompt as instruction and a prompt as guaranteed execution. Let

$$\mathcal{I}(p)$$

denote the intended transformation specified by prompt p, and let

$$\mathcal{B}(M, p, x)$$

denote the transformation behavior actually realized by model M.

In conventional deterministic programming, one often seeks approximately

$$\mathcal{B} = \mathcal{I}$$

subject to implementation correctness. In probabilistic language-model programming,

$$\mathcal{B}(M, p, x)$$

is a random variable whose distribution may include protocol violations.

Define semantic deviation

$$\delta_p = d\left(\mathcal{I}(p), \mathcal{B}(M, p, x)\right),$$

where d measures deviation from the intended role.

Prompt reliability can then be represented as

$$\mathcal{R}(p, M) = 1 - \mathbb{E}[\delta_p].$$

This quantity depends jointly upon prompt and model. A protocol that is reliable for the 8B model may not be equally reliable for the 3B model.

Theorem 7.2 (Semantic operators are model–prompt pairs)**.** *If two prompts applied to the same model can induce observably different stage behaviors, and the same prompt applied to two models can induce observably different stage behaviors, then neither the model nor the prompt alone uniquely determines the semantic operator.*

Proof. Let M be fixed and suppose there exist prompts p_1, p_2 and input x such that

$$P_M(Y \mid p_1(x)) \neq P_M(Y \mid p_2(x)).$$

Then M alone does not determine the output transformation.

Now fix prompt p and suppose models M_1, M_2 satisfy

$$P_{M_1}(Y \mid p(x)) \neq P_{M_2}(Y \mid p(x)).$$

Then p alone does not determine the transformation.

Therefore the relevant semantic operator must contain at least the ordered pair

$$(M, p),$$

with input and inference conditions supplying additional parameters. □ □

This result provides a more precise description of the pipeline than statements such as "Granite writes the essay." The system contains several operators even when the same Granite model appears at multiple stages:

$$\Omega_{3B,p_o}, \qquad \Omega_{3B,p_d}, \qquad \Omega_{8B,p_r}, \qquad \Omega_{3B,p_v}.$$

The 3B model therefore occupies three different functional roles because it is embedded in three different semantic operators.

Finally, the prompt architecture can be viewed as a constrained transition system. Let the persistent artifact state at stage i be X_i, and let

$$K_i$$

be the semantic contract encoded by prompt p_i. Then

$$X_{i+1} \sim \Omega_{M_i,p_i}(X_i)$$

is accepted as a valid stage transition only when

$$A_i(X_{i+1}; K_i) = 1.$$

The current implementation does not automatically enforce this final predicate. It records the output regardless of whether semantic compliance is perfect. This is appropriate for an experiment because violations themselves are observations. A production system might reject or regenerate noncompliant outputs; an experimental system often benefits from preserving them.

The prompt is therefore neither a decorative instruction nor a deterministic program. It is a specification presented to a stochastic computational component. Its success must be judged both by the quality of what the component produces and by the fidelity with which the intended transformation is realized.

Once prompts are treated in this manner, the entire repository becomes susceptible to a more rigorous experimental methodology. Models, prompts, frozen artifacts, stage assignments, and repeated runs can be varied independently. The next problem is consequently one of experimental design: how the exploratory shell pipeline can be transformed into a controlled laboratory for measuring the marginal contribution of planning, model asymmetry, criticism, persistence, and repair.

8 From Exploratory Pipeline to Controlled Experiment

The first executions of the essay pipeline are best understood as exploratory experiments. They demonstrate that the architecture functions, that the smaller Granite model can generate outlines and substantial drafts, that the larger Granite model can produce criticism of those drafts, and that the smaller model can subsequently attempt revision under the influence of that criticism. More importantly, the preserved artifacts expose phenomena that would have remained difficult to observe in a one-shot writing system. Reviews can identify defects that revisions fail to remove. Outlines can constrain drafts without determining them completely. A larger model can contribute locally without replacing the smaller model throughout the pipeline. These observations justify further investigation, but they do not yet establish that the architecture is superior to simpler alternatives. That question requires controlled comparison.

The central methodological difficulty is that an essay is a high-dimensional object. There is no single natural scalar quantity corresponding to "essay quality." An essay can become clearer while becoming less original, more rigorous while becoming less readable, more concise while omitting necessary qualifications, or more comprehensive while losing argumentative focus. Any experiment that compresses these dimensions into a single score risks concealing precisely the structural effects that motivated the pipeline. Evaluation should therefore preserve several dimensions of performance rather than assuming that all improvements can be reduced to one number.

The simplest useful comparison is between the staged system and one-shot generation. A topic can be supplied directly to the 3B model under an instruction to write a complete analytical essay, producing a baseline artifact E_{direct}. The same topic can then pass through the outline, draft, review, and revision pipeline, producing E_{pipe}. If independent evaluation consistently favors the latter, this would provide evidence that decomposition contributes something beyond additional token generation. Such a comparison would still be imperfect because the pipeline uses more inference calls and therefore more computation. Nevertheless, it establishes the first baseline against which architectural complexity must justify itself.

A stronger comparison would equalize computational expenditure. The one-shot model could

be given a comparable token budget, or allowed several independent generations from which one is selected. This matters because improvement attributed to staging may actually result from simply spending more inference. If four model calls outperform one call, the result does not by itself show that the particular organization of those calls matters. The relevant question is whether structured allocation of computation produces greater improvement than an alternative allocation of approximately the same resources.

This distinction between computational quantity and computational organization is fundamental. The pipeline's hypothesis is not merely that more inference is useful. It is that the topology of inference matters. One call produces a plan, another realizes it, another changes epistemic role and diagnoses the realization, and another attempts a constrained repair. If the same computational budget were spent generating four independent essays and selecting one, the system would have comparable multiplicity without the same causal structure. Comparing these conditions would test whether persistent staged dependence has value beyond repeated sampling.

The current model asymmetry introduces another experimental variable. The 8B model appears only at the review stage. This assignment should eventually be compared with a uniform 3B pipeline, a uniform 8B pipeline, and permutations in which the larger model performs outlining, drafting, or revision. Such experiments would test whether the observed usefulness of the larger critic reflects a genuine stage-specific advantage or merely the general superiority of the larger model. If replacing only the reviewer produces most of the gain obtainable from replacing every stage, the asymmetric architecture would have strong practical justification.

The review–revision divergence observed in the exploratory outputs makes the revision stage particularly important. A larger critic may identify defects that the smaller reviser cannot repair. The natural experiment is therefore to freeze a draft and its 8B review and submit the same pair independently to both Granite models for revision. Because the input artifacts are identical, differences between the resulting essays would provide direct evidence about repair capacity rather than differences in diagnosis. Conversely, a frozen draft can be reviewed independently by both models and the reviews compared before either is allowed to influence revision.

Frozen artifacts are essential because generative variability otherwise propagates downstream. If two pipelines begin by independently generating outlines, their drafts may diverge so substantially that differences in later reviews cannot be attributed cleanly to reviewer capability. By persisting intermediate states, the architecture permits intervention at an arbitrary boundary. A particular outline can become a fixed experimental specimen. A particular draft can become a fixed object of criticism. A particular review can become a fixed repair specification. This converts a sequential writing tool into a branching experimental apparatus.

Repeated sampling is equally necessary. A single model invocation is one draw from a conditional generative distribution. Even when inference settings are relatively conservative, the output should not be treated as a deterministic manifestation of the model's competence. An outline that happens to be unusually strong or weak can affect every downstream artifact. Controlled experiments should therefore repeat conditions across several random realizations and several topics. The object of interest becomes a distribution of outcomes rather than an anecdote about one essay.

The existing topics provide a useful beginning because they test related theoretical concerns without being identical. Repair and difference, admissibility and optimization, continuation and trajectories, and technological capability and diffusion require different argumentative moves. Yet all belong to a sufficiently coherent conceptual family that the same general expectations concerning analytical writing remain meaningful. Future experiments could expand this topic set while maintaining a distinction between theory-near topics, for which the prompts contain conceptual vocabulary already aligned with the research program, and more neutral topics that test whether the pipeline's advantages generalize beyond that vocabulary.

Topic selection matters because a model's apparent competence depends strongly upon the representational resources available in its training and prompt context. A topic such as optimization has extensive conventional literature and familiar argumentative structures. A more idiosyncratic concept such as repair preserving difference may require the model to infer a framework from the wording supplied to it. A pipeline that performs well only when the topic resembles established discourse may be doing something different from a pipeline that can maintain novel conceptual distinctions across several transformations. Evaluation should therefore distinguish conventional knowledge retrieval from preservation and development of locally defined theory.

This point suggests another controlled intervention: supply the same topic under different amounts of theoretical context. One condition might contain only the title. Another might contain a short statement of the intended thesis. A third might contain formal definitions drawn from the surrounding theoretical program. If the pipeline is genuinely capable of developing a supplied conceptual framework rather than merely associating familiar language, its outputs should become more structurally faithful as relevant definitions are made explicit. The experiment would thereby measure how much theory must be externalized before the model can continue it reliably.

Evaluation itself should be separated from the models being tested. Using the 8B reviewer as both a pipeline component and the final judge would introduce an obvious circularity. The review model may favor revisions that conform to its own earlier recommendations even when those recommendations are questionable. Human evaluation remains valuable, particularly for

conceptual essays whose central claims cannot be validated through automatic tests. Multiple model evaluators can also be useful if their judgments are treated as measurements with known limitations rather than ground truth.

A practical evaluation protocol could ask independent judges to compare anonymized pairs without knowing which pipeline produced them. The comparison should focus separately on thesis clarity, inferential validity, definition stability, handling of objections, structural coherence, novelty, unsupported generalization, and preservation of useful material. Pairwise comparison may be preferable to absolute numerical scoring because evaluators often discriminate which of two essays handles a particular criterion better more reliably than they assign calibrated values on an arbitrary scale.

The review artifacts permit an additional evaluation that is unavailable in endpoint-only experiments. Each criticism can be traced forward into the revision. A judge can determine whether the criticism identified a genuine problem, whether the revision attempted to address it, whether the attempted repair succeeded, and whether the intervention introduced collateral damage. This produces a causal evaluation of revision rather than merely a global comparison between drafts.

The same idea can be applied backward. If a final essay contains a defect, the investigator can ask when that defect first became visible. Some defects may originate in the outline. Others may be introduced during drafting. Still others may appear only during revision. A revision can therefore decrease some classes of error while increasing others. Error provenance provides a richer picture than final defect count.

The experimental design should also distinguish improvement from homogenization. Repeated review and revision may cause essays to converge toward generic academic prose. Such outputs can appear more polished and less obviously flawed while losing unusual conceptual distinctions. This is especially relevant for a research program that intentionally introduces nonstandard concepts. A reviewer trained predominantly on conventional academic discourse may treat novelty as ambiguity and repair it toward familiar formulations. The preservation objective should therefore include conceptual identity as well as grammatical and argumentative quality.

This concern can be operationalized by identifying invariants before revision. For a given essay, certain claims, distinctions, or definitions may be designated as constitutive of the experiment. A successful repair may clarify or qualify these invariants but should not silently replace them with more conventional propositions. The revision can then be evaluated for both defect reduction and invariant preservation. This extends the principle that repair should preserve difference: improvement should not be defined as convergence toward whatever text the model would have generated independently.

A useful controlled condition would therefore compare revision from the original draft against regeneration from the same topic after supplying the review. In the first condition, the model is explicitly asked to repair the existing artifact. In the second, it is allowed to produce a fresh essay informed by the criticism. If both yield similar quality but the repair condition preserves substantially more of the original conceptual structure, the preservation constraint has demonstrated value. If regeneration consistently produces stronger arguments without meaningful conceptual loss, the current repair philosophy would require reconsideration.

The experiment can also examine whether criticism itself should be iterative. A revised essay can be returned to the reviewer, producing a second diagnosis. The tempting stopping rule is to continue until the reviewer reports no important defects. Such a rule is unsafe because reviewer silence is not proof of correctness, and repeated revision may eventually degrade the essay. A more defensible stopping criterion would consider whether newly detected defects are diminishing, whether previously repaired defects remain absent, whether collateral changes are increasing, and whether additional interventions produce measurable improvement relative to the null action of stopping.

This introduces a notion of convergence that differs from simple textual stability. Two consecutive drafts can be nearly identical because the reviser has become unable to act upon the remaining criticism. Conversely, two drafts can differ substantially while their argumentative quality remains unchanged. Convergence should therefore be defined over defect structure and preserved invariants rather than character-level similarity alone.

The shell environment is well suited to these experiments because conditions can be represented explicitly in filenames and directories. A future run might contain a frozen draft, two reviews, four revisions produced by reviewer–reviser combinations, evaluation files, model metadata, prompt hashes, timing information, and token statistics. Ordinary Unix tools can then aggregate results across topics. The experiment need not become a large framework in order to become systematic. Indeed, preserving its textual simplicity may make the causal structure easier to inspect.

The objective is not to construct a benchmark in the conventional sense. Benchmarking usually seeks a stable score that allows models to be ranked. The more interesting question here concerns architecture: how should heterogeneous generative systems be organized so that inexpensive components can produce candidate structures, stronger components can contribute where their marginal value is greatest, intermediate states remain inspectable, and repair does not erase successful distinctions? The model ranking is secondary to the organization of model labor.

The experimental progression therefore moves from demonstration to ablation. The initial scripts show that the pipeline can operate. Controlled experiments should remove components

one at a time, replace them, reverse them, or hold their outputs fixed. If removing the outline has little effect, the planning stage may not justify its cost. If removing review causes a measurable increase in unsupported claims, criticism has demonstrated value. If replacing the 8B reviewer with the 3B model has little effect, the asymmetry may be unnecessary. If using the 8B model only for revision yields greater improvement than using it only for review, the current allocation should change.

Ablation is especially important because sophisticated pipelines can accumulate ceremonial stages. Each stage sounds intellectually defensible, and each produces an artifact that looks meaningful, but the total system may gain little from some of them. The purpose of decomposition is not to maximize the number of named cognitive operations. It is to make their contributions measurable enough that unnecessary operations can be removed.

The mature form of the project would therefore be less like an automated author and more like a laboratory for compositional cognition. Essay generation supplies a rich test domain because success requires local language production, global structure, conceptual stability, criticism, memory, and repair. Yet the architecture is general. The same experimental principles could apply to code generation, mathematical derivation, research synthesis, design documents, or any other task in which an artifact develops through inspectable transformations.

What began as a collection of Bash scripts consequently supports a broader methodological claim. The useful unit of experimentation with language models need not be the isolated model response. It can be the trajectory of persistent representations produced by several differently constrained model invocations. Once the trajectory is preserved, one can intervene on its stages, branch from earlier states, compare counterfactual continuations, and ask where additional computational capacity actually changes the reachable quality of the final artifact.

8.1 Formalization: Factorial Design, Ablation, and Causal Attribution

Let a complete pipeline configuration be represented by

$$\Theta = (O, D, C, R),$$

where O specifies the outlining condition, D the drafting condition, C the criticism condition, and R the revision condition.

Each component may contain both a model assignment and a prompt:

$$O = (M_o, p_o), \qquad D = (M_d, p_d), \qquad C = (M_c, p_c), \qquad R = (M_r, p_r).$$

For topic T_j and stochastic replicate k, let the terminal essay be

$$E_{jk}(\Theta).$$

Let

$$\mathbf{Q}(E) = \begin{pmatrix} q_1(E) \\ q_2(E) \\ \vdots \\ q_m(E) \end{pmatrix}$$

be a multidimensional evaluation vector, with components corresponding to properties such as argumentative validity, structural coherence, definition stability, objection handling, originality, and preservation.

The expected performance of configuration Θ over topic distribution $\mathcal{T}$ is

$$\boldsymbol{\mu}(\Theta) = \mathbb{E}_{T\sim\mathcal{T}}\mathbb{E}_{E\sim P(E|T,\Theta)}[\mathbf{Q}(E)].$$

A one-shot baseline can be represented as configuration

$$\Theta_0,$$

with

$$E \sim P_M(E \mid T, p_{\text{direct}}).$$

The raw pipeline effect is

$$\Delta_{\text{pipe}} = \boldsymbol{\mu}(\Theta_{\text{pipe}}) - \boldsymbol{\mu}(\Theta_0).$$

However, if computational cost differs, define

$$C(\Theta)$$

as expected inference cost. A cost-adjusted comparison may use

$$\mathbf{U}(\Theta) = \boldsymbol{\mu}(\Theta) - \lambda C(\Theta)\mathbf{1},$$

or a Pareto analysis over quality and cost rather than collapsing them into one scalar.

Ablation of stage s produces configuration

$$\Theta_{-s}.$$

The marginal contribution of stage s is then

$$\Delta_s = \boldsymbol{\mu}(\Theta) - \boldsymbol{\mu}(\Theta_{-s}).$$

Definition 8.1 (Architecturally useful stage)**.** *A stage s is architecturally useful relative to evaluation vector $\mathbf{Q}$ and cost regime C if its inclusion produces a reproducible improvement in at least one valued quality dimension without unacceptable degradation in the others or disproportionate computational cost.*

This definition avoids assuming that every stage must improve a universal scalar score.

Consider the reviewer–reviser experiment. Let

$$i \in \{3, 8\}$$

index reviewer size and

$$j \in \{3, 8\}$$

index reviser size. For frozen draft D, define

$$R_i = C_i(D)$$

and

$$E_{ij} = V_j(D, R_i).$$

For scalar quality measure Q, the reviewer main effect is

$$\Delta_C = \frac{1}{2}\left[Q(E_{83}) + Q(E_{88})\right] - \frac{1}{2}\left[Q(E_{33}) + Q(E_{38})\right].$$

The reviser main effect is

$$\Delta_R = \frac{1}{2}\left[Q(E_{38}) + Q(E_{88})\right] - \frac{1}{2}\left[Q(E_{33}) + Q(E_{83})\right].$$

The interaction is

$$\Delta_{CR} = Q(E_{88}) - Q(E_{83}) - Q(E_{38}) + Q(E_{33}).$$

If

$$\Delta_{CR} > 0,$$

the benefit of the stronger reviser is greater when the stronger review is available. This supports the hypothesis that some diagnostic information requires additional downstream capability

before it can be converted into successful repair.

Repeated sampling permits uncertainty estimation. Suppose each condition is run n times. Let

$$Q_{ijk}$$

denote score k for reviewer condition i and reviser condition j. The sample mean is

$$\overline{Q}_{ij} = \frac{1}{n} \sum_{k=1}^{n} Q_{ijk},$$

with sample variance

$$s_{ij}^2 = \frac{1}{n-1} \sum_{k=1}^{n} \left(Q_{ijk} - \overline{Q}_{ij} \right)^2.$$

The point is not that a small local experiment immediately warrants elaborate significance testing. Rather, repeated runs distinguish a stable architectural effect from a favorable single sample.

The benefit of frozen artifacts can be expressed through variance reduction. Suppose two reviewer conditions are compared using independently generated drafts:

$$D_a \sim P_D, \qquad D_b \sim P_D.$$

Then the observed difference contains variation from both reviewer and draft:

$$\mathrm{Var}(\Delta) = \mathrm{Var}_C(\Delta) + \mathrm{Var}_D(\Delta) + \text{interaction terms}.$$

If both reviewers operate on the same frozen draft D^*, the draft-level variation is conditioned away:

$$\mathrm{Var}(\Delta \mid D^*) < \mathrm{Var}(\Delta)$$

whenever draft variation contributes positively to the uncontrolled variance.

Proposition 8.2 (Frozen-state comparison improves attribution)**.** *If an upstream artifact influences downstream quality and varies across runs, then holding that artifact fixed when comparing downstream operators removes upstream variation as a direct source of between-condition difference.*

Proof. Let downstream output be

$$Y = f(X, Z),$$

where X is the upstream artifact and Z the downstream condition. Comparing

$$f(X_1, Z_1)$$

with

$$f(X_2, Z_2)$$

confounds differences in X and Z. Conditioning on

$$X_1 = X_2 = X^*$$

yields comparison

$$f(X^*, Z_1) \quad \text{versus} \quad f(X^*, Z_2),$$

so any systematic difference cannot be attributed to variation in X, which is constant by construction. □ □

The pipeline's persistent artifacts make such conditioning operationally trivial.

We can also formalize conceptual preservation. Let an essay possess a set of designated invariants

$$I(D) = \{\iota_1, \ldots, \iota_k\}.$$

For revision D', define preservation

$$P_I(D, D') = \frac{1}{k} \sum_{j=1}^{k} \mathbf{1}\left[\iota_j \text{ remains semantically preserved in } D'\right].$$

Let defect reduction be

$$\Delta L = L(D) - L(D').$$

A repair-quality functional can then be written

$$Q_{\text{repair}} = \alpha \Delta L + \beta P_I(D, D') - \gamma N(D, D') - \eta C_{\text{change}}(D, D'),$$

where N measures newly introduced defects and C_{change} measures unnecessary alteration.

This formalizes the intuition that a revision can become conventionally smoother while becoming a worse repair if it destroys the conceptual identity of the original artifact.

A stopping rule for iterative review can similarly be defined. Let

$$D_0, D_1, \ldots, D_n$$

be successive revisions and

$$L(D_i)$$

their estimated defect loss. Let

$$C_i$$

be the expected cost and collateral risk of another review–repair cycle.

Continue from D_i only if

$$\mathbb{E}\left[L(D_i) - L(D_{i+1}) \mid R_i\right] > C_i.$$

Equivalently, stop when

$$\mathbb{E}[\Delta L_{i+1}] \leq C_i.$$

This criterion treats stopping as the null intervention rather than assuming that another iteration is inherently beneficial.

Theorem 8.3 (Iteration alone does not imply convergence to admissibility)**.** *Let F be a review–revision operator over essay space $\mathcal{X}$,*

$$D_{n+1} = F(D_n).$$

Without assumptions such as contraction toward an admissible set, monotonic decrease of defect loss, or a valid stopping rule, repeated application of F does not imply convergence to an admissible essay.

Proof. Consider two essays $A, B \in \mathcal{X}$ such that the reviewer of A recommends a change causing the reviser to produce B, while the reviewer of B recommends the inverse change causing the reviser to produce A. Then

$$F(A) = B, \qquad F(B) = A.$$

Hence

$$F^{2n}(A) = A, \qquad F^{2n+1}(A) = B,$$

and the process oscillates indefinitely.

Alternatively, let F introduce one new defect for every defect removed. Then defect loss need not decrease even if textual states converge. Therefore iteration by itself supplies no convergence guarantee. □ □

A sufficient but strong condition would be contraction in an appropriate defect metric. Let

$$d_A(X, Y)$$

measure difference relative to an admissible target set. If

$$d_A(F(X), F(Y)) \leq \kappa d_A(X, Y), \qquad 0 \leq \kappa < 1,$$

then familiar fixed-point arguments become available. There is currently no reason to assume language-model revision satisfies such a condition. The experiment should therefore measure improvement rather than presuppose convergence.

Finally, consider the allocation of a fixed computational budget B. Let n_i denote the number of inference calls allocated to stage i, with per-call cost c_i. Then

$$\sum_i n_i c_i \leq B.$$

Let expected essay quality be

$$Q = Q(n_o, n_d, n_c, n_r).$$

The architecture problem is

$$\max_{n_o, n_d, n_c, n_r} Q(n_o, n_d, n_c, n_r)$$

subject to the budget constraint.

The current pipeline chooses approximately

$$(n_o, n_d, n_c, n_r) = (1, 1, 1, 1)$$

with heterogeneous model costs. This is only one point in a much larger design space. It may prove preferable to generate several outlines and one draft, one outline and several reviews, or several candidate repairs from a single diagnosis.

The general research question is therefore not

"Which model writes the best essay?"

but

"How should bounded inference be distributed across a persistent sequence of heterogeneous cognitive opera

That question transforms the project from a model demonstration into an experiment in computational organization. The scripts provide an unusually simple environment in which to study it because the operations are explicit, the states are persistent, and the apparatus can itself be versioned. What remains is to relate this organization back to the broader theoretical

motivations that produced it: continuation, admissibility, repair, preservation of difference, and the possibility that useful intelligence lies not only in increasingly capable components but in the structures through which their partial competencies are allowed to accumulate.

9 Continuation, Admissibility, and Repair as Principles of Computational Organization

The architecture developed in the preceding sections was not designed merely as an efficient way to divide an essay-writing task among language models of different sizes. Its structure reflects a more general theoretical position concerning persistence, admissibility, repair, and continuation. The central claim of that position is that a system should not be understood only by examining the states it occupies or the objective values associated with those states. It should also be understood through the constrained transitions by which one state becomes another while preserving enough organization for the resulting trajectory to remain identifiable as a continuation of the same process. The essay pipeline provides an unusually concrete environment in which this claim can be implemented rather than merely asserted.

A conventional description of automated essay generation begins with an input and ends with an output. A topic is supplied to a model, and an essay is returned. The natural abstraction is therefore a function

$$f : T \to E.$$

From the perspective of ordinary software use, this description may be sufficient. The user cares primarily about whether the resulting essay is useful. From the perspective of the present experiment, however, the mapping suppresses precisely the structure of interest. It tells us nothing about which distinctions were formed during planning, which survived drafting, which were rejected during criticism, which repairs were attempted, or whether the final artifact preserves the conceptual identity of its predecessors.

The staged pipeline instead replaces the endpoint mapping with a trajectory

$$T \to O \to D \to R \to F.$$

The important object is no longer simply F. It is the history by which F became reachable. Two final essays that appear similar at the surface may have substantially different histories. One may have emerged from an outline whose central distinction remained stable throughout drafting and revision. Another may have begun from a different argument and converged toward similar prose only after criticism eliminated its original structure. Endpoint similarity therefore does not imply process equivalence.

This is the first connection to continuation. Continuation is not treated as the persistence of an unchanged state. If that were the criterion, no substantive revision could count as continuation because every successful repair alters the artifact. Nor is continuation merely temporal succession. A completely unrelated essay written after the draft is temporally subsequent to it but does not thereby constitute its continuation in the relevant sense. Continuation instead requires structured dependence: the later state must arise from the earlier state through transformations that preserve or legitimately revise relations constitutive of the developing artifact.

The preservation requirement is consequently selective. Some features should survive a transformation while others should not. A typo has no claim to persistence merely because it existed in the draft. An unsupported inference should disappear if an admissible repair is available. A central conceptual distinction, by contrast, may be essential to the identity of the essay even if its formulation changes. Continuation therefore cannot be defined as maximal preservation. It requires discrimination between what belongs to the identity of the process and what constitutes a correctable defect within it.

The review stage makes this discrimination explicit. It identifies candidate features that should not continue unchanged. The revision stage then attempts to alter those features without destroying successful surrounding structure. The architecture thus implements a simple form of selective persistence: continuation is achieved through repair rather than through immobility.

This is closely related to the proposition that repair preserves difference. In the context of textual generation, difference refers not merely to lexical variation but to distinctions carrying argumentative information. The difference between admissibility and optimization, between trajectory and state, between capability diffusion and capability accumulation, or between repair and replacement is part of the conceptual organization of the corresponding essay. If revision makes the prose more conventional by collapsing such distinctions, it may improve stylistic smoothness while reducing theoretical content.

A repair operation should therefore be evaluated partly by what distinctions survive it. The strongest possible normalization of a text is not necessarily the strongest possible repair. An editor that replaces every unfamiliar concept with a familiar one may reduce ambiguity while simultaneously erasing the reason the essay existed. Likewise, a language model instructed only to "improve" an unusual theoretical argument may map it toward patterns more strongly represented in its training distribution. The resulting text may become easier to recognize as conventional academic prose precisely because it has become less faithful to the conceptual difference introduced by the draft.

The pipeline's preservation instruction is intended to resist this tendency. Material that has not been implicated by criticism is provisionally protected from unnecessary rewriting. This

protection is not absolute, because dependencies can require collateral change, and the reviewer can fail to detect real defects. Nevertheless, it establishes an important default: difference is not itself evidence of error.

This principle distinguishes repair from normalization. Normalization moves an artifact toward some expected or canonical form. Repair moves an artifact away from a diagnosed failure while attempting to preserve the organization that remains admissible. The two operations can coincide when deviation from the canonical form is itself the defect, as in malformed syntax. They diverge when novelty is semantically meaningful. Conceptual writing frequently occupies the latter case.

Admissibility provides the second theoretical connection. Optimization ordinarily presupposes an objective function whose values allow candidate states to be ordered. If essay revision were formulated entirely as optimization, one might posit a quality function

$$Q(E)$$

and seek

$$E^* = \arg\max_E Q(E).$$

Such a formulation is attractive but conceals several difficulties. There is no uncontested scalar quality function for theoretical essays. Different dimensions of quality conflict. More importantly, the globally highest-scoring essay need not constitute an admissible continuation of the original draft. If an unrelated but conventionally excellent essay receives a higher score, optimization alone supplies no reason to preserve the original argument.

The repair-oriented pipeline imposes a different problem. Given current artifact D, diagnosis R, and a set of permitted transformations, find a successor F that removes or reduces relevant defects while preserving the relations necessary for legitimate continuation. The question is therefore constrained by history. The successor is evaluated not only as an isolated object but relative to the state from which it emerged.

This makes admissibility inherently relational. An essay can be excellent in isolation yet inadmissible as a revision because it abandons the problem it was supposed to repair. Conversely, a modest local change can be an excellent repair even if the resulting essay is not globally optimal among all essays that could have been written on the topic. The appropriate comparison is against the reachable alternatives from the current state, including the null intervention.

The null intervention is theoretically indispensable. Once doing nothing is represented as an available action, every proposed repair must compete with preservation. Revision is no longer justified simply because a model is capable of generating another version. The intervention

should be preferred only when the expected improvement exceeds its risk of destroying successful structure, introducing new defects, or consuming resources without meaningful gain.

This gives the writing pipeline a form analogous to viability reasoning. At any stage, the current artifact occupies a region of possible continuations. Some transformations preserve the thesis while improving its support. Some produce stylistic variation with little semantic consequence. Some move the argument into contradiction. Some abandon the original conceptual program entirely. The system's problem is not to locate an abstract global optimum over all text but to remain within a region of transformations from which the developing argument can continue coherently.

The outline can be understood as an early approximation to this admissible region. It establishes constraints on what later prose should accomplish. The draft realizes one trajectory through those constraints. The review identifies locations where the trajectory appears to approach or cross a boundary of admissibility. Revision attempts to return the trajectory to a better-supported region without restarting the entire process.

The metaphor of a region should not suggest that admissibility is static. The criteria themselves can change as the argument develops. A definition introduced during drafting may reveal that an assumption in the outline was inadequate. A serious objection may require the thesis to be weakened. A repair can therefore alter not only the artifact but the representation of what counts as a successful artifact. In this respect the pipeline remains simpler than the general theory it motivates, because the current implementation largely treats prompt-defined stage criteria as fixed during a run. Nevertheless, the preserved history makes changes in those criteria conceivable and inspectable.

The distinction between state and trajectory becomes particularly important here. Consider a final essay F judged independently of its history. An evaluator may conclude that it is coherent and well written. Yet if F was produced by deleting every novel claim from D, its quality as an isolated state does not establish the quality of the transformation

$$D \to F.$$

A trajectory-sensitive evaluation asks what was lost, what was repaired, and whether the successor remains an admissible development of its predecessor.

This is why the architecture preserves the draft even after the final essay exists. If only F survived, trajectory-sensitive evaluation would become impossible. Persistence is therefore not merely a technical requirement for supplying context to later models. It is an epistemological requirement for judging transformation.

The relationship between repair and history can be stated more strongly. A defect is not always identifiable from a state alone. Suppose a revision contains a definition that is internally coherent but differs subtly from the definition upon which earlier arguments depended. Viewed locally, the new definition may appear superior. Viewed historically, it may constitute semantic drift because dependent claims were not updated. Whether the change is a repair therefore depends upon relations extending across states.

The Git repository extends this historical principle beyond the essay trajectory to the experimental apparatus itself. Just as the final essay should not erase its draft, the current pipeline should not erase the scripts and prompts from which earlier experiments were generated. The theory of continuation applies recursively. The writing process has a history, and the machinery governing that process has a history. Both histories matter when attempting to explain present behavior.

The use of local models also has a conceptual significance in this context. Because inference occurs through explicitly invoked Ollama models under local shell control, the boundaries of the process remain unusually visible. The experiment can identify which model performed which transformation and can preserve the exact textual artifact transferred between stages. This does not make the models internally transparent, but it prevents the surrounding workflow from disappearing behind a service interface.

The resulting architecture distributes competence without requiring centralized agency. No component possesses the entire essay-writing process as an enduring internal state. The shell knows the order of operations but not the meaning of the argument. The prompt templates define semantic roles but do not execute them. The 3B model generates and repairs but does not persist. The 8B model diagnoses but does not control whether its recommendations are accepted. The filesystem remembers without understanding. Git preserves history without evaluating it. The behavior of the whole emerges from the constrained interaction among these partial functions.

This distribution is relevant to the broader question of technological competence. A common response to increasingly capable models is to concentrate more work inside a single increasingly capable inference. The present experiment explores the opposite possibility: some useful intelligence may arise from external organization of imperfect components. A weaker model can become more useful when its outputs are persisted, inspected, criticized by another model, and returned under a constrained repair protocol. Capability is therefore partly a property of the surrounding architecture.

This does not imply that architecture can substitute indefinitely for model capability. The review–revision divergence demonstrates the contrary. A sufficiently weak reviser may be unable to realize a repair even when the architecture supplies an accurate diagnosis. Reacha-

bility remains constrained by the capabilities of the participating components. The important point is that capability and organization interact. Increasing either can alter the useful region reachable by the system.

The pipeline can consequently be interpreted as a small laboratory for continuation under bounded competence. Each component has limited ability. The process succeeds when the persistent structures connecting those components allow partial competencies to accumulate without allowing their failures to propagate unchecked. Planning constrains generation. Persistence prevents amnesia. Criticism exposes candidate failures. Repair attempts correction. Historical preservation makes the trajectory available for evaluation. None of these mechanisms guarantees success, but together they create conditions under which success and failure can be localized.

The phrase "repair preserves difference" acquires a concrete computational interpretation in this environment. The objective is not to preserve every token. It is to preserve distinctions that remain constitutive while modifying relations shown to be defective. The resulting artifact should therefore satisfy two competing demands:

$$\text{change enough to repair}$$

and

$$\text{preserve enough to continue.}$$

Either demand without the other produces a degenerate system. Pure preservation prevents learning. Pure replacement prevents continuity.

This tension is not a defect of the architecture. It is the problem the architecture is designed to expose.

9.1 Formalization: Admissible Continuation Under Repair

Let the textual state of the developing essay at time t be

$$X_t \in \mathcal{X},$$

where $\mathcal{X}$ is the space of possible textual artifacts.

Let a transition operator be

$$\tau_t : \mathcal{X} \to \mathcal{X},$$

so that

$$X_{t+1} = \tau_t(X_t).$$

Temporal succession alone requires only that X_{t+1} follow X_t. Define instead a continuation relation

$$X_t \rightsquigarrow X_{t+1}$$

when the transition preserves a designated set of constitutive invariants while satisfying the admissibility conditions of the next state.

Let

$$I(X_t) = \{\iota_1, \ldots, \iota_n\}$$

be the set of relevant invariants of X_t, and let

$$P_\tau(\iota_i) \in [0, 1]$$

measure the degree to which invariant ι_i survives transition τ.

Define weighted preservation

$$\Pi(\tau, X_t) = \sum_{i=1}^{n} w_i P_\tau(\iota_i), \qquad \sum_i w_i = 1.$$

Let

$$A(X_{t+1} \mid X_t, \tau_t) \in \{0, 1\}$$

denote whether the successor constitutes an admissible transformation of its predecessor.

Then continuation requires

$$X_t \rightsquigarrow X_{t+1}$$

only if

$$A(X_{t+1} \mid X_t, \tau_t) = 1$$

and

$$\Pi(\tau_t, X_t) \geq \pi_{\min}$$

for some context-dependent preservation threshold

$$\pi_{\min}.$$

This formulation immediately distinguishes continuation from identity:

$$X_{t+1} = X_t$$

is sufficient for maximal preservation but not necessarily for admissibility if X_t contains known

defects.

Conversely,

$$X_{t+1} \neq X_t$$

does not imply discontinuity if the relevant invariants survive.

Let defect loss be

$$L(X) \geq 0.$$

The null intervention is

$$\tau_0(X) = X.$$

A repair transformation

$$\rho$$

is preferable to the null intervention when

$$L(\rho(X)) + \lambda C(\rho, X) < L(X),$$

where

$$C(\rho, X)$$

represents intervention cost, including collateral structural damage.

To incorporate preservation explicitly, define

$$C(\rho, X) = C_{\text{resource}}(\rho) + \alpha\big[1 - \Pi(\rho, X)\big] + \beta N(\rho, X),$$

where N measures newly introduced defects.

The repair objective becomes

$$J(\rho \mid X) = L(\rho(X)) + \lambda_r C_{\text{resource}}(\rho) + \lambda_p\big[1 - \Pi(\rho, X)\big] + \lambda_n N(\rho, X).$$

The null intervention has

$$\Pi(\tau_0, X) = 1, \qquad C_{\text{resource}}(\tau_0) = 0,$$

but retains the full defect loss

$$L(X).$$

A nontrivial repair is warranted when

$$J(\rho \mid X) < J(\tau_0 \mid X) = L(X).$$

Definition 9.1 (Difference-preserving repair)**.** *A repair ρ is difference-preserving relative to invariant set $I(X)$ if*

$$L(\rho(X)) < L(X)$$

while

$$\Pi(\rho, X) \geq \pi_{\min}.$$

The definition excludes two degenerate operations. The identity transformation may preserve every invariant but fail to reduce loss. Arbitrary replacement may reduce some measured loss while failing the preservation threshold.

Proposition 9.2 (Neither preservation nor improvement alone is sufficient for repair)**.** *There exist transformations with maximal preservation that are not repairs, and transformations with reduced defect loss that are not admissible continuations.*

Proof. For the first claim, let

$$\tau_0(X) = X$$

for defective X. Then

$$\Pi(\tau_0, X) = 1,$$

but

$$L(\tau_0(X)) = L(X).$$

No defect has been repaired.

For the second claim, let Y be an unrelated essay with

$$L(Y) < L(X)$$

under an isolated quality measure but with

$$\Pi(X \to Y) < \pi_{\min}.$$

Then replacement by Y reduces measured loss but fails the continuation criterion. Therefore neither preservation nor isolated improvement is individually sufficient. □ □

The admissible repair set is consequently

$$\mathcal{R}_A(X) = \{\rho : J(\rho \mid X) < L(X),\ \Pi(\rho, X) \geq \pi_{\min}\}.$$

The revision problem is

$$\rho^* = \arg\min_{\rho \in \mathcal{R}_A(X)} J(\rho \mid X).$$

This differs fundamentally from unconstrained essay optimization

$$X^* = \arg\min_{Y \in \mathcal{X}} L(Y),$$

because the admissible repair problem is indexed by the current state X. History enters the objective.

We can make this dependence explicit by defining a trajectory

$$\gamma = (X_0, X_1, \ldots, X_n).$$

Let

$$A_\gamma(X_{t+1})$$

denote admissibility relative not merely to X_t but to the relevant trajectory prefix

$$\gamma_{\leq t} = (X_0, \ldots, X_t).$$

Then

$$A_\gamma(X_{t+1}) = A\left(X_{t+1} \mid X_0, \ldots, X_t\right).$$

This permits detection of historical inconsistency. A new definition may be locally coherent relative to X_t but incompatible with commitments established in X_{t-2} that remain active.

Theorem 9.3 (Endpoint quality does not determine trajectory admissibility)**.** *There exist trajectories γ_1, γ_2 terminating in endpoint-equivalent states under an isolated quality functional Q, while only one trajectory preserves the required historical invariants.*

Proof. Let

$$Q(F_1) = Q(F_2).$$

Suppose trajectory

$$\gamma_1 : D \to F_1$$

preserves invariant set $I(D)$, so that

$$\Pi(\gamma_1) \geq \pi_{\min}.$$

Let

$$\gamma_2 : D \to F_2$$

replace the original thesis with an unrelated but equally high-quality thesis, yielding

$$\Pi(\gamma_2) < \pi_{\min}.$$

Because Q evaluates endpoints independently of historical preservation,

$$Q(F_1) = Q(F_2)$$

does not distinguish the trajectories. Yet only γ_1 satisfies the continuation condition. Therefore endpoint quality is insufficient to determine trajectory admissibility. □ □

The architecture can also be expressed through reachable sets. Let

$$\mathcal{K}_M(X)$$

denote states reachable from X by model M under available prompts and context. Let

$$\mathcal{A}_X$$

denote states that are admissible continuations of X.

The useful reachable region is

$$\mathcal{V}_M(X) = \mathcal{K}_M(X) \cap \mathcal{A}_X.$$

Adding a reviewer does not necessarily enlarge

$$\mathcal{K}_M(X)$$

directly. Instead, it can change the conditioning information and thereby alter the effective reachable set:

$$\mathcal{K}_M(X) \to \mathcal{K}_M(X \mid R).$$

The review is useful when

$$|\mathcal{K}_M(X \mid R) \cap \mathcal{A}_X| > |\mathcal{K}_M(X) \cap \mathcal{A}_X|$$

in an appropriate measure.

This provides a formal interpretation of criticism as reachability modification. The reviewer need not perform the repair itself. Its output is valuable if it reshapes the reviser's conditional search space so that admissible successors become more reachable.

The review–revision failure observed earlier occurs when

$$R$$

correctly improves information about

$$\mathcal{A}_X$$

but

$$\mathcal{K}_{3B}(X \mid R) \cap \mathcal{A}_X$$

remains small or empty for the required repair.

A larger reviser may instead satisfy

$$\mathcal{K}_{8B}(X \mid R) \cap \mathcal{A}_X \neq \varnothing.$$

The distinction between knowledge of the admissible region and reachability of that region is therefore mathematically explicit.

Finally, define the complete pipeline trajectory

$$\Gamma = (T, O, D, R, F)\,.$$

Because R is a diagnostic artifact rather than a replacement state of the essay, a more precise representation is a branching graph:

$$T \to O \to D,$$

$$D \to R,$$

$$(D, R) \to F.$$

Let

$$\mathcal{H}(\Gamma)$$

denote the persisted history of this graph. If only F remains, then

$$\mathcal{H}(\Gamma) \to \{F\},$$

and information required to evaluate preservation, defect provenance, and repair uptake is lost.

Thus persistent history is not ancillary to trajectory-sensitive evaluation. It is a necessary observable.

The architecture can therefore be summarized by the coupled conditions

$$\text{Persistence} + \text{Diagnosis} + \text{Admissible Repair} + \text{Invariant Preservation} \implies \text{Inspectable Continuation}.$$

The implication is not a guarantee of correctness. It identifies the conditions under which continuation becomes available for inspection rather than disappearing into a single terminal generation.

This provides the deeper motivation for the experiment. The essay pipeline is useful not because essays are uniquely important computational objects, but because writing makes the problem of continuation unusually visible. A text acquires structure, commits itself to distinctions, encounters objections, and changes while attempting to remain recognizably the development of an earlier argument. The same problem appears wherever computational systems must improve without treating their existing organization as disposable.

Under this interpretation, the pipeline is a deliberately small model of a much larger question: how can a bounded computational system change what is wrong without erasing what made its previous state worth continuing?

10 Heterogeneous Local Models and the Unix Philosophy of Compositional Intelligence

The decision to construct the experiment from shell scripts, ordinary files, prompt templates, a small Python rendering utility, Git, and locally executed Ollama models is not merely an implementation convenience. It expresses a particular view of computational organization. Rather than treating intelligence as a property that must be concentrated inside a single increasingly capable model invocation, the pipeline treats useful cognitive behavior as something that can emerge from the composition of partial competencies whose interactions remain externally visible. The distinction matters because the monolithic and compositional approaches answer different engineering questions. A monolithic system asks how capable a single model can

become when supplied with a sufficiently elaborate instruction and context. A compositional system asks how much reliable work can be obtained by arranging bounded components so that the output of one becomes a persistent and inspectable condition for the operation of another.

The Unix tradition provides an unusually appropriate conceptual background for this experiment. Its familiar engineering preference for small programs connected through simple interfaces is sometimes reduced to the slogan that each program should do one thing well. That formulation is useful but incomplete for present purposes. The deeper property is that intermediate representations remain available outside the components that transform them. A program reads a stream or file, performs an operation, and emits another representation that can be inspected, redirected, transformed, archived, compared, or supplied to an entirely different program. The surrounding environment therefore does not require every component to understand the complete computation in which it participates.

The essay pipeline applies this principle to language-model inference. The outlining model does not need to know how the final essay will be reviewed. The reviewer does not need to know how the outline was generated. The reviser does not need direct access to the hidden activations that produced the review. Each stage receives a textual representation sufficient for its local task. The filesystem functions as the interface through which these otherwise transient computational processes become composable.

This is a significant departure from conversational use of a language model. In a long conversation, intermediate reasoning is often represented implicitly by the accumulated context. The system appears to remember earlier decisions because those decisions remain somewhere in the conversation supplied to later inference. Such persistence is useful, but it leaves several operations entangled. The user may not know which earlier statements remain influential, whether a later response has silently abandoned an earlier distinction, or which part of the accumulated context caused a particular change. The conversation itself becomes both history and active memory.

The file-oriented architecture separates these functions. `outline.md` is not merely something that happened earlier; it is a named representation with a particular semantic role. `draft.md` is not overwritten when `essay.md` is created. `review.md` remains available independently of whether its recommendations are accepted. The system therefore possesses not one undifferentiated context but several typed historical objects. Their types are conventional rather than enforced by a formal type system, but the distinction is operationally real because different scripts and prompts consume them differently.

The resulting architecture resembles a collection of communicating processes more than a single artificial author. Ollama supplies inference processes. Bash supplies orchestration. Python

performs deterministic prompt substitution. The prompt files define local semantic contracts. Markdown files carry state across process boundaries. Git supplies historical persistence. None of these components independently constitutes the essay-writing system. The system exists in their composition.

This distribution has practical consequences for model selection. If every stage were hidden inside one model invocation, the model would have to be selected for the hardest operation that might arise anywhere in the task. If strong criticism requires the 8B model, the entire essay would ordinarily be generated with the 8B model even if outlining and first-draft production are adequately handled by the 3B model. The staged architecture permits computational capability to be allocated locally. The expensive component is invoked where its marginal contribution is expected to be greatest.

This is analogous to heterogeneous computing more generally. A computer does not require every operation to execute on its most specialized or expensive processor. Different tasks can be assigned to CPUs, GPUs, accelerators, storage devices, or network services according to their computational structure. The important question is not which component is globally strongest but which assignment of operations to components produces the best behavior under resource constraints.

Language models introduce an unusual version of this problem because the components differ not only in speed but in apparent cognitive capacity. The 8B model may detect argumentative defects that the 3B model misses. The 3B model may nevertheless be sufficient for generating large amounts of competent prose. If so, using the larger model for every token wastes capacity on transformations for which its marginal advantage is small. Conversely, using only the smaller model may make certain admissible successors unreachable. The architecture attempts to exploit the region between these extremes.

The current assignment should therefore be regarded as a hypothesis rather than a doctrine. The 3B model performs outlining, drafting, and revision; the 8B model performs review. This division was motivated by the intuition that generation tolerates approximation while criticism benefits disproportionately from additional capacity. The exploratory results provide some support for the second half of that intuition because the larger reviewer identifies meaningful conceptual problems. They simultaneously cast doubt on whether the smaller model should always perform the final repair. A correct diagnosis that cannot be acted upon reveals a mismatch between informational and transformational capability.

This is exactly the kind of mismatch that a compositional architecture makes visible. In a monolithic 8B invocation, successful repair would reveal little about whether the model's advantage arose from better diagnosis or better rewriting. In a staged architecture, those capacities can be separated experimentally. The larger model may be valuable primarily because

it changes the informational environment in which the smaller model operates. Alternatively, its diagnostic output may expose repairs that remain beyond the smaller model's reachable set. These possibilities imply different optimal allocations of inference.

The architecture also makes substitution unusually cheap. Because stages communicate through ordinary text, replacing a model does not require redesigning the entire system. If another local model becomes available, it can be tested as a reviewer while the existing outline, draft, prompt, and revision stages remain fixed. A remote model could even be introduced at one boundary without forcing the rest of the system into a remote service architecture. The interface contract is textual rather than model-specific.

This property gives the system a form of model independence. It is not absolute independence because models differ in prompt syntax, context limits, tokenization, instruction following, and other behaviors. Nevertheless, the conceptual architecture does not depend upon Granite itself. Granite is the current experimental substrate. The deeper object is the graph of transformations and persistent artifacts.

This distinction is especially important because model ecosystems change rapidly. An architecture tied to the identity of one model can become obsolete when that model is replaced. An architecture expressed through roles such as planner, drafter, critic, and reviser can survive model substitution. The scientific question then becomes comparative: which available model best occupies which role under a particular computational budget?

The use of local models strengthens this experimental freedom. There is no per-request interface hidden behind an external application whose behavior can change without notice. The model names are explicit in the scripts. The experiment can be run repeatedly without sending the developing theoretical material to an external inference service. Latency and computational cost are experienced directly on the local machine rather than converted into an abstract API charge. This makes model size a tangible experimental resource.

Local execution also changes the economics of repeated experimentation. Once the model weights are available, another generation is primarily a cost in time, electricity, and occupied computational resources rather than a separately metered transaction. This encourages a style of investigation in which many small variations can be attempted without every branch becoming a purchasing decision. Frozen artifacts can be reviewed repeatedly. Alternative prompts can be tested against identical drafts. Several revisions can be generated and compared. Failed experiments can remain cheap enough to be informative.

At the same time, local inference imposes real constraints. A larger model consumes more memory, takes longer to generate, and may compete with other processes for limited hardware. These constraints are not incidental inconveniences to be abstracted away. They are part of the

motivation for heterogeneous allocation. If computation were unlimited, the simplest architecture might be to use the strongest available model at every stage and sample it repeatedly. Under bounded resources, the location of expensive inference becomes a design variable.

This is where the Unix-style decomposition and the theoretical concern with admissibility converge. Each component is locally bounded, but the persistent environment allows its useful contribution to survive beyond the process that produced it. The 8B reviewer does not need to remain resident as an agent supervising the 3B model. It can execute once, externalize its diagnosis into `review.md`, and terminate. Its contribution persists as text. The computational process disappears while the informational consequence remains available.

The distinction between persistent artifact and transient competence is fundamental. During inference, the 8B model temporarily instantiates capacities that are unavailable when it is not running. Once its output is written to disk, some consequence of those capacities has been converted into a durable representation. The pipeline can then attempt to exploit that representation using cheaper processes. This is a form of computational amortization: expensive cognition is invoked briefly and its output is reused.

Yet the review–revision experiments show why durable information should not be confused with durable competence. The review may preserve the conclusion of a sophisticated critical operation without preserving the entire process that made the conclusion actionable. A sentence identifying an ambiguity is a compressed residue of a much larger computation. If the smaller model cannot reconstruct enough of the missing structure from that residue, the expensive competence has not been fully transferred.

This suggests that intermediate representations themselves can be optimized. A review is not merely an explanation for a human reader. It is an interface between heterogeneous cognitive components. Its structure determines how much of the reviewer's useful computation can survive serialization into text. A poorly designed intermediate representation can become an information bottleneck even when both participating models are individually capable.

The problem resembles interface design in conventional software. Two powerful components can fail to cooperate if the representation exchanged between them omits necessary state. Increasing the power of either component may not solve the problem. The interface itself must be enriched. In the present system, this could mean moving from general criticism toward explicit repair specifications, localized references, proposed formal distinctions, dependency annotations, or confidence estimates. The appropriate representation should be discovered experimentally rather than assumed.

The filesystem makes such experimentation natural because intermediate representations are first-class artifacts. A future `review.json` could coexist with `review.md`. The former might

contain structured machine-oriented diagnoses while the latter remains human-readable. A revision stage could consume both. Nothing about the surrounding architecture requires that every cognitive interface remain free-form prose.

Nevertheless, beginning with plain text has methodological advantages. It minimizes infrastructure, keeps every artifact directly inspectable, and prevents the experiment from prematurely encoding an elaborate ontology of writing errors. The current system allows categories to emerge from observed failures before they are frozen into schemas. This is consistent with the exploratory character of the project: representation should become more structured when repeated observations justify the structure.

The shell scripts themselves embody a similar preference for transparent composition over framework construction. A larger software system could provide queues, databases, dependency graphs, model abstractions, configuration management, and dashboards. Such features may eventually become useful, but they would also introduce additional layers between the investigator and the actual transformations. Bash is sufficient because the current computational graph is small and sequential. Its limitations become useful constraints: the architecture remains legible.

This legibility matters because the object of study is not merely the generated essay. The apparatus itself is part of the argument. A researcher can open `essay-pipeline.sh`, inspect the order of commands, inspect the prompt templates, examine the intermediate files, run an individual stage manually, replace a model name, or use standard tools such as `diff`, `grep`, `sed`, and `git` to interrogate the experiment. There is little distinction between using the system and inspecting its implementation.

The `icepick.sh` utility extends this philosophy by collapsing the distributed repository into a single textual view when desired. Ordinarily, modularity is advantageous because each file has a distinct function. For inspection, however, excessive distribution can make the total system difficult to apprehend. Concatenating the relevant scripts, prompts, and generated texts creates a temporary observational representation of the whole. The repository can therefore alternate between decomposed execution and aggregated inspection.

This duality is characteristic of good experimental tooling. The representation best suited to operation need not be the representation best suited to analysis. The filesystem stores separate artifacts because separation preserves provenance and enables selective execution. The concatenated icepick output places those artifacts into a common context because analysis often benefits from seeing their relations at once. Neither representation replaces the other.

The use of Git adds another compositional layer. A commit does not merely save files. It binds a collection of artifact states into a historical coordinate. The exact scripts, prompts, and

outputs associated with a particular experimental state can be recovered together. This makes it possible to ask not merely what the current pipeline does but what the pipeline did when a particular essay was generated.

The earlier monotonic-commit experiments make this point especially vivid. Building an artifact incrementally while committing successive changes converts its history into an inspectable sequence rather than a before-and-after comparison. The essay pipeline generalizes the same intuition at a coarser semantic level. Instead of treating history as something to discard once the latest state exists, history becomes part of the experimental object.

The resulting computational philosophy can be summarized as externalized cognition under explicit continuation. Models perform transient transformations. Files preserve their consequences. Prompts constrain their roles. Shell scripts establish causal order. Git preserves the evolution of the apparatus. Stronger computation is introduced selectively where cheaper computation appears insufficient. The system does not attempt to simulate a unified mind. It constructs a sequence of bounded operations whose interactions can be observed.

This is not merely a concession to limited hardware. It represents an alternative answer to the question of what increasingly capable artificial systems should look like. One possibility is continued concentration: larger contexts, larger models, more internal tool use, and increasingly autonomous agents whose internal organization becomes difficult for the user to inspect. Another possibility is compositional: models remain replaceable components embedded in persistent environments where plans, diagnoses, decisions, and repairs exist as inspectable artifacts outside any one inference.

The two approaches are not mutually exclusive. A powerful model can itself be one component in a compositional system. The important distinction concerns where organizational structure resides. In a monolithic architecture, much of the organization is internal to the model interaction. In the present architecture, a significant fraction is deliberately moved into the surrounding environment.

External organization creates friction. Files must be named. Stages must be specified. Prompts must be maintained. Intermediate outputs occupy disk space. Errors that an integrated system might conceal become visible. But that friction is partly the source of experimental value. The architecture forces distinctions that a one-shot system allows to collapse.

The pipeline therefore embodies a claim about competence that parallels the argument concerning technological diffusion. Useful capability need not remain enclosed within the component that originally produced it. A stronger model can externalize a diagnosis into a representation that becomes available to weaker components, humans, future models, and later experiments. Whether that externalization succeeds completely is an empirical question, but the architectural

objective is clear: computational competence should leave behind inspectable structure rather than disappearing when inference ends.

10.1 Formalization: Heterogeneous Allocation and Externalized Competence

Let the pipeline contain a set of stages

$$S = \{s_1, s_2, \ldots, s_n\}$$

and a set of available models

$$M = \{M_1, M_2, \ldots, M_k\}.$$

Let

$$a : S \to M$$

be a model-allocation function assigning one model to each stage.

For stage s_i, model M_j has expected stage-specific capability

$$q_{ij} = \mathbb{E}\left[Q_i(\Omega_{M_j,p_i}(X_i))\right],$$

where Q_i evaluates success relative to the semantic contract of stage i.

Let the expected computational cost be

$$c_{ij} > 0.$$

A naive uniform allocation selects one model M_j and uses

$$a(s_i) = M_j$$

for every i.

A heterogeneous allocation solves

$$a^* = \arg\max_a \mathbb{E}[Q_{\text{terminal}}(a)]$$

subject to

$$\sum_{i=1}^{n} c_{i,a(s_i)} \leq B,$$

where B is the available computational budget.

If stage quality contributed independently to terminal quality, a simplified objective could be

written

$$\max_{a} \sum_{i=1}^{n} w_i q_{i,a(s_i)}$$

subject to the same budget constraint. In practice, stage interactions make the true objective nonseparable:

$$Q_{\text{terminal}} = Q(q_{1,a(s_1)}, \ldots, q_{n,a(s_n)}, I_{12}, I_{23}, \ldots),$$

where I_{ij} represents interactions between stages.

This interaction term explains why selecting the individually strongest model for each isolated task may still fail to produce the strongest pipeline. The output representation of one stage must be usable by the next.

Let

$$Z_i$$

be the persistent representation emitted by stage i. The next stage computes

$$Z_{i+1} \sim \Omega_{a(s_{i+1}),p_{i+1}}(Z_i).$$

Define interface usability

$$U_{ij}(Z_i)$$

as the degree to which model M_j can exploit the information encoded in Z_i.

The effective contribution of stage i is therefore not merely its intrinsic quality

$$Q_i(Z_i),$$

but something closer to

$$Q_i^{\text{effective}} = Q_i(Z_i) \cdot U_{i+1,a(s_{i+1})}(Z_i).$$

A sophisticated review with low downstream usability may contribute less to terminal quality than a simpler but more actionable review.

This yields an interface bottleneck result.

Theorem 10.1 (Intermediate representation bottleneck)**.** *Suppose stage A possesses information sufficient to identify an admissible downstream action, but the serialized representation Z emitted by A does not contain sufficient information for downstream component B to distinguish that action from inadmissible alternatives. Then increasing the internal capability of A without changing Z need not improve the terminal system.*

Proof. Let internal state h_A contain information I^* required to select admissible action r^*. Let serialization

$$\sigma(h_A) = Z$$

discard some information necessary to distinguish r^* from alternative r'.

The downstream component receives only Z, so its conditional decision is

$$P_B(r \mid Z).$$

Any additional information in h_A that is not preserved by σ is conditionally unavailable to B.

Now increase the capability of A, producing richer internal state

$$h'_A.$$

If

$$\sigma(h'_A) = \sigma(h_A) = Z,$$

then the downstream conditional distribution remains

$$P_B(r \mid Z).$$

Therefore the increase in upstream capability produces no necessary improvement in downstream behavior. □ □

The theorem explains why prompt and artifact design must be considered alongside model scaling.

We can describe externalization more directly. Let

$$H_M$$

denote a transient internal computational state of model M, and let

$$\sigma : H_M \to Z$$

be serialization into persistent textual artifact Z.

Once inference terminates, assume H_M is unavailable to the pipeline. Only

$$Z = \sigma(H_M)$$

persists.

The retained information about some task-relevant variable Y is

$$I(Y;Z),$$

whereas the internal computation may have contained

$$I(Y;H_M).$$

By the data-processing inequality,

$$I(Y;Z) \leq I(Y;H_M)$$

when

$$Y \to H_M \to Z$$

forms the relevant information-processing chain.

Thus externalization is necessarily a potential bottleneck. The objective is not to preserve the entire internal state, which is neither available nor desirable, but to construct Z such that the task-relevant information required downstream is retained.

Define externalization efficiency

$$\eta_{\text{ext}} = \frac{I(Y;Z)}{I(Y;H_M)}$$

conceptually, with

$$0 \leq \eta_{\text{ext}} \leq 1.$$

Although these mutual informations are not directly measurable in the current experiment, the formulation clarifies the design problem. A stronger reviewer is useful only insofar as its useful distinctions survive into a representation that can influence later computation.

Persistence changes the temporal availability of this information. Let

$$Z(t)$$

denote availability of the serialized artifact after the producing inference has terminated. Without persistence,

$$Z(t) = \varnothing$$

for later independent processes unless the output is manually reconstructed.

With filesystem persistence,

$$Z(t) = Z$$

for all later times at which the file remains accessible.

Thus the expensive operation

$$H_M \to Z$$

can be amortized across several downstream transformations:

$$B_1(Z), \quad B_2(Z), \quad \ldots, \quad B_m(Z).$$

If the upstream inference cost is C_A, the effective upstream cost per downstream experiment becomes

$$\frac{C_A}{m}$$

when the same frozen artifact is reused m times, ignoring storage and coordination costs.

This is one formal advantage of persistent intermediate representations over repeatedly regenerating context.

The Unix-style architecture can be represented as a directed graph

$$G = (V, E),$$

where vertices include both transformations and persistent artifacts. Let transformation nodes be

$$V_T$$

and artifact nodes

$$V_A.$$

Edges alternate:

$$V_A \to V_T \to V_A.$$

For the current pipeline,

$$T \to P_O \to O \to P_D \to D,$$

followed by

$$D \to P_R \to R,$$

and finally

$$(D, R) \to P_V \to F,$$

where P_i denotes a model invocation under stage prompt i.

This bipartite structure has an important property: transformation nodes may disappear after execution while artifact nodes persist.

Let

$$\text{life}(P_i)$$

be the lifetime of an inference process and

$$\text{life}(Z_i)$$

the lifetime of its persisted output. Ordinarily,

$$\text{life}(Z_i) \gg \text{life}(P_i).$$

The architecture therefore converts short-lived computation into long-lived constraints on future computation.

Model substitution can now be defined precisely. Suppose

$$P_i = (M_a, p_i).$$

Replace it with

$$P_i' = (M_b, p_i)$$

while holding its incoming artifact edges and outgoing artifact type fixed.

If the surrounding graph remains executable, the architecture is substitutable at stage i.

Define substitution cost

$$C_{\text{sub}}(M_a \to M_b, s_i)$$

as the amount of nonlocal modification required to perform the replacement. Text-mediated modularity seeks

$$C_{\text{sub}} \approx 0$$

relative to tightly integrated architectures.

The model-allocation problem can finally be expressed in terms of marginal value. Let

$$M_s$$

be the smaller model and

$$M_\ell$$

the larger model. At stage i, define the marginal quality gain

$$\Delta q_i = q_{i\ell} - q_{is}$$

and marginal cost

$$\Delta c_i = c_{i\ell} - c_{is}.$$

The stage-specific return on additional inference expenditure is

$$\rho_i = \frac{\Delta q_i}{\Delta c_i},$$

provided

$$\Delta c_i > 0.$$

Under a simple additive approximation, scarce larger-model capacity should first be allocated to stages with high

$$\rho_i.$$

The present architecture implicitly hypothesizes

$$\rho_{\text{review}} > \rho_{\text{draft}},$$

which is why the 8B model is assigned to criticism rather than routine prose generation.

The experiment is designed so that this inequality can eventually be tested rather than merely believed.

More generally, let

$$C_{\text{system}}$$

denote useful competence of the complete pipeline. The compositional hypothesis is not

$$C_{\text{system}} = \max_j C(M_j).$$

Nor is it simply

$$C_{\text{system}} = \sum_j C(M_j).$$

Instead,

$$C_{\text{system}} = \Phi\left(\{C(M_j)\}, G, \{p_i\}, \{Z_i\}, \mathcal{H}\right),$$

where G is the orchestration graph, p_i are semantic protocols, Z_i are persistent intermediate representations, and $\mathcal{H}$ is preserved history.

The function Φ expresses organization.

Two systems containing identical models can therefore possess different effective competence because their components are connected differently. Likewise, a system containing a weaker model can sometimes outperform a poorly organized system containing a stronger one, while still remaining bounded by the transformations its components can reach.

This leads to the architectural proposition motivating the entire experiment:

Effective intelligence is a property of capability under organization, not capability in isolation.

The proposition does not diminish the importance of model scale. It specifies where scale enters a larger computational system. Larger models enlarge some reachable sets. Persistent representations preserve selected consequences of those enlarged sets. Interfaces determine whether downstream components can exploit them. Repair determines whether new information can alter an artifact without destroying its successful structure. History determines whether that alteration remains inspectable as continuation.

The small collection of Bash scripts in the repository is therefore sufficient to pose a question considerably larger than automated essay writing. If increasingly capable models are treated as transient computational resources rather than self-contained agents, what forms of persistent external organization allow their competencies to accumulate, remain inspectable, and become available to components that do not individually possess them?

The answer cannot be obtained by scaling a single invocation alone, because the question concerns what happens between invocations. It concerns the architecture of continuation.

11 Provenance, Reproducibility, and the Monotonic Construction of Knowledge Artifacts

The persistence of intermediate representations acquires a second significance when the essay pipeline is placed inside a version-controlled repository. The filesystem preserves states; Git preserves relations among states. This distinction transforms the repository from a container of generated documents into a historical record of computational construction. An outline can be inspected independently because it exists as a file, but a commit history can additionally

establish when that outline appeared, which scripts and prompts existed when it was produced, what changed before the subsequent draft was generated, and whether later modifications altered the apparatus itself or only its outputs. Provenance therefore extends the concept of continuation from the essay to the experiment that produced the essay.

This historical perspective is closely related to the monotonic commit experiments that preceded the present pipeline. In those experiments, a document was deliberately constructed incrementally, with successive portions committed rather than allowing the completed artifact to appear in the repository as an unexplained whole. The purpose was not that every line of a text naturally deserves its own commit. Such granularity would ordinarily be excessive for conventional software development. The experiment instead exaggerated a property that version control already possesses: a finished artifact can be represented not merely as a state but as the terminal point of a recoverable sequence of transformations.

The distinction between a terminal artifact and its construction history is particularly important for generated material. A completed essay can easily create an illusion of simultaneity. Once several thousand words exist in a single Markdown or LaTeX file, nothing visible in that file reveals whether the text was produced in one inference, assembled from several independent generations, progressively revised over several days, or generated by heterogeneous models under distinct prompts. The terminal document contains the result but not the causal organization that produced it.

This problem is not unique to language models. Scientific papers similarly compress months or years of experiments into a polished narrative. Software releases compress thousands of intermediate edits into a working version. Mathematical publications usually suppress abandoned conjectures and failed proof strategies. Such compression is often necessary for communication, but it can obscure the process required to evaluate provenance, reproduce results, or understand why a particular representation was adopted. Version control provides one mechanism for preserving part of this discarded temporal structure.

In the essay pipeline, the problem becomes unusually tractable because many relevant intermediate states are already textual. Prompts, topics, outlines, drafts, reviews, revisions, shell scripts, configuration values, and logs can all be represented in ordinary files. Git can therefore preserve not only source code but a significant portion of the cognitive apparatus and its outputs within the same historical system. The repository becomes an executable laboratory notebook.

The phrase "executable laboratory notebook" should be interpreted carefully. A traditional laboratory notebook records observations and procedures in enough detail that a researcher can reconstruct what was done. The repository adds the possibility that some of those procedures are themselves executable. The shell scripts do not merely describe that the 3B model should

produce an outline; they invoke the model. The prompt templates do not merely summarize the intended review criteria; they instantiate the instructions supplied during review. The generated artifacts are stored alongside the mechanisms that produced them. Documentation and apparatus therefore partially coincide.

This does not guarantee reproducibility in the strongest sense. A stochastic language model can produce different output on another execution. Ollama versions can change. Model files identified by the same human-readable name may eventually differ across environments unless their underlying identities are recorded. Hardware and inference settings can affect performance. Shell utilities and Python versions can change. A repository containing the scripts is therefore not equivalent to a perfectly reproducible computational environment.

Nevertheless, reproducibility is not binary. There is a substantial difference between an essay whose provenance consists only of the statement "generated with an AI model" and one for which the topic, model assignments, prompts, orchestration script, intermediate artifacts, timestamps, and repository state are recoverable. The second system does not eliminate uncertainty, but it narrows the set of plausible histories considerably.

This suggests that provenance should itself be decomposed. One can ask whether the semantic inputs are recoverable, whether the computational procedure is recoverable, whether the model identity is recoverable, whether inference parameters are recoverable, whether the software environment is recoverable, and whether the exact stochastic trajectory can be replayed. Different experiments require different levels of reconstruction. For conceptual analysis of the current essays, exact token-for-token regeneration may be less important than knowing which prompt and model produced each artifact. For benchmarking model behavior, stronger environmental control would become necessary.

The current repository already captures the most important conceptual layer because the semantic transformations are explicit. A reader can determine that an outline preceded a draft, that the review operated on that draft, and that the final essay was conditioned on both draft and review. This is sufficient to reconstruct the causal topology even if the exact random generation cannot be replayed.

The topology matters because provenance is not simply a list of files. A folder containing `outline.md`, `draft.md`, `review.md`, and `essay.md` suggests an ordering through its naming conventions, but the scripts establish the actual dependency relation. The review depends upon the draft. The final revision depends jointly upon draft and review. The outline and topic constrain the draft. Provenance therefore has graph structure.

Git adds a second graph. The artifact dependency graph describes how files within an experimental run depend upon one another. The commit graph describes how the repository

itself evolves. These graphs should not be conflated. An essay's review may be generated after its draft while both files are introduced in the same commit. Conversely, a prompt may change in one commit and influence dozens of later experimental runs. Understanding the experiment requires both local artifact provenance and global repository provenance.

This distinction provides a reason not to rely on Git alone as the representation of pipeline state. A commit history is excellent for reconstructing changes to a repository, but semantic stages should still be represented explicitly in the directory structure. Requiring an investigator to infer the existence of a review stage by comparing arbitrary commits would make the experiment unnecessarily opaque. The filesystem represents the logical architecture; Git represents its evolution.

The monotonic construction principle can consequently be generalized. A monotonic history need not mean that content is never removed. Revision necessarily deletes and replaces text. Rather, monotonicity can be applied to evidence: later states should not require earlier states to disappear. The system may move from D to F, but the existence of F should not destroy the evidence that D existed. A criticism can supersede an earlier judgment without erasing the earlier judgment from history. A prompt can be improved without making the previous protocol unrecoverable.

This is a form of epistemic monotonicity at the level of records rather than beliefs. The system's current conclusions can change non-monotonically while the archive of how those conclusions changed grows monotonically. Such a distinction is crucial. Intellectual progress often requires withdrawing claims, correcting definitions, abandoning models, and revising arguments. A knowledge system that prohibited retraction would be unusable. A provenance system, however, can preserve the fact that a retracted state once existed.

The repository therefore distinguishes current validity from historical existence. Git's ordinary semantics already embody this distinction. Deleting a file from the current working tree does not necessarily remove it from previous commits. Replacing a prompt does not imply that the earlier prompt never existed. The current state can remain clean while the historical state remains recoverable.

This provides an architectural response to a common problem in AI-assisted work: silent overwriting. When a generated answer is replaced by a supposedly improved version, the user may lose the ability to determine what changed. If the new version contains a subtle regression, reconstructing the previous state may require memory or manual backups. Persistent artifacts and version control instead make comparison normal. Improvement becomes a claim that can be tested against a predecessor.

The same principle applies to generated code. A model may repair one bug while introducing

another. If the original implementation disappears, evaluation is limited to the new state. If both versions remain available, the repair can be examined as a transformation. The conceptual framework developed for essays therefore extends naturally to software engineering: diagnosis, repair, invariant preservation, collateral damage, and historical provenance are equally relevant.

The `icepick.sh` script occupies an interesting place in this historical architecture. By concatenating textual files into a single representation, it creates a snapshot optimized for reading rather than execution. The original repository remains decomposed, but the snapshot captures enough of its distributed state to allow the experiment to be examined as a whole. In the present case, that snapshot became the basis for describing the pipeline itself. The system therefore generated an observational artifact about its own structure.

There is a mild recursion here. Scripts generate essays. Another script gathers the scripts, prompts, and essays into a common textual representation. That representation becomes input to an analysis explaining how the generation system works. The analysis can itself be placed back into the repository and subjected to the same mechanisms of versioning, review, and revision. The apparatus can therefore become one of its own experimental objects.

This recursion should not be mistaken for autonomous self-understanding. The system does not acquire privileged access to its own operation merely because its source files can be concatenated and supplied to a model. What it gains is inspectability. Its external structure can be represented in the same medium that its language models already process well. This reduces the translation cost between operation and analysis.

The resulting workflow has an important epistemological advantage: claims about the system can remain close to the evidence from which they were inferred. If an essay states that the review prompt prohibits rewriting, the prompt file can be inspected. If it states that the larger model was assigned to review, the shell script can be inspected. If it claims that a criticism survived into the final revision, `draft.md`, `review.md`, and `essay.md` can be compared. The explanatory document is therefore capable of remaining tethered to a relatively compact evidential substrate.

This tethering does not make interpretation automatic. Determining whether a conceptual criticism was actually resolved remains a semantic judgment. Determining whether a final essay is genuinely better still requires evaluative criteria. Provenance does not eliminate interpretation; it makes the objects of interpretation recoverable.

The distinction is important because transparency is sometimes treated as though merely exposing more information guarantees understanding. A repository containing thousands of generated files can be technically transparent while practically inscrutable. Provenance

must therefore be organized. Names, directory structure, logs, commit messages, prompt boundaries, and generated summaries such as the icepick output all contribute to making history navigable rather than merely retained.

There is consequently an optimal granularity problem. Recording too little history prevents causal reconstruction. Recording every insignificant transient operation can overwhelm the meaningful structure with noise. The earlier line-by-line monotonic commit experiment intentionally explored an extreme. The essay pipeline can adopt a more semantic granularity: topic selection, outline generation, drafting, review, revision, prompt modification, model reassignment, and evaluation are natural historical events because they correspond to experimentally meaningful transformations.

The correct unit of provenance is therefore not necessarily a line, token, file, or inference call. It is an intervention whose identity matters to later explanation. This criterion parallels the broader theory of diagnostic coordinates. Observation becomes useful when it separates states relevant to deciding what intervention is warranted. Historical resolution should similarly be high enough to distinguish changes that alter the interpretation of an experiment.

A prompt modification is such a change. A correction to a comment in a shell script may not be. A switch from Granite 3B to Granite 8B at revision is clearly significant. Regenerating an essay under identical conditions may also be significant because it samples stochastic variation. The repository should preserve enough information that these cases can be distinguished.

The long-term result is a research archive in which artifacts do not appear as isolated products. Each belongs to a trajectory. The essay has a trajectory through planning, drafting, criticism, and repair. The experimental pipeline has a trajectory through script and prompt revisions. The theoretical program has a trajectory through successive experiments. Git provides a common historical substrate upon which these nested continuations can coexist.

This nesting is important because scientific knowledge itself has similar structure. A theory is not merely its latest formulation. Its meaning partly depends upon the problems it was introduced to solve, the alternatives it rejected, the anomalies that forced its revision, and the distinctions that survived those revisions. Publishing only terminal states compresses this developmental structure. A sufficiently inexpensive computational archive makes it possible to preserve more of it.

Language models make this possibility both more necessary and more practical. They increase the rate at which candidate representations can be generated, which makes manual memory less adequate. At the same time, their outputs are naturally serializable into textual artifacts that version-control systems already handle well. The increased generative capacity that threatens to overwhelm provenance can therefore be paired with automated historical preservation.

This is one of the deeper motivations for the repository experiment. If generative systems make intellectual production dramatically cheaper, the appropriate response is not merely to produce more terminal documents. It is to improve the structures through which their origins, revisions, criticisms, and dependencies remain recoverable. Otherwise generative abundance produces epistemic opacity: many polished artifacts with increasingly weak histories.

The alternative is generative abundance with provenance. Under that regime, increased production can support experimentation because failed paths remain inspectable, successful transformations can be isolated, and theoretical development can be reconstructed. The repository becomes not merely a publishing destination but a memory structure for computational inquiry.

11.1 Formalization: Historical Monotonicity and Provenance Graphs

Let the state of an artifact at stage t be

$$X_t.$$

A conventional destructive update may be represented as

$$X_t \mapsto X_{t+1},$$

where only X_{t+1} remains observable.

Define the observable history after n destructive updates as

$$\mathcal{H}_n^{\text{destructive}} = \{X_n\}.$$

Under persistent historical storage, define

$$\mathcal{H}_n = \{X_0, X_1, \ldots, X_n\}.$$

Then

$$\mathcal{H}_n \subseteq \mathcal{H}_{n+1}.$$

This inclusion defines historical monotonicity.

Definition 11.1 (Historical monotonicity)**.** *A process has historically monotonic persistence if every newly accepted state may alter the current artifact while the set of recoverable prior states never decreases:*

$$\mathcal{H}_t \subseteq \mathcal{H}_{t+1}.$$

Notice that this does not require

$$X_t \subseteq X_{t+1}$$

or any equivalent monotonicity of content.

Indeed, textual content can change arbitrarily:

$$X_t \neq X_{t+1},$$

while historical evidence remains monotonic.

Proposition 11.2 (Historical monotonicity permits non-monotonic belief revision)**.** *A system can retract arbitrary content while preserving a monotonic record of its representational history.*

Proof. Let proposition p be asserted in X_t and rejected in X_{t+1}. Thus the current representational state changes non-monotonically with respect to p:

$$p \in X_t, \qquad p \notin X_{t+1}.$$

Under historical persistence,

$$X_t \in \mathcal{H}_{t+1}$$

and

$$X_{t+1} \in \mathcal{H}_{t+1}.$$

The current state can therefore reject p while the history continues to preserve evidence that p was previously asserted. Hence content revision and historical monotonicity are compatible.

□ □

This distinction provides the formal basis for repair without historical erasure.

Now let an experimental run be represented by a provenance graph

$$G_P = (V, E).$$

Each vertex

$$v \in V$$

is an artifact or transformation, and each directed edge

$$(v_i, v_j) \in E$$

means that v_j depends causally upon v_i.

For the essay pipeline, a simplified graph is

$$T \to O \to D,$$

$$D \to R,$$

$$D \to F,$$

$$R \to F.$$

Thus F has two immediate parents:

$$\mathrm{Pa}(F) = \{D, R\}.$$

The outline itself has

$$\mathrm{Pa}(O) = \{T, P_O, M_O\},$$

if prompt and model identity are represented explicitly.

More generally, an artifact X_i can be associated with provenance tuple

$$\pi(X_i) = (\mathrm{Pa}(X_i), M_i, p_i, \theta_i, t_i, g_i)\,,$$

where M_i is the model, p_i the prompt, θ_i inference configuration, t_i timestamp, and g_i repository state.

Perfect reconstruction would require enough of $\pi(X_i)$ to reproduce the conditional process generating X_i. The current experiment preserves only a subset. We can therefore define provenance completeness relative to a required variable set K.

Let

$$K = \{k_1, \ldots, k_m\}$$

be provenance fields considered relevant. Define

$$c_j(X) = \begin{cases} 1, & k_j \text{ is recoverable for } X, \\ 0, & \text{otherwise.} \end{cases}$$

Then

$$C_P(X) = \frac{1}{m} \sum_{j=1}^{m} c_j(X)$$

is a simple provenance-completeness score.

A stronger weighted form is

$$C_P^w(X) = \sum_{j=1}^{m} w_j c_j(X), \qquad \sum_j w_j = 1.$$

The weights depend upon the scientific question. Model identity may receive high weight in a model-comparison experiment, while exact hardware may receive lower weight if only semantic process reconstruction is required.

Reproducibility should similarly be decomposed. Let

$$\mathcal{R}_s$$

denote structural reproducibility: the ability to reconstruct the same pipeline topology.

Let

$$\mathcal{R}_c$$

denote configuration reproducibility: the ability to recover prompts, model assignments, and inference settings.

Let

$$\mathcal{R}_o$$

denote output reproducibility: the probability of regenerating an output equivalent under some criterion.

Exact textual reproducibility would require

$$X_i' = X_i.$$

Semantic reproducibility may instead require

$$d_{\text{sem}}(X_i', X_i) \leq \epsilon.$$

Thus one can have

$$\mathcal{R}_s \approx 1, \qquad \mathcal{R}_c \approx 1,$$

while

$$\Pr[X_i' = X_i] \ll 1$$

for stochastic generation.

The experiment can still be scientifically useful because causal topology and experimental

conditions remain recoverable even when exact stochastic output does not.

Git can be represented as a second directed acyclic graph

$$G_G = (C, E_C),$$

where C is the set of commits and

$$(c_i, c_j) \in E_C$$

indicates parentage.

Each commit maps to a repository snapshot

$$S : C \to \mathcal{F},$$

where $\mathcal{F}$ denotes filesystem states.

An artifact X generated under commit c can therefore be associated with coordinate

$$(c, \text{path}(X)).$$

If the repository state is clean at generation time and the relevant environment is recorded, this coordinate provides a compact reference to the apparatus surrounding X.

The two provenance graphs can be related by mapping

$$\mu : V(G_P) \to C(G_G),$$

assigning each experimental artifact to the repository state under which it was produced.

This allows questions such as whether two outputs were generated under identical prompts:

$$p_r^{\mu(X_1)} = p_r^{\mu(X_2)}.$$

Without repository provenance, later changes to `review.txt` could make the current prompt appear to be the prompt that generated historical reviews even when it was not.

The monotonic evidence principle can now be stated.

Theorem 11.3 (Recoverable predecessor requirement for transformation audit)**.** *Let*

$$Y = \tau(X)$$

be a transformation whose quality depends partly upon preservation of properties of X. If X is not

recoverable after Y is produced, then direct retrospective measurement of preservation relative to X is impossible unless an equivalent representation of the relevant properties of X survives elsewhere.

Proof. Let preservation be evaluated by

$$P(X, Y).$$

After destructive replacement, suppose only Y remains and no sufficient representation

$$Z = f(X)$$

of the relevant properties of X survives. Then X cannot be supplied to P, nor can its relevant properties be reconstructed from Z. Consequently $P(X, Y)$ cannot be directly evaluated from the surviving state. Therefore preservation auditing requires either recoverability of X or a sufficient retained representation of its relevant properties. □ □

This theorem gives a direct methodological justification for retaining drafts after revision.

We can also formalize historical resolution. Let the actual process contain transformations

$$\tau_1, \tau_2, \ldots, \tau_n.$$

A provenance system records only a partition

$$\mathcal{P} = \{B_1, \ldots, B_k\},$$

where each block B_j contains transformations collapsed into one recorded historical transition.

If a causal question requires distinguishing

$$\tau_a$$

from

$$\tau_b$$

but

$$\tau_a, \tau_b \in B_j,$$

the recorded history lacks sufficient resolution for that question.

Thus provenance adequacy is relative to the interventions one wishes to distinguish.

Definition 11.4 (Diagnostic historical resolution)**.** *A historical record has sufficient diagnostic*

resolution for question Q if every pair of transformations whose distinction can alter the answer to Q remains distinguishable in the record.

This provides a principled alternative to the extremes of recording everything and recording only final states.

Finally, consider generative throughput. Let

$$\lambda_g$$

be the rate at which candidate artifacts are generated and

$$\lambda_p$$

the rate at which their provenance can be manually documented.

As model-assisted generation increases,

$$\lambda_g \gg \lambda_p$$

can occur rapidly.

If provenance requires manual reconstruction, the fraction of adequately documented artifacts may decline approximately as

$$\rho = \min\left(1, \frac{\lambda_p}{\lambda_g}\right).$$

Automated provenance changes the scaling relation by coupling documentation to generation. If every stage automatically writes its artifacts and metadata, then effective provenance throughput becomes

$$\lambda_p' \approx \lambda_g$$

within the capacity of the storage and orchestration system.

This produces a general design principle:

As generation becomes cheaper, provenance must become automatic.

Otherwise increased generative capacity produces an expanding set of artifacts whose developmental histories cannot be reconstructed.

The essay pipeline is deliberately modest in scale, but it already embodies this principle. Each experiment leaves behind more than its terminal essay. It leaves a topic, a plan, a draft, a

diagnosis, a revision, and the surrounding executable apparatus. Git can preserve changes to that apparatus, while the filesystem preserves the internal states of individual runs.

The result is not merely a collection of generated essays. It is a growing historical structure in which claims, criticisms, repairs, and computational procedures remain available as distinct but related objects. Such a structure is necessary if generative systems are to become instruments of cumulative inquiry rather than machines for producing increasingly numerous terminal texts whose origins disappear as quickly as they are created.

12 Failure as Data: Diagnostic Coordinates and the Localization of Error

A pipeline designed only to produce successful essays would naturally treat failure as waste. A weak outline would be regenerated, an unsatisfactory review discarded, a malformed response overwritten, and an unsuccessful revision replaced until the final artifact appeared acceptable. The experimental architecture developed here adopts a different attitude. Failure is valuable when the system preserves enough structure to determine where the failure occurred, what information was available when it occurred, and which intervention would have been required to avoid it. The purpose of decomposition is therefore not merely to increase the probability of success. It is to increase the diagnostic resolution of unsuccessful trajectories.

This distinction is particularly important for language-model systems because a disappointing terminal output is radically underdetermined with respect to its cause. If a one-shot model produces an essay with a confused thesis, several explanations remain possible. The original task may have been ambiguous. The model may have selected a poor conceptual decomposition. It may have understood the intended distinction but failed to maintain it across a long generation. It may have generated a locally plausible paragraph that contradicted an earlier commitment. It may simply lack the representational capacity required by the argument. When all of these operations occur inside one inference, the terminal artifact exposes the defect but only weakly identifies its origin.

A staged architecture converts some of these hidden possibilities into observable coordinates. If the outline already confuses admissibility with optimization, the defect precedes drafting. If the outline distinguishes them correctly but the draft collapses them, the failure belongs to the transition from plan to prose. If the draft contains the error and the reviewer identifies it accurately, diagnosis has succeeded even though generation failed. If the final revision preserves the error despite an explicit diagnosis, the failure has moved again: the problem is no longer primarily detection but repair. The same terminal defect therefore has different meanings depending upon the trajectory through which it survived.

This is why intermediate artifacts should not be understood merely as convenient checkpoints. They form a coordinate system over the process. Each artifact exposes a different boundary of the computation. By comparing adjacent states, the investigator can ask which distinctions entered, disappeared, persisted, or changed meaning at each transition. The architecture thereby transforms a vague judgment such as "the model got this wrong" into a more precise statement such as "the distinction was represented correctly in the outline, lost during drafting, rediscovered by the reviewer, and not successfully restored by the reviser."

Such a statement has substantially greater experimental value. It distinguishes four capacities that a terminal evaluation would conflate: initial representation, representational maintenance, diagnostic recovery, and corrective transformation. A model can succeed at one while failing at another. Indeed, the exploratory pipeline already suggests that such dissociations are common. The ability to recognize a defect in an existing artifact does not imply the ability to generate an artifact without that defect. The ability to describe the required repair does not imply the ability to execute it while preserving surrounding structure.

This separation resembles the distinction between diagnostic coordinates and repair warrant developed elsewhere in the broader theoretical program. Observation is useful when it supplies enough information to distinguish states that require different interventions. Merely detecting that "something is wrong" is insufficient. A diagnostic representation should localize the failure relative to the transformations available to the system. In the writing pipeline, a final quality score is therefore a weak diagnostic coordinate. It can indicate that an essay is poor without revealing whether the appropriate intervention is to alter the topic specification, regenerate the outline, strengthen the drafting model, change the review protocol, improve the intermediate representation, or replace the reviser.

The staged artifacts provide a richer coordinate system because each boundary permits a different family of interventions. Before outlining, one can modify the topic or theoretical context. After outlining, one can repair the conceptual plan without paying the cost of regenerating a full essay. After drafting, one can diagnose prose-level and argument-level defects while preserving the plan. After review, one can change the reviser or repair protocol while holding both diagnosis and draft fixed. The architecture therefore increases not merely observability but intervention specificity.

This is an important distinction. More observations do not automatically produce better diagnosis. A system can generate enormous logs without revealing which action should be taken. Diagnostic value depends upon whether the observations separate states with different repair requirements. The present pipeline is useful because its stage boundaries correspond to semantically different operations. The resulting coordinates are coarse, but they are aligned with meaningful intervention points.

The review artifact occupies a particularly interesting diagnostic role because it represents an explicit attempt to map defects into repair-relevant language. The draft itself contains evidence of its failures, but those failures remain embedded within the artifact. The review externalizes a hypothesis about them. It names ambiguities, unsupported transitions, hidden assumptions, structural weaknesses, or excessive claims. In doing so, it transforms a raw textual state into a diagnostic representation.

The review should therefore not be assumed correct merely because it is diagnostic in form. It is itself a generated artifact and can fail in several ways. It can miss a real defect, invent a defect where none exists, correctly identify a symptom while misunderstanding its cause, recommend a repair that destroys an important distinction, or spend attention on stylistic irregularities while overlooking structural problems. The diagnostic stage must therefore itself become an object of evaluation.

This recursive possibility is not an inconvenience but a consequence of using bounded cognitive components. There is no privileged layer at which fallibility disappears. A critic can require criticism. A repair can require repair. An evaluator can be poorly calibrated. The architecture should therefore avoid treating any generated intermediate artifact as an oracle. Its role is to contribute evidence to subsequent intervention, not to terminate epistemic uncertainty.

The distinction between detection and warrant becomes essential at this point. Suppose the reviewer identifies a sentence as ambiguous. It does not follow that the sentence should be rewritten. The ambiguity may be intentional, the reviewer may have misunderstood a technical term, or the proposed clarification may introduce a stronger distortion than the original ambiguity. A detected anomaly creates a candidate repair site; it does not automatically authorize intervention.

The null intervention must therefore remain available at every repair boundary. If the evidence for a defect is weak or the proposed repair carries substantial risk, preservation may be preferable. This is especially important when the essay develops nonstandard concepts. A conventional reviewer may experience unfamiliarity as error. If every such signal automatically triggers normalization, the pipeline will systematically erase precisely the distinctions it is supposed to develop.

A repair warrant consequently requires more than a diagnosis. It requires some independently grounded reason to believe that the proposed transformation improves the artifact relative to leaving it unchanged. In the present experiment, such grounding may come from logical inconsistency, violation of an explicit definition, failure to support a conclusion, contradiction with an invariant specified by the project, or agreement among independent evaluators. The exact admissible reference class remains an open experimental problem, but the architecture should preserve the distinction between observing a candidate defect and possessing sufficient

warrant to modify it.

This has direct implications for prompt design. A review prompt that merely asks for weaknesses encourages unrestricted anomaly detection. A more mature protocol could ask the reviewer to distinguish high-confidence structural failures from optional improvements, to identify the evidence supporting each diagnosis, and to estimate whether the defect can be repaired locally or requires upstream reconsideration. Such a representation would make the review more useful as a diagnostic coordinate rather than a generic collection of editorial preferences.

The same distinction can improve revision. Instead of treating every review statement as an instruction, the reviser could treat the review as a set of candidate interventions with varying warrant. It could preserve the null intervention when a recommendation conflicts with a protected invariant or when the proposed repair cannot be justified from the supplied context. Revision would then become a decision process rather than unconditional obedience.

This architecture would also make disagreement informative. If two reviewers inspect the same frozen draft and identify different problems, their disagreement reveals uncertainty in the diagnostic representation. If they identify the same defect but recommend different repairs, the defect may be well localized while the intervention remains underdetermined. If both recommend the same repair and independent evaluation confirms improvement, the repair warrant becomes stronger. Multiple reviewers can therefore be used not merely to vote on quality but to estimate the structure of diagnostic uncertainty.

Failure localization can extend backward through the pipeline. Suppose a reviewer identifies an unsupported transition in the draft. Inspection of the outline may show that the missing premise was already absent there. In that case, repairing only the prose treats a downstream symptom. The stronger intervention may be to repair the outline and regenerate the dependent portion of the essay. Conversely, if the outline contains the required premise but the draft omits it, the problem lies in realization rather than planning. The same textual defect calls for different interventions depending upon its provenance.

This suggests that repair should sometimes propagate upstream. The current linear pipeline primarily moves forward:

$$T \to O \to D \to R \to F.$$

A more general architecture would allow diagnosis to redirect computation toward an earlier state:

$$R \to O'$$

or

$$R \to D'$$

before another revision is attempted. Such backward edges convert the pipeline from a fixed sequence into a repair graph.

The danger is uncontrolled recursion. If every criticism can reopen every previous stage, the system may enter cycles of continual regeneration. The repair graph therefore requires escalation criteria. A local defect should receive a local repair when possible. Upstream regeneration should occur only when the defect cannot be repaired without changing assumptions established earlier in the trajectory. This follows the general principle of minimal warranted intervention.

The hierarchy of interventions can be described informally as increasing structural depth. A lexical defect can be corrected within a sentence. A missing justification may require modification of a paragraph. A contradiction between sections may require restructuring the draft. A false premise inherited from the outline may require returning to the plan. A conceptual failure in the topic specification may require redefining the experiment itself. The appropriate repair depth depends upon the earliest dependency that must change for the defect to become removable.

This creates a natural connection to hierarchical admissibility. An artifact can satisfy local constraints while violating higher-level ones. A sentence may be grammatical while contradicting the paragraph's argument. A paragraph may be coherent while undermining the essay's thesis. An essay may be internally coherent while failing to answer the topic it was intended to investigate. Diagnostics should therefore operate across several representational scales.

A purely local reviewer may overvalue sentence-level improvements because those defects are easy to identify. A stronger reviewer should preserve access to higher-level constraints. The outline is useful partly because it externalizes such constraints before drafting. It provides a reference against which local prose can be evaluated. Without that representation, global coherence must be inferred entirely from the draft itself.

The pipeline can consequently be understood as creating diagnostic coordinates at several scales. The topic specifies the problem coordinate. The outline specifies the structural coordinate. The draft specifies the realized argumentative coordinate. The review specifies a diagnostic coordinate over that realization. The final essay specifies the post-intervention coordinate. Git supplies a historical coordinate over changes to the entire apparatus.

The resulting system is not guaranteed to know what is wrong. It is designed so that when something goes wrong, the investigator has a better chance of determining where the relevant distinction was lost. This is a more modest objective than autonomous correctness, but it may be a more useful one for experimental systems. A fallible process whose failures are localizable can often be improved. A system that produces impressive terminal outputs while concealing

the origins of its failures is harder to repair systematically.

This principle becomes increasingly important as language-model pipelines grow more complex. Adding agents, tools, retrieval systems, memory layers, evaluators, and iterative loops can improve capability while simultaneously making causal attribution more difficult. When a final answer fails, the number of plausible failure sites increases. Unless the architecture preserves diagnostic boundaries, greater compositional complexity can produce less intelligibility.

The present experiment deliberately begins from the opposite direction. Its stages are few, their interfaces are textual, and their outputs are persistent. This simplicity allows failure localization to remain conceptually manageable. Additional stages should be introduced only when they correspond to a meaningful distinction that can be experimentally justified.

The architecture therefore treats modularity as an epistemic property rather than merely a software-engineering convenience. A module is valuable when its boundary exposes a distinction relevant to diagnosis and intervention. Splitting a process arbitrarily into more scripts does not necessarily improve understanding. The decomposition is useful when the resulting boundaries correspond to separable hypotheses about what the system is doing.

The distinction between outline, draft, review, and revision satisfies this criterion because each stage answers a different question. What structure should the argument have? Can that structure be realized as prose? What is defective in the realization? Can those defects be repaired without destroying successful structure? These are not merely four prompts. They are four experimentally distinguishable capacities.

The same criterion can guide future extensions. A fact-checking stage would be justified if factual warrant constitutes a failure class not adequately localized by the existing reviewer. A formalization stage would be justified if natural-language arguments repeatedly conceal mathematical inconsistencies. A citation stage would be justified if evidential provenance requires independent treatment. The architecture should grow by discovering stable diagnostic distinctions, not by accumulating fashionable agent roles.

This is also why failures should remain in the repository. A failed revision is evidence about the reachability of repair under a particular model, prompt, and diagnostic representation. Deleting it because it is not publishable destroys experimental information. A malformed outline can reveal weaknesses in topic specification. A hallucinated criticism can reveal a reviewer failure mode. An essay that becomes smoother but conceptually flatter after revision can reveal normalization pressure. The failed artifact may therefore be more informative about the system than a successful one.

The repository can eventually accumulate a taxonomy of such failures. That taxonomy should emerge from repeated observations rather than being imposed too early. Once stable categories

appear, they can be encoded into evaluation scripts, structured review formats, or automated tests. The process thereby moves from exploratory observation toward formal instrumentation.

This progression mirrors scientific measurement more generally. One begins with phenomena that are recognizable but poorly quantified. Repeated observation reveals recurring distinctions. Those distinctions become named variables. Variables become measurable coordinates. Measurements support controlled interventions. The essay pipeline is currently near the beginning of this progression, but its persistent structure makes later formalization possible.

The larger theoretical implication is that intelligence under constraint may depend as much upon the geometry of failure as upon the height of peak performance. A system capable of occasionally producing extraordinary outputs but unable to localize its mistakes may be less useful for cumulative inquiry than a somewhat weaker system whose failures remain bounded, visible, and repairable. The relevant capability is therefore not simply the probability of immediate success. It is the probability that the system can remain within a trajectory from which failure can be diagnosed and continuation recovered.

12.1 Formalization: Diagnostic Resolution and Repair Warrant

Let the pipeline trajectory contain states

$$X_0, X_1, \ldots, X_n$$

with transition operators

$$\tau_i : X_i \to X_{i+1}.$$

Suppose terminal defect e is observed in X_n. Let

$$o(e)$$

denote the earliest transition at which the defect becomes present:

$$o(e) = \min \{ i : e \notin X_i \wedge e \in X_{i+1} \}.$$

If the relevant states are preserved, $o(e)$ can in principle be investigated by comparing successive artifacts. If only X_n survives, $o(e)$ is generally not identifiable.

Define the failure-location variable

$$L_e \in \{\text{topic}, \text{outline}, \text{draft}, \text{review}, \text{revision}\}.$$

An observation Z has diagnostic value to the extent that it reduces uncertainty about L_e. One information-theoretic measure is

$$\mathcal{D}(Z; e) = H(L_e) - H(L_e \mid Z),$$

where H denotes entropy.

A terminal quality score $Q(X_n)$ may satisfy

$$\mathcal{D}(Q(X_n); e) \approx 0$$

even when it reliably indicates that some defect exists, because it does not distinguish where the defect originated.

By contrast, the tuple

$$Z = (O, D, R, F)$$

can substantially reduce uncertainty about L_e.

This yields a formal distinction between detection and localization:

$$\Pr(e \mid Z)$$

measures evidence that a defect exists, while

$$\Pr(L_e = i \mid e, Z)$$

measures evidence concerning its origin.

The two quantities need not covary.

A reviewer produces diagnostic claim

$$d_j = (\ell_j, \kappa_j, \epsilon_j),$$

where ℓ_j denotes the alleged defect location, κ_j its type, and ϵ_j the evidence offered for the diagnosis.

Let

$$P(e_j \mid d_j, X)$$

represent confidence that the diagnosed defect is real.

A candidate repair

$$\rho_j$$

has expected benefit

$$B(\rho_j) = \mathbb{E}\left[L(X) - L(\rho_j(X)) \mid d_j\right].$$

Let expected collateral cost be

$$C(\rho_j) = C_{\text{structural}} + C_{\text{semantic}} + C_{\text{resource}} + C_{\text{risk}}.$$

The null intervention

$$\rho_0(X) = X$$

has

$$C(\rho_0) = 0$$

for intervention-specific cost.

Define repair warrant

$$W(\rho_j \mid d_j, X) = B(\rho_j) - C(\rho_j).$$

A repair is warranted only if

$$W(\rho_j \mid d_j, X) > 0.$$

Detection alone corresponds merely to

$$P(e_j \mid d_j, X) > \theta$$

for some threshold θ.

Therefore

$$P(e_j \mid d_j, X) > \theta$$

does not imply

$$W(\rho_j \mid d_j, X) > 0.$$

Theorem 12.1 (Detection does not entail repair warrant)**.** *There exist states in which a defect can be identified with arbitrarily high confidence while every available non-null repair has expected utility less than or equal to the null intervention.*

Proof. Let defect e be known with certainty:

$$P(e \mid X) = 1.$$

Suppose every available repair ρ_i removes e but destroys a constitutive invariant with cost C_i

exceeding the benefit B_i:

$$C_i \geq B_i.$$

Then

$$W(\rho_i) = B_i - C_i \leq 0$$

for every non-null repair.

The null intervention leaves the known defect in place but incurs no intervention cost. Therefore no available repair is warranted despite perfect defect detection. □ □

This result is particularly relevant when a reviewer identifies a genuine weakness that the available reviser cannot repair safely.

Now let the pipeline contain hierarchical representational levels

$$\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_m,$$

where lower levels represent local textual structure and higher levels represent increasingly global argumentative constraints.

Let defect e first violate a constraint at level

$$\ell(e).$$

A repair operating only at level

$$k < \ell(e)$$

may alter symptoms without modifying the constraint responsible for the failure.

Define repair depth

$$d(\rho)$$

as the highest structural level modified by repair ρ.

A necessary condition for certain structural defects is therefore

$$d(\rho) \geq \ell(e).$$

This motivates escalation.

Let

$$\rho^{(0)}, \rho^{(1)}, \ldots, \rho^{(m)}$$

be repair operators of increasing depth. A minimal-intervention policy selects

$$\rho^* = \arg\min_{\rho} d(\rho)$$

subject to

$$W(\rho) > 0$$

and

$$L(\rho(X)) < L(X).$$

The system therefore seeks the shallowest warranted repair sufficient to remove the defect.

Proposition 12.2 (Upstream defects require dependency-aware repair)**.** *If downstream artifact X_j inherits property p necessarily from upstream state X_i, and p is the source of defect e, then repairing only a downstream manifestation of p without altering either p or its dependency relation cannot guarantee elimination of e.*

Proof. Assume

$$p(X_i) \Rightarrow p(X_j)$$

through the pipeline dependency relation and that

$$p \Rightarrow e.$$

A downstream transformation that leaves both p and the inheritance relation unchanged preserves the sufficient condition for e. Therefore the defect may recur or remain structurally present even if one local manifestation is rewritten. Guaranteed removal requires modification of p, modification of the dependency relation, or introduction of an additional constraint that blocks

$$p \Rightarrow e.$$

□ □

The pipeline can thus be generalized from a chain into a repair graph

$$G_R = (V, E_f \cup E_b),$$

where E_f contains ordinary forward-generation edges and E_b contains diagnostically warranted backward edges.

For example,

$$D \to R$$

may be followed by

$$R \to O'$$

when the review determines that the draft defect originates in the outline.

To prevent uncontrolled cycling, associate every proposed backward transition with expected repair warrant

$$W_b.$$

Permit escalation only when

$$W_b > \theta_b$$

for threshold $\theta_b > 0$.

A further safeguard can require that the expected reduction in structural loss exceed the cost of invalidating dependent downstream artifacts:

$$\mathbb{E}[\Delta L_{\text{structural}}] > C_{\text{invalidate}}.$$

This expresses why returning to the outline should be rarer than editing a sentence: upstream changes invalidate larger portions of the trajectory.

Finally, define system recoverability after failure. Let

$$\mathcal{A}$$

be the admissible region and suppose current state

$$X \notin \mathcal{A}.$$

Let

$$\mathcal{R}(X)$$

be the set of states reachable through available repair operations.

Define recoverability

$$\chi(X) = \begin{cases} 1, & \mathcal{R}(X) \cap \mathcal{A} \neq \varnothing, \\ 0, & \mathcal{R}(X) \cap \mathcal{A} = \varnothing. \end{cases}$$

A probabilistic version is

$$\chi_p(X) = \Pr\left[\exists\, \rho : \rho(X) \in \mathcal{A}\right].$$

Peak one-shot performance measures something like

$$\Pr[X_1 \in \mathcal{A}],$$

the probability of immediate success.

A repair-oriented architecture is additionally concerned with

$$\Pr\left[X_t \notin \mathcal{A} \wedge \chi_p(X_t) > 0\right],$$

the probability that an initially defective state remains recoverable.

This yields a broader measure of useful system competence:

$$C_{\text{useful}} = P_{\text{immediate}} + \lambda P_{\text{recoverable}},$$

with care taken not to double-count immediately successful states.

The architectural objective is therefore not merely to maximize the probability that no error occurs. It is also to preserve enough diagnostic structure that errors which do occur remain localized and repairable.

The resulting principle can be written compactly as

A failure that preserves its diagnostic coordinates is more useful than a failure that destroys its own history.

For experimental language-model systems, this changes the meaning of robustness. Robustness is not only resistance to error. It is the capacity to fail in ways from which the location, cause, and admissible repair of the failure can still be investigated.

The essay pipeline makes this principle concrete. Its intermediate artifacts are not incidental debris left behind by generation. They are the coordinates through which the trajectory becomes diagnosable. Preserving them allows a weak final essay, a mistaken review, or an unsuccessful repair to remain part of cumulative knowledge about the system rather than becoming merely another output to delete and regenerate.

13 Limits, Risks, and the Boundary of What the Pipeline Can Establish

The experimental pipeline makes the writing process more inspectable, but inspectability should not be confused with epistemic sufficiency. The persistence of outlines, drafts, reviews,

and revisions allows the investigator to reconstruct a sequence of transformations, yet this sequence remains only a partial representation of the reasoning involved. The models' internal computations are not revealed, the prompts do not guarantee compliance, and the artifacts preserve expressed judgments rather than the full representational structure that produced them. The architecture therefore improves the observability of composition without dissolving the opacity of the components performing it.

This distinction is important because a visible pipeline can create an exaggerated sense of methodological control. The shell scripts are simple, the files are named clearly, and the sequence of operations is explicit. These features make the procedure legible, but they do not ensure that the semantic operation corresponding to each stage has occurred successfully. A file named `outline.md` may contain a weak or incoherent plan. A file named `review.md` may contain confident but irrelevant criticism. A file named `essay.md` may be nearly identical to the draft despite a detailed review. The architecture provides coordinates for evaluation; it does not automatically validate the artifacts occupying those coordinates.

The first major limitation concerns evaluation. The pipeline can record that a reviewer identified a problem and that a reviser changed a passage, but determining whether the diagnosis was correct and the repair successful still requires an external standard. For grammatical errors, the standard may be relatively stable. For mathematical claims, formal verification may sometimes be available. For conceptual essays, however, evaluation depends upon interpretation, domain knowledge, and theoretical judgment. A model can identify an "unsupported transition" where the transition is actually licensed by background assumptions familiar to the intended audience. It can also fail to notice that a seemingly plausible example contradicts the theory being developed.

The current experiments therefore cannot establish that the 8B review is objectively superior merely because it is more elaborate or more critical. Length and sophistication of vocabulary are weak proxies for diagnostic accuracy. A longer review can contain more false positives. A reviewer trained to seek flaws may manufacture objections in order to satisfy the prompt. The very prompt that improves critical vigilance can create a bias toward overdiagnosis.

This problem is intensified by the familial relation between the two Granite models. Although they differ in scale, they are not independent epistemic systems. Their training data, architectural lineage, and learned conventions are likely strongly correlated. If both accept the same hidden assumption, agreement between them provides little independent confirmation. If the 8B model criticizes a 3B draft, the difference may reflect additional capacity, but it may also reflect differences in instruction following, verbosity, sampling, or sensitivity to the specific prompt.

The pipeline should therefore not be described as a peer-review system in the full institutional

sense. Scientific peer review relies, at least ideally, upon reviewers whose knowledge, incentives, theoretical commitments, and evidential access are not identical to those of the author. A larger model from the same family supplies functional separation without guaranteeing epistemic independence. The review is better understood as an internal adversarial pass performed by a heterogeneous component.

A second limitation concerns factual grounding. The present prompts focus primarily on argumentative structure, definitions, objections, and inferential warrant. They do not require retrieval of sources, verification of quotations, checking of empirical claims, or reconstruction of the relevant literature. Consequently the pipeline can produce essays that are internally coherent while remaining poorly grounded in external evidence. A strong critic may notice that a claim lacks evidence, but neither model necessarily possesses the information required to supply that evidence accurately.

This is visible in several generated reviews, which recommend empirical studies, case examples, or measurable definitions. Such recommendations can be correct while remaining unfulfilled because the pipeline contains no research stage. The reviser may respond by inventing generic examples or broadening claims without acquiring new knowledge. The result can create the appearance of repair while merely changing the rhetorical surface.

The absence of retrieval is not an accidental omission. It reflects the deliberately narrow scope of the first experiment. The goal was to study planning, drafting, criticism, and revision under local model constraints. Introducing external research would add source selection, evidence extraction, citation management, and factual verification as new transformations. Those operations deserve separate interfaces and diagnostics rather than being hidden inside a general request to "make the essay more rigorous."

Nevertheless, the limitation should remain explicit. The pipeline can improve the organization of available representations. It cannot guarantee that those representations correspond to the world. Internal argumentative repair and external evidential repair are distinct processes. A coherent falsehood remains false.

A third limitation concerns contamination by the prompt's assumptions. The outline protocol requests a conceptual problem, thesis, objection, and consequences. This structure is productive for many analytical essays, but it is not neutral. It favors argumentative forms in which a central proposition is progressively defended. Some subjects may be better treated through historical reconstruction, comparative analysis, phenomenological description, mathematical development, or unresolved dialectic. A rigid protocol can force every topic into the same rhetorical geometry.

The generated essays already show traces of this tendency. Several outlines follow a familiar

sequence of introduction, conceptual problem, thesis, development, objection, consequences, and conclusion. This consistency makes experiments easier to compare, but it may also encourage formulaic prose. The architecture risks confusing procedural regularity with intellectual rigor.

The problem is not that the protocol is wrong. It is that every protocol defines a restricted class of acceptable outputs. If the system repeatedly generates essays under one template, the template can become an invisible prior over thought. The writer may begin to select topics that fit the pipeline rather than adapting the pipeline to the topic. A useful experimental apparatus can thereby become an intellectual constraint.

This risk is especially relevant when the pipeline operates on a developing theoretical framework. Concepts such as admissibility, continuation, repair, and distinction may require different forms of exposition at different stages. A polemical essay, a mathematical monograph, a historical analysis, and a software methods paper should not necessarily share the same outline protocol. The current prompts should therefore be treated as one genre-specific configuration rather than a universal writing method.

A fourth limitation concerns the preservation instruction. Asking the reviser to preserve material that survived criticism protects conceptual identity, but it can also stabilize undetected defects. Reviewer silence is ambiguous. A passage may be sound, or it may simply have escaped attention. The preservation rule treats absence of criticism as weak evidence for retention. This is often reasonable, but it should not be mistaken for validation.

The same rule can inhibit global repair. Some defects are distributed across the essay rather than localized in one passage. A definition may subtly shape several later sections. Repairing the definition while preserving all uncriticized dependent prose may produce inconsistency. A thesis may need to be narrowed, requiring changes to examples, objections, and conclusion even if the reviewer mentioned only the thesis explicitly. Conservative repair is therefore safest when the review accurately represents dependency structure.

The current natural-language review does not encode those dependencies systematically. It identifies problems in prose but does not construct a formal repair graph. The reviser must infer which surrounding passages depend upon each criticized claim. A smaller model may fail to propagate the change far enough, producing the review–revision divergence observed in the experiments.

A fifth limitation concerns stochasticity. Repeated runs under nominally identical conditions can produce different outlines, drafts, and reviews. This variation is not inherently undesirable. It can reveal alternative structures and support sampling-based exploration. But it complicates causal attribution. If a later run produces a better essay after a prompt change, the improvement

may result from stochastic variation rather than the intervention.

The current repository preserves outputs but does not yet record enough inference metadata to characterize this variability fully. Sampling parameters, context settings, model digests, runtime versions, and hardware conditions should eventually be associated with each run. Even then, exact reproduction may remain unavailable. The appropriate response is repeated sampling and statistical comparison, not the assumption that one observed output represents the stable behavior of the model.

A sixth limitation concerns computational resource measurement. The architecture is motivated partly by the idea that the smaller model can perform inexpensive generative work while the larger model is reserved for criticism. Yet the current scripts do not measure inference time, token throughput, memory use, energy consumption, or total context length systematically. The cost advantage is therefore plausible but not quantified.

Model file size alone is insufficient. The 8B reviewer may generate fewer tokens than the 3B drafter, reducing the cost difference. Long revision prompts containing both draft and review may consume substantial context. A pipeline invoking the smaller model three times can be slower or more energy-intensive than one well-designed invocation of the larger model. Without measurement, claims of efficiency remain provisional.

A proper cost analysis should record wall-clock duration, prompt and output token counts, peak memory use, and perhaps approximate energy consumption for each stage. The relevant quantity is not merely model size but marginal quality per unit resource. The asymmetric assignment should survive comparison against cost-equalized alternatives before being described as efficient.

A seventh limitation concerns scaling. The current essays are short enough that complete drafts and reviews can be passed between stages directly. Longer books, large codebases, or research archives introduce context constraints. A reviewer may be unable to inspect the entire artifact at once. The filesystem can store arbitrarily large histories, but model context remains bounded. Selective loading then becomes necessary, reintroducing the problem of lossy representation.

Hierarchical summaries, section-level reviews, retrieval systems, and dependency maps may become necessary for larger projects. Each addition creates new opportunities for omission and distortion. The architecture's present transparency may therefore diminish as it scales unless the new layers preserve equally clear boundaries.

This scaling problem also affects global coherence. A section-by-section reviser can improve local passages while missing contradictions across distant sections. A book-length pipeline must represent not only text but cross-document invariants, definitions, notation, unresolved

claims, and dependency relations. The simple topic–outline–draft–review–revision sequence is an informative prototype, not a complete architecture for long-form scholarship.

An eighth limitation concerns intellectual homogenization. Language models are trained to continue patterns represented in their data. When asked to improve a nonstandard theory, they may translate it into familiar academic language and conceptual categories. Review can intensify this pressure because unfamiliarity is often interpreted as ambiguity, lack of definition, or insufficient grounding. Some of these criticisms are valid; others can function as forces of normalization.

The risk is not merely stylistic. A genuinely new distinction may initially appear awkward because the surrounding language has not yet stabilized. Excessive repair can erase it before its implications are understood. The pipeline therefore requires a way to distinguish productive novelty from accidental confusion. No prompt can resolve this automatically.

Protected invariants provide one partial solution. The user can specify claims, definitions, or distinctions that the pipeline must preserve unless an explicit higher-level decision authorizes revision. This does not make the protected material true, but it prevents routine editing from silently replacing the research program with a more conventional one. The system can then criticize the invariant, mark it as problematic, or request escalation rather than normalizing it implicitly.

A ninth limitation concerns authorship and responsibility. The pipeline distributes textual production across human topic selection, prompt design, model generation, model criticism, revision, and later human interpretation. The resulting essay does not have a single computational author in any simple sense. Yet responsibility for publication cannot be distributed away into the machinery.

The human operator chooses the theoretical topic, defines the protocols, decides which artifacts to preserve, judges whether criticism is warranted, and ultimately decides whether the resulting text should be presented as scholarship. The pipeline can expose its history, but provenance does not substitute for responsibility. A traceable false claim remains a false claim.

The role of the human investigator is therefore not external to the architecture. Even when the scripts execute automatically, the experimental meaning of their outputs depends upon human decisions about evaluation, repair, preservation, and publication. The system is better represented as human-guided computational inquiry than autonomous essay authorship.

A tenth limitation concerns the possibility of automation bias. Because the 8B model is assigned the role of critic, its review may acquire unwarranted authority. The architecture's labels can influence interpretation. A file called `review.md` produced by the "stronger" model may be treated as more reliable than it deserves. A human reader may search the draft for evidence

confirming the model's criticisms rather than evaluating them independently.

This bias can be reduced by anonymizing reviews during evaluation, comparing multiple reviewers, recording rejected criticisms, and requiring evidence for high-impact repair recommendations. The stronger model should be treated as another fallible instrument, not as the epistemic court of appeal.

These limitations do not invalidate the pipeline. They define the boundary of what the initial experiment can establish. The system demonstrates that staged local-model writing can be implemented transparently, that intermediate representations can preserve diagnostic structure, and that stronger criticism can be separated from cheaper generation. It does not demonstrate that the resulting essays are true, publishable, universally improved, or efficiently produced under every resource regime.

The distinction between demonstration and validation should be preserved. The current repository is a proof of architectural possibility and a source of exploratory observations. It establishes that the components can be composed and that their disagreements are informative. Validation requires controlled experiments, external evaluation, factual grounding, repeated sampling, and comparison with alternative architectures.

The architecture's value lies partly in making these limitations visible. A monolithic system may conceal the same problems behind a polished final answer. The staged pipeline leaves enough evidence to show that a review failed, a repair did not occur, or a definition remained vague. Its weaknesses become research questions rather than hidden liabilities.

13.1 Formalization: Epistemic Bounds and Non-Guarantee Results

Let the pipeline produce final essay

$$F = \mathcal{W}(T; \Theta),$$

where T is the topic and Θ the complete pipeline configuration.

Let

$$Q_{\text{int}}(F)$$

denote internal argumentative quality and

$$Q_{\text{ext}}(F)$$

denote external factual or evidential adequacy.

The current pipeline primarily acts upon representations related to

$$Q_{\text{int}}.$$

Without retrieval or verification, there is no general implication

$$Q_{\text{int}}(F) \text{ high} \quad \Longrightarrow \quad Q_{\text{ext}}(F) \text{ high}.$$

Theorem 13.1 (Internal coherence does not entail external truth)**.** *There exist essays that are internally coherent under the pipeline's argumentative criteria while containing false claims about the world.*

Proof. Let P be a false but logically consistent premise. Construct essay F whose definitions, transitions, objections, and conclusion follow coherently from P. Then F may score highly under internal criteria measuring structure and inferential consistency relative to its premises. Yet because P is false, at least some externally referential conclusions of F are false. Therefore internal coherence is insufficient for external truth. □ □

This establishes the necessity of an independent grounding stage for empirical scholarship.

Now let reviewer output be

$$R = C_M(D).$$

Define review accuracy

$$A_R = \frac{|E(D) \cap \widehat{E}(R)|}{|E(D) \cup \widehat{E}(R)|},$$

using a semantic Jaccard-style relation between actual and proposed defect sets.

The title "review" supplies no mathematical guarantee that

$$A_R > \theta$$

for any useful threshold θ.

Proposition 13.2 (Role labels do not imply contract satisfaction)**.** *Naming an artifact according to its intended stage does not entail that the artifact satisfies the semantic admissibility conditions of that stage.*

Proof. Let any arbitrary string x be written to path `review.md`. The filesystem operation succeeds regardless of whether x diagnoses the draft. Therefore the filename encodes intended role but does not enforce semantic type. □ □

Automatic stage validation would therefore require additional predicates.

Consider prompt-induced protocol bias. Let

$$\mathcal{E}$$

be the space of possible essay structures and let prompt p induce distribution

$$P_p(E \mid T).$$

A protocol is structurally neutral only if it leaves relative probabilities of all admissible structures unchanged. In practice,

$$P_p(E \mid T) \neq P(E \mid T)$$

for almost every nontrivial prompt p.

The prompt therefore acts as a prior over essay form.

Define structural bias toward class $\mathcal{C} \subseteq \mathcal{E}$ as

$$B_p(\mathcal{C}) = P_p(E \in \mathcal{C} \mid T) - P(E \in \mathcal{C} \mid T).$$

The repeated thesis–objection–consequence structure likely has

$$B_p(\mathcal{C}_{\text{argumentative}}) > 0.$$

This bias may be beneficial for the target genre while harmful for others.

The preservation problem can be formalized through undetected defects. Let

$$E(D)$$

be true defects and

$$\widehat{E}(R)$$

review-detected defects.

The undetected set is

$$U(D, R) = E(D) \setminus \widehat{E}(R).$$

If the revision rule preserves all material outside the diagnosed support, then defects in $U(D, R)$ may be systematically retained.

Proposition 13.3 (Conservative revision inherits reviewer false negatives)**.** *Under a strict preservation rule that changes only diagnosed regions, every defect outside the reviewer's diagnostic support survives revision.*

Proof. Let defect $e \in U(D, R)$, so e is not implicated by the review. Under strict preservation, the textual and argumentative support of e is unchanged. Therefore e remains present in the revised artifact unless removed incidentally by a dependent change, which strict locality excludes. Hence reviewer false negatives propagate into the final essay. □ □

This demonstrates why preservation should be treated as a default rather than an absolute rule.

Now consider stochastic variance. For fixed topic and configuration,

$$F_k \sim P(F \mid T, \Theta)$$

for replicate k.

An observed improvement after changing configuration from Θ_0 to Θ_1 is

$$\Delta_k = Q(F_k^{(1)}) - Q(F_k^{(0)}).$$

A single positive sample

$$\Delta_1 > 0$$

does not establish

$$\mathbb{E}[\Delta] > 0.$$

Repeated sampling is required to estimate

$$\mu_\Delta = \mathbb{E}[\Delta].$$

Let the sample estimator be

$$\widehat{\mu}_\Delta = \frac{1}{n} \sum_{k=1}^{n} \Delta_k.$$

The uncertainty decreases with n only under assumptions about independence and variance. Thus anecdotal output comparison is exploratory evidence rather than a causal estimate.

Cost claims require similar discipline. Let stage i consume time t_i, energy e_i, input tokens p_i,

and output tokens o_i. Define total resource vector

$$\mathbf{C} = \sum_i \begin{pmatrix} t_i \\ e_i \\ p_i \\ o_i \end{pmatrix}.$$

Model file size s_i is at most one proxy among several:

$$s_i \Rightarrow t_i, \qquad s_i \Rightarrow e_i.$$

Therefore the claim that asymmetric allocation is more efficient requires direct measurement of $\mathbf{C}$, not only model sizes.

Scaling introduces context selection. Let total persistent state be

$$S = \{x_1, \ldots, x_n\}$$

with total serialized length

$$L(S) = \sum_i |x_i|.$$

A model with context bound K cannot receive all state whenever

$$L(S) > K.$$

A selector

$$\Gamma(S) \subset S$$

must satisfy

$$L(\Gamma(S)) \leq K.$$

If task-relevant artifact $x_j \notin \Gamma(S)$, then downstream performance may degrade even though the information remains stored externally.

Persistence therefore solves storage but not bounded-context selection.

Authorship can be represented as a responsibility relation rather than a generation count. Let contributors be

$$\mathcal{C} = \{H, M_{3B}, M_{8B}, P, O\},$$

where H is the human operator, P the prompt architecture, and O the orchestration system.

Causal contribution to textual form may be distributed, but publication responsibility remains assigned to the human decision function

$$\Pi_H(F) \in \{\text{publish}, \text{withhold}, \text{revise}\}.$$

No decomposition of generation eliminates the necessity of this terminal decision.

Finally, define the strongest claim supported by the initial experiment as

$$C_{\text{supported}}.$$

Let stronger claims include

$$C_1 = \text{the pipeline always improves essays},$$

$$C_2 = \text{the 8B model is objectively superior at review},$$

$$C_3 = \text{the architecture is more efficient than one-shot generation},$$

$$C_4 = \text{the resulting essays are factually reliable}.$$

The current observations do not entail any C_i.

The supported claim is narrower:

$C_{\text{supported}} =$ the staged local-model architecture produces persistent, independently inspectable artifacts that p

Theorem 13.4 (Architectural demonstration is weaker than performance validation)**.** *Successful execution of a pipeline and preservation of its intermediate artifacts establish feasibility of the architecture but do not establish superiority over alternative architectures.*

Proof. Let architecture A execute successfully and produce outputs. Feasibility requires only that

$$\Pr[\text{execution completes under } A] > 0.$$

Superiority over alternative B requires some comparative criterion Q such that

$$\mathbb{E}[Q(A)] > \mathbb{E}[Q(B)]$$

under controlled conditions. Successful execution of A supplies no value for

$$\mathbb{E}[Q(B)]$$

and no reliable estimate of the expectation for A. Therefore feasibility does not imply comparative superiority. □ □

This non-guarantee result defines the proper status of the project. It is an experimental apparatus whose first outputs reveal useful distinctions. Its strongest contribution is not that it has solved automated writing, but that it has made several previously collapsed questions separable.

The limitations are therefore productive. Each identifies a missing transformation, measurement, or control condition. Factual grounding suggests a research stage. Reviewer uncertainty suggests multiple critics or confidence estimates. Repair failure suggests structured repair specifications or stronger revisers. Scaling suggests hierarchical memory. Protocol bias suggests genre-specific prompt families. Efficiency claims suggest resource instrumentation. Authorship concerns suggest explicit human approval boundaries.

A system becomes scientifically mature not when every limitation disappears, but when its limitations can be represented as hypotheses about where the next intervention should occur. The pipeline is valuable because it has begun to create precisely that representational structure.

14 Writing as a Trajectory of Representational Construction

The preceding analysis suggests that the most important change introduced by the pipeline is not the substitution of several language-model calls for one language-model call. It is a change in the object being modeled. In a conventional generative interface, the apparent object of computation is the essay. A topic enters the system and a document emerges. Planning, selection, reconsideration, criticism, and repair may occur implicitly, but whatever structure they possess is compressed into the terminal response. The pipeline instead treats the essay as one visible state within a longer trajectory of representational construction. What is being produced is not merely a text but a sequence of representations whose relations determine what the eventual text can become.

This interpretation provides a more precise account of why the outline matters. An outline is not simply a shorter version of the essay. If it were, drafting would amount primarily to expansion. The outline instead establishes a preliminary organization of distinctions and dependencies. It decides, provisionally, which conceptual problem is central, which propositions require explanation before others can be introduced, which objection deserves explicit treatment, and which consequences follow if the thesis survives. Its value lies not in containing the final prose but in constraining the space from which that prose will be generated.

The draft then performs a representational transformation. Relations that were compressed

into headings and short statements must be realized through sentences, paragraphs, examples, qualifications, and transitions. This transformation introduces information, but it also introduces opportunities for failure. A dependency obvious in the outline may disappear when several hundred words intervene between premise and conclusion. A distinction represented by two neighboring headings may blur when both are translated into ordinary prose. A qualification expressed compactly in the plan may be omitted because the model follows a locally more probable continuation.

Drafting should therefore not be regarded as a neutral rendering operation. It is a constructive transition during which the argument can acquire properties that did not exist at the outline stage. Some of these properties are desirable. The model can discover explanatory connections that were not explicit in the plan. It can introduce useful examples or identify a more natural order of exposition. Other emergent properties are defects. The generated prose can overstate the thesis, collapse distinctions, introduce unsupported empirical claims, or create contradictions.

The review stage operates on this realized representation rather than on the abstract plan. Its purpose is not merely to compare the draft with an ideal essay. It attempts to identify where the realized trajectory has departed from its own requirements. Some of those requirements originate in the outline; others emerge from claims made during drafting. Once the draft introduces a definition, later paragraphs become accountable to that definition even if it did not exist in the original plan. The artifact therefore generates new constraints as it develops.

This is an important property of writing that a simple optimization model tends to conceal. The target is not fully specified before generation begins. Writing partly constructs the criteria under which later writing must be evaluated. A definition creates obligations. A distinction creates cases that must remain distinguishable. An example can expose an exception that forces qualification. An objection can reveal that the thesis must be narrowed. The representational state therefore changes both the artifact and the space of admissible successors.

The pipeline is still only a coarse approximation to this process because its stages are externally imposed. Nevertheless, it exposes enough of the trajectory to demonstrate that the final essay cannot be understood entirely as the solution to a fixed initial objective. The problem changes as the representation develops.

This provides another reason to distinguish optimization from admissibility. Suppose the initial topic is represented by objective Q_0. If the entire writing task consisted of maximizing Q_0, every later state could be evaluated against the same fixed criterion. In actual argumentative construction, however, state X_t can introduce constraint c_t that did not exist explicitly at $t = 0$.

The admissible region at the next stage becomes

$$\mathcal{A}_{t+1} = \mathcal{A}_t \cap C(c_t),$$

where $C(c_t)$ denotes states consistent with the newly established constraint.

A successful trajectory can therefore become progressively more constrained. This is not necessarily a loss of freedom. The constraints encode commitments that give the artifact identity. Before a thesis is chosen, many essays are possible. After the thesis is chosen, essays contradicting it become inadmissible unless the thesis itself is explicitly revised. After a definition is introduced, inconsistent uses of the term become inadmissible. After evidence is accepted, conclusions incompatible with that evidence require explanation. Development converts possibility into structured commitment.

The process resembles construction more than search over a static space. A builder does not select a finished structure from an already enumerated collection of possible buildings. Each construction step changes the conditions under which later steps can occur. Once a foundation has been laid, it constrains the geometry and loads available to subsequent structures. Those constraints can be revised, but revision carries cost because dependent structures have already been built upon them.

Writing has an analogous dependency structure. An early conceptual decision can become load-bearing. Changing it late in the process may require rewriting several sections. This is why upstream repair is more expensive than local repair and why provenance matters for determining which commitments support which later claims.

The pipeline's filesystem externalizes part of this dependency structure, but much remains implicit inside the text. The outline supplies a coarse map. The review attempts to infer failures from the draft. A future architecture could make dependencies more explicit by representing definitions, claims, evidence, objections, and inferential relations separately from the prose that realizes them. Yet such formalization should be introduced cautiously because the richness of natural-language argument cannot always be reduced to a simple graph without loss.

The present experiment therefore occupies an intermediate position between unstructured generation and fully formal knowledge representation. It retains prose as the primary semantic medium while introducing enough external structure to make major transformations visible. This may be a productive region for human–model collaboration because the artifacts remain readable to humans while also being manipulable by ordinary computational tools.

The human reader can inspect `outline.md` without learning a specialized representation language. The model can consume the same file directly. Git can compare versions. Shell scripts can move the artifact between stages. Python can substitute it into a prompt. The

representation is therefore simultaneously communicative, computational, and archival.

This interoperability is one reason plain text remains unusually powerful despite its lack of formal semantic guarantees. A more elaborate internal representation might encode dependencies more precisely but would require translation whenever a human wished to inspect it or a model wished to discuss it. Markdown provides weaker structure but extremely low interface cost.

The pipeline consequently exploits a property of language models that differentiates them from many earlier computational components: natural language itself can function as an intermediate machine representation. A review written for a human can also condition another model. An outline that communicates argumentative structure to a reader can guide a generator. A theoretical definition can be both scholarly content and executable context.

This dual use should not be exaggerated. Natural language remains ambiguous. The same review can be interpreted differently by different models. Yet its partial machine usability changes the economics of externalized cognition. Intermediate representations no longer require complete translation into formal code before another computational component can act upon them.

The resulting system can therefore preserve more of its cognitive trajectory in representations that remain legible outside the machinery. This is one of the most consequential properties of language-model pipelines. Earlier computational systems often separated machine state sharply from human explanation. A program manipulated internal structures, and documentation described those structures separately. Here, some intermediate artifacts serve both roles.

The review is an especially clear example. It is simultaneously an explanation of perceived defects, a record of the critic's intervention, and input to the reviser. The artifact is not merely metadata about computation. It participates causally in later computation.

The same is true of the outline. It is both a human-readable plan and a machine-readable constraint. This duality allows the pipeline to move portions of its organization outside the models without making them inaccessible to human inspection. Externalization does not require replacing language with an opaque orchestration format.

This provides a possible model for human participation that differs from both manual writing and passive acceptance of generated prose. The human can intervene at representational boundaries. Instead of rewriting an entire essay, the investigator can modify the outline, reject a criticism, strengthen a definition, protect an invariant, or redirect a repair. Because the downstream stages operate on persistent artifacts, a relatively small human intervention can alter a substantial subsequent trajectory.

Such interventions may provide better leverage than editing only the terminal text. Correcting a conceptual distinction in the outline before drafting can prevent the error from propagating through thousands of words. Rejecting an invalid review recommendation can prevent normalization during revision. Adding a missing premise to the draft can make later criticism more informative. The architecture therefore creates several locations at which human judgment can be inserted.

The human need not intervene at every stage. Automation remains useful precisely because routine transformations can proceed without constant supervision. The important property is that intervention remains possible at semantically meaningful boundaries. The system is neither fully autonomous nor dependent upon continuous micromanagement.

This suggests a more general conception of human–AI collaboration as trajectory steering. The model supplies generative transitions; the human need not specify every token but can alter constraints governing which trajectories remain admissible. The relationship resembles navigation more than transcription. The computational system explores representational continuations while the human can modify the geometry within which those continuations are judged.

Under this interpretation, a prompt is one steering mechanism, but persistent artifacts are another. Changing the artifact can be more powerful than making the instruction increasingly elaborate. If a model repeatedly misunderstands a theoretical distinction, adding several paragraphs of prompt admonition may be less effective than placing a precise definition into a persistent framework file consumed by every relevant stage. The representation itself becomes part of the control system.

This observation connects the essay experiment to the broader problem of model memory. A language model does not need to contain every project-specific commitment in its parameters if those commitments can be represented externally and supplied when relevant. The filesystem becomes a form of durable project memory. Unlike conversational memory, it can be explicitly inspected, edited, versioned, and branched.

The challenge is retrieval. As the project grows, not every artifact can be supplied to every inference. The system must determine which historical representations constrain the present operation. This is not merely a context-window problem. It is a relevance problem. The correct continuation depends upon selecting the right history.

A mature architecture would therefore require a theory of representational reachability: which prior commitments remain causally relevant to a current stage, which can be summarized, which have been superseded, and which must be preserved verbatim because compression would destroy important distinctions. The simple pipeline avoids much of this difficulty

because each essay is small enough that its immediate predecessor artifacts can be supplied directly.

Nevertheless, the underlying problem is already visible. The final revision consumes the draft and review, but it does not necessarily consume the original outline. If the reviewer fails to notice that the draft has drifted from an outline invariant, the reviser has no direct access to that earlier constraint. The chosen context topology therefore determines which historical information can influence repair.

This means that pipeline architecture is partly a theory of memory. Every edge in the computational graph specifies which past representations remain available to which future transformations. Omitting an edge is a form of forgetting. Adding every possible edge is not necessarily desirable because excessive context can introduce distraction, contradiction, and cost. Memory architecture therefore requires selective continuation.

The essay experiment can consequently be understood as a small instance of a much broader problem: how should computational systems maintain enough history to preserve identity without forcing every future operation to replay the entire past?

The answer suggested here is hierarchical persistence. The complete history remains externally recoverable, while each stage receives only the representations judged relevant to its local transformation. Git can preserve everything at archival scale. The filesystem can preserve stage artifacts at project scale. Prompts can select immediate working context at inference scale. These are distinct memory horizons.

The architecture therefore contains several nested temporal structures. Model inference operates over seconds or minutes. A pipeline run persists across several invocations. An essay project persists across revisions. The repository persists across experiments. The theoretical program persists across repositories and documents. Continuation at each scale depends upon representations inherited from the scale below while imposing constraints upon the scale above.

This nesting explains why the pipeline is more than an essay generator. It is a prototype of a persistent representational environment in which transient models contribute to longer-lived intellectual structures. The models themselves can be replaced. The prompts can change. Individual drafts can fail. Yet the repository can preserve enough of these events that later work does not begin from zero.

The deepest motivation for such a system is cumulative competence. A model invocation is ephemeral. A well-constructed artifact can outlive the model that generated it. A useful review can become a permanent diagnostic record. A corrected definition can constrain later essays. A failed experiment can prevent repetition of the same architectural mistake. The durable unit

of progress is therefore not the inference but the representation that survives it.

This reverses a common emphasis in discussions of artificial intelligence. Attention is naturally directed toward what the model can do at the moment of inference. The present architecture asks what remains after inference ends. If nothing durable survives except a terminal answer, then each invocation contributes only its immediate product. If plans, distinctions, diagnoses, failures, repairs, and provenance survive as reusable representations, transient computational competence can participate in a much longer trajectory.

The essay-writing experiment is therefore an investigation into how temporary generative capacity can be converted into persistent intellectual structure. Its success should ultimately be judged not by whether one generated essay is impressive, but by whether repeated operation produces an environment in which later work becomes better informed, more diagnosable, and more capable of preserving distinctions discovered by earlier work.

14.1 Formalization: Endogenous Constraints and Representational Trajectories

Let the writing process be represented by trajectory

$$\gamma = (X_0, X_1, \ldots, X_n).$$

A fixed-objective model assumes some quality functional

$$Q : \mathcal{X} \to \mathbb{R}$$

defined independently of the trajectory, with writing represented as

$$X^* = \arg\max_{X \in \mathcal{X}} Q(X).$$

The trajectory model permits constraints to emerge during construction.

Let

$$C_t = \{c_1, \ldots, c_{m_t}\}$$

be the active constraint set after state X_t.

The admissible successor set is

$$\mathcal{A}_{t+1} = \{X \in \mathcal{X} : X \models C_t\}.$$

Suppose state X_t introduces new commitment

$$c_{m_t+1}.$$

Then

$$C_{t+1} = C_t \cup \{c_{m_t+1}\},$$

and therefore

$$\mathcal{A}_{t+2} = \{X : X \models C_{t+1}\}.$$

If the new constraint is nonredundant,

$$\mathcal{A}_{t+2} \subsetneq \mathcal{A}_{t+1}.$$

Thus representational development can progressively restrict future admissibility.

This restriction should not be interpreted as monotonic accumulation of immutable commitments. A later repair may revise constraint c_j. Let

$$\rho_c : C_t \to C_t'$$

replace

$$c_j$$

with

$$c_j'.$$

Then every downstream state depending upon c_j becomes a candidate for reevaluation.

Define dependency relation

$$c_j \prec x_k$$

when component x_k of the artifact depends upon c_j.

The invalidation set induced by repair is

$$I(\rho_c) = \{x_k : c_j \prec x_k\}.$$

The cost of upstream conceptual revision therefore grows with the measure of its dependency closure:

$$C_{\text{repair}}(\rho_c) \propto |I(\rho_c)|$$

under a simple approximation.

This formalizes why early conceptual decisions become increasingly expensive to change as an essay develops.

Now let the outline be

$$O = (V_O, E_O),$$

where V_O represents conceptual units and E_O their intended dependency relations.

Drafting is transformation

$$\delta : O \to D.$$

A perfect realization would preserve every relevant relation:

$$(u, v) \in E_O \Rightarrow \delta(u) \prec_D \delta(v).$$

In practice, some relations are lost:

$$E_{\text{lost}} = E_O \setminus \delta^{-1}(E_D).$$

Others emerge during drafting:

$$E_{\text{new}} = E_D \setminus \delta(E_O).$$

Drafting is therefore not simple expansion because generally

$$E_{\text{lost}} \neq \varnothing$$

or

$$E_{\text{new}} \neq \varnothing.$$

Proposition 14.1 (Drafting is generative rather than merely expansive)**.** *If drafting can introduce constraints or dependency relations not represented in the outline, then the draft cannot in general be reconstructed as a purely length-expanding homomorphism of the outline.*

Proof. Assume drafting were purely expansive and structure preserving. Then every semantically relevant relation in D would correspond to a relation already represented in O. Suppose instead that drafting introduces new relation

$$e^* \in E_D$$

such that

$$e^* \notin \delta(E_O).$$

Then D contains structure not inherited from O. Therefore δ is not merely a length-expanding homomorphism; it is a generative transformation. □ □

This explains why reviewing the outline cannot substitute for reviewing the draft.

The role of human intervention can now be modeled as modification of constraints rather than token-level generation.

Let automated transition be

$$X_{t+1} = F_M(X_t, C_t).$$

A human intervention modifies

$$C_t \to C_t'$$

or

$$X_t \to X_t'$$

before the next automated transition.

The resulting successor is

$$X_{t+1}' = F_M(X_t', C_t').$$

A small intervention can have large downstream effect when

$$d(X_t, X_t') \ll d(X_{t+k}, X_{t+k}')$$

for later k.

Define intervention leverage

$$\Lambda_H = \frac{d(X_{t+k}, X_{t+k}')}{d(X_t, X_t') + \epsilon},$$

where $\epsilon > 0$ prevents division by zero.

High-leverage intervention occurs when

$$\Lambda_H \gg 1.$$

Correcting a load-bearing outline distinction can therefore be more influential than editing many sentences after the essay is complete.

Memory can similarly be represented as graph connectivity. Let artifact X_i be available to later transformation F_j when

$$(X_i, F_j) \in E_M,$$

where E_M is the memory-access relation.

If

$$(X_i, F_j) \notin E_M,$$

then F_j cannot condition directly upon X_i, even if X_i remains stored elsewhere.

Thus

$$\text{stored} \Rightarrow \text{active}.$$

This distinction separates archival memory from working memory.

Let

$$\mathcal{H}$$

be complete persisted history and

$$W_j \subseteq \mathcal{H}$$

the working context selected for stage j.

Then

$$F_j = F_j(W_j).$$

The context-selection problem is

$$W_j^* = \arg\max_{W \subseteq \mathcal{H}} U_j(W)$$

subject to

$$L(W) \leq K_j,$$

where K_j is the context capacity and U_j is downstream utility.

This is a constrained memory-allocation problem.

A hierarchical memory architecture introduces representations

$$\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_m$$

at different temporal and semantic resolutions.

For example,

$$\mathcal{H}_0 = \text{current draft context},$$

$$\mathcal{H}_1 = \text{essay-level artifacts},$$

$$\mathcal{H}_2 = \text{repository history},$$

$$\mathcal{H}_3 = \text{research-program history}.$$

A retrieval operator

$$\Gamma_j : \bigcup_k \mathcal{H}_k \to W_j$$

selects the subset relevant to the current transformation.

The quality of continuation therefore depends upon both generative capability and historical selection:

$$Q_{t+1} = Q(M, p, X_t, \Gamma(\mathcal{H})).$$

Even a highly capable model can fail when

$$\Gamma(\mathcal{H})$$

omits a necessary invariant.

Finally, define transient computational competence at inference i as

$$C_i^{\text{transient}}.$$

Let persistent artifact Z_i encode some reusable consequence of that competence. Its durable contribution to future trajectories is

$$C_i^{\text{durable}} = \sum_{j>i} \omega_{ij} U_j(Z_i),$$

where ω_{ij} weights future uses and U_j measures the usefulness of Z_i to later transformation j.

An inference that produces a spectacular but nonreusable terminal response may have high immediate value

$$C_i^{\text{transient}}$$

but low

$$C_i^{\text{durable}}.$$

An inference producing a compact definition, diagnosis, or structural representation that constrains many later operations may have the opposite profile.

This motivates a different measure of computational contribution:

$$C_i^{\text{total}} = C_i^{\text{immediate}} + \lambda C_i^{\text{durable}}.$$

The pipeline is designed to increase the second term by ensuring that useful model outputs do not vanish when inference ends.

The central proposition of this section can therefore be stated as follows.

Theorem 14.2 (Persistent representation converts transient inference into potential cumulative competence)**.** *Let inference I_t produce information Z_t relevant to at least one future transformation F_{t+k}. If Z_t is unavailable at $t+k$, its direct conditioning effect on F_{t+k} is zero. If Z_t is persistently stored, retrievable, and included in the working context of F_{t+k}, then its information can alter the distribution of reachable successors. Therefore persistence is a necessary, though not sufficient, condition for the direct accumulation of inference-specific information across otherwise independent model invocations.*

Proof. Without persistence or equivalent reconstruction,

$$Z_t \notin W_{t+k}.$$

Hence

$$P(X_{t+k+1} \mid W_{t+k})$$

cannot condition directly upon Z_t.

With persistence and retrieval,

$$Z_t \in W_{t+k},$$

and the successor distribution becomes

$$P(X_{t+k+1} \mid W_{t+k}, Z_t).$$

Whenever Z_t contains task-relevant information,

$$P(X_{t+k+1} \mid W_{t+k}, Z_t) \neq P(X_{t+k+1} \mid W_{t+k})$$

in general.

Thus persistent retrieval permits information produced by an earlier transient inference to influence later independent inference. Persistence alone does not guarantee usefulness because

Z_t may be irrelevant, incorrect, or poorly represented, but without persistence or reconstruction its direct cumulative contribution is unavailable. □ □

The essay pipeline can therefore be understood as an elementary mechanism for converting transient inference into persistent representational capital. Outlines, drafts, reviews, scripts, prompts, and revisions become resources available to later computation rather than disposable by-products of individual model calls.

What accumulates is not intelligence in the sense of permanently changing the model's parameters. What accumulates is an environment increasingly structured by the consequences of earlier intelligence. That distinction may prove more important than it initially appears. A system need not continually retrain itself in order for its future behavior to benefit from its past, provided that the past leaves behind representations that remain accessible, discriminating, and admissible.

The resulting research question is therefore no longer merely how to make a model write an essay. It is how to construct an environment in which acts of generation become durable contributions to a continuing process of inquiry.

15 Experimental Extensions and a Program of Comparative Evaluation

The initial pipeline was constructed as an exploratory apparatus rather than as a completed benchmark. Its first purpose was to determine whether heterogeneous local language models could be assigned distinct intellectual roles, whether their intermediate products could be preserved as ordinary files, and whether those products would expose differences between generation, diagnosis, and repair that disappear in a one-shot interaction. Having established that the architecture is operational, the next stage is not simply to add more components. It is to convert the architecture into a controlled experimental system in which competing explanations can be separated. The central methodological transition is therefore from demonstration to comparison.

The most immediate experiment concerns model allocation. The current pipeline assigns Granite 4.1 3B to the relatively generative stages and Granite 4.1 8B to review. This allocation embodies a hypothesis: that additional model capacity has greater marginal value during criticism than during initial generation. The existing essays provide suggestive evidence because the larger reviewer often identifies conceptual weaknesses that are not adequately repaired by the smaller reviser. Yet the observation remains confounded by stage, model size, prompt, and stochastic variation. The architecture should therefore be run under alternative assignments

while holding the topic, prompt files, and as much of the surrounding environment as possible fixed.

The simplest comparison would place the 3B model in every role, producing a homogeneous low-cost baseline. A second condition would place the 8B model in every role, producing a homogeneous higher-capacity baseline. The existing asymmetric configuration would remain as a third condition, with 3B generating and revising while 8B reviews. A fourth condition should reverse the asymmetry: 8B generates the outline and draft while 3B performs criticism and perhaps 8B performs final revision. Further conditions could isolate revision capacity by holding a frozen draft and frozen review constant while assigning different models only to the final repair operation.

These comparisons would answer different questions. If the 8B-everywhere condition substantially outperforms the heterogeneous pipeline at modest additional computational cost, the architectural argument for asymmetric allocation would weaken. If the heterogeneous pipeline approaches the quality of the 8B-everywhere condition while consuming substantially fewer resources, selective allocation would receive empirical support. If a strong reviewer followed by a weak reviser repeatedly underperforms a weak reviewer followed by a strong reviser, the original assumption concerning the location of marginal model value would require revision. The purpose of the experiment is precisely to make such reversals possible.

A particularly important test concerns the apparent separation between diagnosis and repair. The existing architecture makes it possible to freeze both the draft and the review. Once these artifacts are fixed, several revisers can receive exactly the same informational state. The resulting essays can then be compared without confounding revision performance with differences in diagnosis. If the 8B reviser consistently resolves criticisms that the 3B reviser leaves untouched, the experiment would provide evidence that repair reachability depends upon model capacity even when diagnostic information is held constant.

The converse experiment is equally important. A single frozen draft can be supplied independently to 3B and 8B reviewers, after which the same reviser receives each resulting review. This separates review quality from repair capacity. If the final artifact improves more after the 8B review even though the reviser is unchanged, then some advantage of the larger model has successfully crossed the textual interface. The review has functioned as an externalized cognitive resource. If no improvement occurs, the failure may indicate either that the stronger review contains little additional useful information or that the representation in which it expresses that information is unusable by the downstream model.

The distinction suggests an experiment in interface design. The current review is free-form prose. Alternative review protocols could require a more explicit structure while preserving the same underlying diagnostic task. One representation might require each criticism to

identify a location, defect, evidence, severity, and proposed repair. Another might distinguish violations of definitions from unsupported claims, structural contradictions, missing evidence, and optional stylistic improvements. Another could require the reviewer to state which earlier artifact or invariant supports each criticism.

The objective would not be to discover the most elaborate review format. Additional structure carries costs. It consumes tokens, may constrain the reviewer into inappropriate categories, and can create false precision. The question is whether some representation increases the fraction of diagnostically useful information that survives the transition from critic to reviser. A shorter structured review could outperform a longer natural-language review if it better specifies actionable transformations.

This can be tested directly. Given frozen draft D, reviewer M_r can produce several diagnostic representations

$$R^{(1)}, R^{(2)}, \ldots, R^{(k)}$$

under different review protocols. A fixed reviser M_v then produces

$$F^{(i)} = M_v(D, R^{(i)}).$$

Independent evaluation of the resulting $F^{(i)}$ estimates the downstream utility of each interface.

A further experiment should test the preservation rule itself. The current revision philosophy favors conservative repair: retain material that has survived criticism and modify only what the review provides sufficient reason to change. This protects successful structure, but the previous section showed that it can also preserve reviewer false negatives. The strength of the preservation constraint should therefore become an experimental variable.

One condition could instruct the reviser to make only explicitly requested repairs. Another could allow local collateral changes when necessary to preserve coherence. A third could permit global rewriting while requiring preservation of specified conceptual invariants. Comparing these conditions would help determine whether repair quality follows a monotonic relation with freedom of intervention or whether there is an intermediate region in which sufficient flexibility is available without allowing wholesale normalization.

This experiment is especially important for essays developing nonstandard theoretical concepts. An unconstrained reviser may produce smoother prose by translating unusual distinctions into familiar vocabulary. Conventional measures of fluency may rate the result more highly even though theoretical fidelity has declined. Evaluation must therefore separate surface quality from invariant preservation.

The project should consequently maintain explicit invariant files for at least some experiments.

An invariant file would contain the definitions, distinctions, or propositions whose preservation is part of the experimental task. It would not declare those propositions true. Rather, it would define the identity conditions of the artifact under revision. A reviewer would remain free to criticize an invariant, but a routine reviser would not be permitted to silently replace it. Criticism of a protected invariant would instead trigger escalation to a higher-level decision.

This distinction makes it possible to test a central claim of the broader repair framework: successful correction should preserve relevant difference rather than merely reduce visible irregularity. If an essay becomes more conventional, fluent, and apparently coherent by eliminating the distinction it was constructed to investigate, the transformation should not count as an unqualified improvement. It may have reduced one loss function while destroying the identity of the research object.

Another experimental extension concerns upstream repair. The current pipeline is predominantly forward-moving. A review is followed by a revision of the draft. Yet some criticisms clearly implicate the outline or even the topic formulation. A future script could classify each review finding according to the earliest stage that must change for the defect to be removed. Local defects would continue directly to revision. Structural defects would trigger a revised outline and regeneration of the affected essay. Conceptual defects might return to the topic specification or invariant file.

This creates an opportunity to compare local and upstream repair experimentally. Given a defect diagnosed as structural, one condition can attempt to repair the final draft directly. Another can revise the outline and regenerate the draft from the repaired plan. The resulting artifacts can then be compared for defect removal, collateral change, computational cost, and preservation of unaffected material.

The expectation is not that upstream repair will always be superior. Its greater structural reach also creates greater collateral risk. Regenerating an essay from a modified outline can alter paragraphs unrelated to the original defect simply because language generation is stochastic. Local repair may therefore be preferable whenever the defect can be removed without changing its dependencies. The experimental question is where the transition occurs.

Repeated sampling is necessary throughout these comparisons. One essay from each condition cannot establish stable differences between models or protocols. The same topic should be run several times under nominally identical conditions so that within-condition variation can be estimated. This will also reveal whether some architectures are more stable than others. Two pipelines can have similar average quality while differing substantially in variance.

Variance itself may be an important property of useful systems. A highly variable generator can be valuable during exploration because it discovers diverse representations. The same

variability can be undesirable during repair, where preservation is important. This suggests another reason for heterogeneous stage design: the desired stochastic behavior may differ by operation. Planning may benefit from diversity, while final revision may benefit from stability.

The experimental apparatus should therefore begin recording inference parameters explicitly. Model identity should include more than a convenient tag such as `granite4.1:8b`. Where possible, the model digest, Ollama version, context length, temperature, sampling parameters, prompt size, output size, and execution time should accompany each artifact. These values can be written to a metadata file in the output directory without altering the basic human-readable structure of the experiment.

The Bash pipeline can also record wall-clock duration for each stage with little additional complexity. Token counts may require model or runtime support, but even coarse timing measurements would improve the current resource analysis. Memory and GPU utilization could later be sampled if computational efficiency becomes a central claim. Instrumentation should grow in proportion to the questions being asked rather than turning the experimental script prematurely into a monitoring framework.

The appropriate cost measure is multidimensional. Wall-clock time matters to the user. Energy matters to physical efficiency. Memory consumption determines which models can coexist with other workloads. Token volume indicates how much textual computation occurs. Model size affects storage and loading. A single scalar cost can eventually be constructed for particular comparisons, but the raw dimensions should initially remain separate.

This is especially important because the larger model may sometimes be cheaper at the system level. If an 8B reviewer prevents three unsuccessful 3B revision cycles, the locally expensive operation may reduce total trajectory cost. The relevant object of optimization is therefore not the cost of an invocation but the cost of reaching an admissible terminal artifact.

The same reasoning applies to human labor. A pipeline that consumes more GPU time but substantially reduces manual diagnosis may be preferable for some workflows. Conversely, a computationally efficient pipeline that requires the human to inspect every intermediate state may simply shift the cost from machine resources to attention. Human intervention should therefore eventually be recorded as another resource dimension.

A simple experimental approximation could record whether human intervention occurred between stages and the approximate duration of that intervention. More detailed measurement would risk disrupting the natural workflow. The purpose is not to quantify every cognitive act but to avoid treating human supervision as free when comparing automation strategies.

The current essay topics provide a useful initial benchmark family because they arise from a common theoretical vocabulary while differing in their immediate claims. Essays on admissi-

bility, continuation, repair, technological diffusion, and related concepts permit comparison across repeated conceptual structures without requiring every run to address an identical subject. At the same time, a benchmark consisting only of theories developed by the investigator would risk overfitting the prompts to one intellectual style.

A second benchmark family should therefore contain topics outside the framework. These could include conventional philosophical questions, explanations of established scientific concepts, software-engineering arguments, and historical analysis. The objective would be to determine whether the pipeline's advantages are specific to the kind of conceptual writing for which it was designed.

A third benchmark family should deliberately stress the architecture. Topics could be selected that require maintaining several closely related definitions, responding to strong objections, distinguishing descriptive from normative claims, or preserving a counterintuitive thesis across revision. Such cases would test the diagnostic and preservation mechanisms more directly than ordinary expository essays.

Factual research should initially remain a separate experiment. Once retrieval is introduced, the pipeline gains another epistemic boundary: the distinction between generated claims and externally supported claims. A research stage could gather candidate sources before drafting, while a verification stage could inspect factual statements after drafting. The resulting evidence artifacts should remain separate from prose so that citations can be traced to their sources rather than generated as decorations.

This would substantially alter the architecture. The current system asks whether one representation follows coherently from another. A grounded pipeline must additionally ask whether claims correspond to evidence not generated by the pipeline itself. The admissible reference class then includes external documents, measurements, or databases. This is a stronger form of correction because the system gains information that cannot be reconstructed merely by reconsidering its own output.

Formal verification provides another extension for essays containing mathematics. A language-model reviewer can inspect a proof, but its judgment remains probabilistic and linguistic. Some claims can instead be translated into forms suitable for symbolic algebra, numerical testing, proof assistants, or executable checks. The appropriate architecture would not ask the language model to replace these tools. It would route claims toward the strongest available verification mechanism.

This principle generalizes the heterogeneous model hypothesis. Heterogeneity should not be limited to different sizes of language model. Different computational tasks have different natural instruments. Shell scripts are better than language models at deterministic file

movement. Git is better at preserving exact historical states. Python is better at structured text processing. Symbolic systems are better at exact algebraic transformations. Retrieval systems are better at locating external evidence. Language models are particularly useful where transformations require flexible semantic interpretation. The mature architecture should therefore assign operations across heterogeneous computational species rather than attempting to make the language model simulate all of them.

The current repository already embodies this principle in elementary form. Bash does not ask Granite to decide where files should be written. Python does not ask Granite to perform template substitution. Git does not ask Granite whether two byte sequences differ. Each component is used where its semantics are stronger and more predictable than generative inference. The future research program should preserve this restraint.

The eventual experimental comparison should include a monolithic baseline. A sufficiently detailed single prompt can ask the 8B model to plan, draft, critique, and revise internally before returning only the final essay. Such a condition may produce excellent results. The point of the staged architecture is not to assume otherwise. The question is what is gained or lost when the intermediate operations are externalized.

The monolithic system may be faster, simpler, and perhaps even better on immediate terminal quality. The staged system may offer stronger provenance, reusable diagnoses, easier substitution, more precise human intervention, and better failure localization. These are different dimensions of system quality. A meaningful comparison should not collapse them into a single aesthetic score.

The experimental program should therefore evaluate terminal quality, resource cost, diagnostic resolution, repair success, invariant preservation, reproducibility, and human intervention separately before considering aggregate measures. The architecture may dominate on some dimensions and lose on others. Such a result would be more informative than declaring one system universally superior.

The broader methodological objective is to discover where explicit decomposition pays for itself. Every boundary has a cost. Files must be written and read. Prompts become longer. Errors can arise at interfaces. More stages create more opportunities for failure. Decomposition is justified only when the distinction exposed by a boundary produces enough diagnostic, computational, or epistemic value to offset that cost.

The current outline–draft–review–revision structure appears promising because its boundaries correspond to recognizable differences in intellectual operation. Future experiments may show that some should be merged or subdivided. Perhaps outline and draft generation benefit from remaining separate while review and revision work better within one stronger model

invocation. Perhaps mathematical checking deserves several sub-stages while ordinary prose does not. The architecture should remain experimentally revisable.

This produces an important methodological symmetry. The pipeline is designed to repair essays, but the pipeline itself must be treated as a repairable artifact. Its scripts, prompts, stage boundaries, model assignments, and evaluation criteria are provisional representations. When experiments expose a failure, the response should not be to defend the architecture because it expresses the theory. The architecture should be revised under the same principles it imposes upon its outputs: preserve what survives criticism, modify what can be shown to fail, retain provenance, and escalate when local repair is insufficient.

The research program is therefore recursive in a productive sense. Generated essays test the pipeline. Failures in the essays reveal weaknesses in the pipeline. Changes to the pipeline produce new essay trajectories. Version control preserves these architectural revisions so that improvement claims can themselves be inspected. The experimental apparatus participates in the same logic of continuation that it is intended to study.

15.1 Formalization: Experimental Factorization and Comparative Tests

Let the principal experimental variables be model allocation

$$A,$$

review representation

$$R,$$

repair policy

$$P,$$

and topic

$$T.$$

Let model allocation be drawn from

$$A \in \{A_{3,3,3,3}, A_{8,8,8,8}, A_{3,3,8,3}, A_{8,8,3,8}, \ldots\},$$

where the entries correspond respectively to outline, draft, review, and revision models.

The current experimental condition is approximately

$$A_{\text{current}} = A_{3,3,8,3}.$$

For replicate k, the final essay is

$$F_{a,r,p,t,k} \sim \mathcal{W}(a, r, p, t).$$

Let the evaluation vector be

$$\mathbf{Q}(F) = \begin{pmatrix} Q_{\text{argument}} \\ Q_{\text{fidelity}} \\ Q_{\text{repair}} \\ Q_{\text{grounding}} \\ Q_{\text{style}} \end{pmatrix}.$$

Let the resource vector be

$$\mathbf{C}(F) = \begin{pmatrix} C_{\text{time}} \\ C_{\text{tokens}} \\ C_{\text{memory}} \\ C_{\text{energy}} \\ C_{\text{human}} \end{pmatrix}.$$

No architecture is assumed to dominate another unless it satisfies the relevant comparative criterion across the dimensions being considered.

For scalar quality measure Q, the estimated mean under allocation a is

$$\widehat{\mu}_a = \frac{1}{N} \sum_{k=1}^{N} Q(F_{a,k}).$$

The comparison between heterogeneous allocation a_h and homogeneous large-model allocation a_ℓ is

$$\Delta Q = \widehat{\mu}_{a_h} - \widehat{\mu}_{a_\ell},$$

with resource difference

$$\Delta C = \widehat{C}_{a_h} - \widehat{C}_{a_\ell}.$$

The heterogeneous architecture is Pareto-superior only if it is no worse on every selected criterion and strictly better on at least one. More realistically, it may occupy a different point on the quality–cost frontier.

Define efficiency

$$\eta(a) = \frac{\mathbb{E}[Q(F_a)]}{\mathbb{E}[C(F_a)]}$$

for a chosen scalarized resource cost.

The hypothesis motivating asymmetric allocation can then be expressed as

$$H_1 : \eta(A_{3,3,8,3}) > \eta(A_{8,8,8,8}),$$

subject to an acceptable bound on absolute quality loss.

A stronger formulation is

$$H_1' : Q(A_{3,3,8,3}) \geq Q(A_{8,8,8,8}) - \epsilon$$

while

$$C(A_{3,3,8,3}) < C(A_{8,8,8,8})$$

for practically meaningful tolerance ϵ.

The diagnosis–repair experiment freezes draft D and review R. Let

$$F_s = M_s(D, R)$$

and

$$F_\ell = M_\ell(D, R).$$

Let diagnosed defect set be

$$E_R = \{e_1, \ldots, e_m\}.$$

Define repair success for model M as

$$S_{\text{repair}}(M) = \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}\left[e_j \text{ is removed without violating protected invariants}\right].$$

The repair-capacity hypothesis is

$$H_2 : \mathbb{E}[S_{\text{repair}}(M_\ell)] > \mathbb{E}[S_{\text{repair}}(M_s)].$$

Because D and R are held fixed, evidence for H_2 cannot be attributed merely to superior diagnosis.

Conversely, let frozen draft D be reviewed by two models:

$$R_s = M_s(D), \qquad R_\ell = M_\ell(D).$$

A fixed reviser M_v produces

$$F_s = M_v(D, R_s), \qquad F_\ell = M_v(D, R_\ell).$$

Then

$$H_3 : \mathbb{E}[Q(F_\ell)] > \mathbb{E}[Q(F_s)]$$

tests whether the larger reviewer's advantage survives externalization through the review artifact.

The interface experiment introduces review encoding

$$\sigma_i.$$

For common internal diagnostic target H_R,

$$R_i = \sigma_i(H_R).$$

Downstream utility is

$$U(\sigma_i) = \mathbb{E}\left[Q(M_v(D, R_i)) - Q(D)\right].$$

The optimal representation within tested class Σ is

$$\sigma^* = \arg\max_{\sigma_i \in \Sigma} U(\sigma_i).$$

This formalizes the claim that review quality and review interface quality are distinct experimental variables.

Now consider protected invariants

$$I = \{i_1, \ldots, i_n\}.$$

For transformation

$$\rho : D \to F,$$

define invariant retention

$$P_I(\rho) = \frac{1}{n} \sum_{j=1}^{n} \mathbf{1}[F \models i_j].$$

Let defect reduction be

$$\Delta E(\rho) = |E(D)| - |E(F)|.$$

A naive revision metric might maximize only

$$\Delta E.$$

A repair-sensitive metric instead evaluates

$$Q_{\text{repair}}(\rho) = \alpha \Delta E(\rho) + \beta P_I(\rho) - \gamma C_{\text{collateral}}(\rho),$$

where

$$\alpha, \beta, \gamma > 0.$$

This captures the principle that removing defects by destroying the identity-bearing structure of the artifact does not constitute successful repair.

Upstream escalation can be treated as an intervention-selection problem. Let available repair levels be

$$L = \{\ell_{\text{sentence}}, \ell_{\text{paragraph}}, \ell_{\text{draft}}, \ell_{\text{outline}}, \ell_{\text{topic}}\}.$$

For diagnosed defect e, each intervention level has expected repair value

$$V(\ell \mid e) = P_{\text{success}}(\ell \mid e) B(e) - C(\ell) - R_{\text{collateral}}(\ell).$$

The optimal intervention is

$$\ell^* = \arg\max_{\ell \in L} V(\ell \mid e).$$

The minimal-repair hypothesis predicts that, other things equal, the preferred intervention will be the shallowest level whose expected value is positive and whose probability of successful repair exceeds the required threshold.

Repeated sampling permits estimation of stability. For architecture A, let

$$Q_1, \ldots, Q_N$$

be replicate quality measurements.

Define

$$\widehat{\sigma}_A^2 = \frac{1}{N-1} \sum_{k=1}^{N} (Q_k - \widehat{\mu}_A)^2.$$

Two systems may satisfy

$$\widehat{\mu}_{A_1} \approx \widehat{\mu}_{A_2}$$

while

$$\widehat{\sigma}^2_{A_1} \ll \widehat{\sigma}^2_{A_2}.$$

If the task is final repair, the lower-variance architecture may be preferable even without higher mean quality. If the task is exploratory ideation, the higher-variance system may have greater discovery value.

This demonstrates why the desired stochastic regime is stage-dependent.

The total cost of reaching an admissible artifact should finally be distinguished from per-inference cost. Let a trajectory contain n inference operations with costs

$$c_1, \ldots, c_n.$$

Then

$$C_{\text{trajectory}} = \sum_{i=1}^{n} c_i + C_{\text{human}} + C_{\text{failed}}.$$

A locally expensive intervention c_j is globally efficient whenever its inclusion reduces expected later costs by more than its marginal expense:

$$c_j < \mathbb{E}[C_{\text{future}} \mid \neg c_j] - \mathbb{E}[C_{\text{future}} \mid c_j].$$

Thus the statement

"the 8B model costs more"

does not imply

"using the 8B model increases total pipeline cost."

The relevant quantity is expected cost to admissibility:

$$C^* = \mathbb{E}\left[\sum_{t=0}^{\tau_{\mathcal{A}}} c_t\right],$$

where

$$\tau_{\mathcal{A}} = \inf\{t : X_t \in \mathcal{A}\}$$

is the first time the trajectory reaches the admissible region.

This yields a more general experimental objective:

Compare architectures by the trajectories they make reachable, the evidence they preserve, and the cost requi

The proposed experiments therefore do more than benchmark two Granite models. They test whether the architecture's central distinctions correspond to measurable differences. They ask whether diagnosis can be separated from repair, whether stronger competence can be transmitted through persistent representations, whether conservative repair preserves useful difference, whether upstream escalation can be justified by dependency structure, and whether heterogeneous allocation can approach the quality of uniformly stronger inference at lower total cost.

Each question can fail. That possibility is essential. If the experiments show that a single 8B invocation consistently dominates the staged system on quality, cost, and reliability, the pipeline should be simplified. If free-form reviews outperform structured diagnoses, the additional interface should be rejected. If preservation constraints systematically inhibit necessary revision, they should be weakened. If the 3B model cannot exploit the 8B model's externalized criticism, the assumed diffusion of competence across the textual boundary has reached a measurable limit.

The research program is valuable precisely because these outcomes would teach us something. The pipeline is not an implementation of conclusions already known to be true. It is an apparatus for turning those conclusions into propositions that can survive, fail, or be repaired under experiment.

16 Discussion: From Model Capability to Cumulative Competence

The experimental pipeline began from an ordinary practical problem. Two local language models of different capacities were available on the same machine, and it seemed unnecessary to assign the larger model to every stage of essay production. The smaller Granite 4.1 3B model could generate substantial prose at relatively modest computational cost, while the Granite 4.1 8B model appeared more useful when asked to inspect arguments critically. Bash, Markdown files, a small Python prompt renderer, Ollama, and Git were sufficient to turn this asymmetry into an executable workflow. What emerged from that implementation, however, was a more general question than how to automate essay writing. The experiment asks how transient differences in computational competence can be converted into persistent improvements in a larger system.

This distinction between capability and cumulative competence is central. A language model

possesses some set of capabilities while it is being executed. It can generate a distinction, identify an inconsistency, propose a definition, or discover an objection. Unless the consequence of that operation is preserved, however, the useful result exists only within the immediate interaction. Another invocation need not inherit it. The computational capacity has been exercised, but the surrounding system has not necessarily become more capable.

Persistent intermediate artifacts change this relation. When the 8B model identifies a defect and writes that diagnosis into `review.md`, the review survives the termination of the inference that produced it. A later process can inspect it. A weaker model can attempt to act upon it. A human can reject or refine it. A future experiment can compare it with another review. Git can preserve it after the current version of the pipeline has changed. A transient act of computation has therefore produced a durable constraint upon possible future computation.

This is not equivalent to transferring the 8B model's competence into the 3B model. The smaller model's parameters remain unchanged. Its intrinsic capacities have not increased. What changes is the environment from which it operates. The reachable set of outputs can nevertheless change because information unavailable before the review is now present in its context. The distinction between improving an agent and improving the environment of an agent is therefore fundamental.

Human intellectual development has always depended heavily upon the latter mechanism. An individual does not independently rediscover arithmetic, algebra, calculus, formal logic, experimental methodology, statistical inference, programming, and every engineering convention required by modern technical work. Competence accumulates culturally because the results of earlier cognitive labor are externalized into durable forms. Notation, textbooks, diagrams, source code, standards, proofs, procedures, instruments, and institutions allow later individuals to begin from representations that earlier individuals had to construct.

The resulting increase in capability cannot be understood entirely as an increase in the intrinsic intelligence of individual humans. A contemporary student can solve problems unavailable to exceptionally intelligent people in earlier centuries because the student's environment contains accumulated representations. The civilization has become more competent even if the biological capacities of its individual members have changed little.

The essay pipeline implements a miniature computational analogue of this process. The larger model performs a cognitive operation that the smaller model may not perform as reliably on its own. The result is externalized into a representation. That representation becomes part of the environment available to subsequent computation. The critical experimental question is how much of the original competence survives this externalization.

This question connects directly to the broader problem of technological diffusion. A tech-

nological capability can remain concentrated inside the system that produces it, or some representation of the method can become available to other systems. These two cases can produce similar immediate services while having very different long-term consequences. If a powerful system performs a task for a user while revealing nothing reusable about how the task was accomplished, the user receives an output without necessarily acquiring additional capability. If the system instead produces inspectable procedures, explanations, standards, code, or intermediate representations, some fraction of the capability can propagate beyond the original service.

The distinction is not absolute. A usable service is itself a form of distributed capability. A person does not need to understand semiconductor fabrication to benefit from a computer, nor must every programmer understand the physics of magnetic storage before using a filesystem. Civilization depends extensively upon abstraction. The problem arises when abstraction becomes enclosure: when the interface permits consumption of results while systematically preventing the surrounding environment from accumulating transferable understanding.

The local essay pipeline provides an unusually simple case because its intermediate representations are deliberately exposed. The models do not merely return a final essay. They leave behind a plan, draft, criticism, revision, prompts, scripts, and history. These artifacts do not fully reveal the models' internal mechanisms, but they expose considerably more of the task decomposition than a terminal response alone.

This is a form of competence diffusion at the level of consequences rather than parameters. The 8B model's weights are not copied into the 3B model. Instead, a useful result of the larger model's computation becomes available in a lower-cost representation. The important quantity is therefore not whether competence is transferred completely but whether enough task-relevant structure survives to alter subsequent reachable states.

The review–revision boundary provides an empirical test of this principle. If the 8B model identifies a defect that the 3B model could not independently identify, and the 3B model successfully repairs the defect after reading the review, then some useful consequence of the larger model's competence has crossed the boundary. The smaller model has performed a transformation that became reachable because the environment was enriched by the larger model's diagnosis.

If the smaller model understands the criticism superficially but cannot perform the repair, the experiment reveals a boundary of diffusion. Information has crossed the interface, but transformational competence has not. The review can tell the smaller model where the destination lies without providing a reachable path to it. This distinction prevents an overly strong interpretation of externalization. Knowledge of a required transformation and capacity to execute that transformation are separable.

A similar distinction occurs in human education. Reading a proof can reveal that a theorem is true without supplying the ability to discover comparable proofs independently. A textbook can explain a programming technique that a novice cannot yet apply reliably. An engineering specification can state tolerances without conferring the tacit skill required to manufacture the component. External representations diffuse competence imperfectly because the recipient must possess enough complementary structure to use them.

The pipeline therefore suggests a conditional principle of capability diffusion. Externalization is most effective when the recipient already occupies a state sufficiently close to the required transformation that the additional representation makes the target reachable. A review cannot transform an arbitrarily weak generator into an arbitrarily strong reviser. It can alter the local geometry of the problem.

This provides a more precise alternative to the intuition that information simply "adds knowledge." Information is useful relative to a reachable set. The same diagnostic artifact can be decisive for one model, redundant for another, and unusable for a third. The value of the representation is relational.

This relation also explains why open publication of knowledge can have effects disproportionate to the immediate information contained in a document. A method becomes socially powerful when it enters environments containing agents capable of adapting, combining, teaching, criticizing, and extending it. The document does not contain all subsequent competence. It modifies the reachable space of a population.

The essay pipeline operates on a much smaller scale, but the structural relation is similar. A persisted review can be reused by several revisers. A prompt improvement can influence every later essay. A definition discovered in one experiment can be placed into a framework file and constrain many subsequent projects. A script can encode a successful procedure so that it no longer needs to be reconstructed manually. The effect of a representation can therefore multiply across later operations.

This is why the repository should not be evaluated merely by counting its finished essays. A terminal essay is one product. A useful prompt, review protocol, evaluation script, invariant representation, or repair procedure may have greater cumulative value because it changes many future trajectories. The repository accumulates not only outputs but reusable transformations.

The distinction resembles the difference between capital and consumption, although the analogy should not be pressed too literally. Some computational outputs are consumed for their immediate result. Others become productive structures that alter the cost or quality of future production. A generated paragraph may be consumed once. A robust prompt template

may participate in hundreds of later generations. A discovered invariant may prevent a recurring conceptual error across several projects. A diagnostic test may reveal the same failure mode repeatedly.

We can therefore speak cautiously of representational capital: durable artifacts whose availability changes the productivity or reachability of future cognitive work. Unlike physical capital, such representations can often be copied at negligible marginal cost. Their scarcity is determined less by physical duplication than by discovery, organization, interpretation, and access.

This returns the experiment to the question of artificial scarcity of knowledge. When a useful computational discovery remains embedded inside a proprietary service, society may receive the service repeatedly without receiving the reusable representation that would allow the method to propagate. The capability accumulates within an organization while its outputs diffuse externally. This is economically valuable but epistemically different from diffusion of the method itself.

The distinction can be stated without assuming that every proprietary mechanism should be disclosed or that every user must become an expert. Some knowledge is dangerous, some is legitimately private, some depends upon infrastructure that cannot be meaningfully reproduced from documentation, and some abstractions are beneficial precisely because they prevent users from confronting unnecessary complexity. The relevant question is not whether all mechanisms must be exposed. It is whether technological progress systematically increases public capability or merely increases public dependence upon privately controlled capability.

The local-model pipeline occupies one end of this spectrum. The models themselves remain complex artifacts whose training cannot be reconstructed from the repository, but the orchestration surrounding them is almost completely exposed. The investigator can see which model performed which operation, inspect the prompt, examine the intermediate state, replace the component, rerun the stage, or preserve the result independently of the model that produced it. The useful organization is therefore not enclosed inside a single application.

This modularity produces a form of institutional resilience at microscopic scale. If Granite is replaced, the repository can survive. If Ollama is replaced by another local inference mechanism, the conceptual stages can survive. If the review prompt changes, earlier reviews remain available. If the 3B model proves inadequate for revision, another model can occupy that role without requiring the essay archive to be discarded. The persistence of the system depends less upon the permanence of any one component.

This is continuation through replaceability. The identity of the project resides partly in its preserved constraints and history rather than in the continued existence of a particular compu-

tational implementation. A component can disappear while the trajectory survives.

The same principle is familiar in long-lived technical institutions. Programming languages survive compiler implementations. Scientific theories survive particular textbooks. File formats survive individual applications. Mathematical notation survives the people who introduced it. Durable systems externalize enough of their organizing structure that component replacement does not require reconstruction from nothing.

The repository therefore provides a small example of why standards and interfaces matter to knowledge diffusion. Plain text, Markdown, shell scripts, Git repositories, and command-line programs are not merely aesthetic preferences. They lower the cost of substitution because their interfaces are widely understood and independently implementable. A proprietary binary project file might offer richer features while making the experimental history more dependent upon one application.

This does not make Unix tools universally superior. Their importance here is architectural. They permit the surrounding organization of the experiment to remain simpler and more durable than the generative components it coordinates. The most sophisticated objects in the system, the language models, can therefore be treated as replaceable processes rather than as the permanent containers of the project's state.

The resulting inversion is worth emphasizing. In many AI systems, the model is treated as the persistent center while conversations, tool outputs, and intermediate states are temporary. In the present architecture, the repository is the persistent center and the models are transient workers. The long-lived object is not the artificial agent. It is the evolving representational environment.

This inversion may offer a useful design principle for computational research more generally. Models are changing rapidly. A workflow whose identity depends upon one model's undocumented behavior may age poorly. A workflow whose state resides in inspectable artifacts and whose model interfaces are explicit can incorporate stronger components as they become available without surrendering its history.

It also changes the interpretation of model improvement. When a new model appears, the question is not merely whether it should replace the old model everywhere. It can be tested at individual boundaries. Perhaps it is dramatically better at mathematical review but only marginally better at drafting. Perhaps it is slower but sufficiently reliable at final repair to reduce total iteration cost. The architecture converts model progress into a local allocation problem.

This creates a pathway by which technological improvement can diffuse through an existing system incrementally. The entire apparatus need not be rebuilt around every new model. New

competence can enter at the location where it changes reachable trajectories most effectively. The rest of the system remains continuous.

The concept of continuation therefore links the practical and theoretical sides of the experiment. The essay continues from topic to outline, draft, review, and revision. The repository continues across commits. The pipeline continues across model substitutions. The research program continues across individual essays. At every scale, persistence depends upon preserving enough structure that change does not require complete reconstruction.

Repair is the mechanism that allows such continuation to remain adaptive rather than merely conservative. A system that preserves everything without correction accumulates errors. A system that replaces everything whenever an error is found loses identity and provenance. Repair occupies the region between these extremes. It changes enough to restore admissibility while preserving enough to maintain continuity.

This is why the apparently mundane instruction not to overwrite intermediate files has theoretical significance. Destructive replacement collapses the distinction between repair and regeneration. Once the predecessor disappears, it becomes harder to determine whether the new state preserved anything important. Persistent history makes repair observable as a relation.

The same principle applies to intellectual traditions. A theory develops through criticism only when enough of its prior structure remains visible to determine what changed. If every revision rewrites the history as though the latest formulation had always existed, conceptual development becomes difficult to reconstruct. Conversely, refusing to alter inherited formulations in order to preserve historical purity prevents correction. Continuation requires both memory and revision.

The pipeline embodies this balance in primitive computational form. It preserves old artifacts while allowing new ones to supersede them operationally. Git makes historical existence monotonic even while current content changes non-monotonically. The resulting architecture can therefore accumulate corrections without pretending that earlier states never occurred.

This may be the most important connection between the essay experiment and the broader theoretical framework. The central object is not a state but a trajectory whose identity depends upon relations among states. Quality cannot be evaluated solely by examining the terminal artifact because some relevant properties concern how that artifact was reached. A final essay produced by indiscriminate regeneration may resemble one produced through careful repair, but the latter possesses a recoverable history of diagnosis and preservation that the former lacks.

For publication, that difference may sometimes be irrelevant. Readers may reasonably care

only about the final text. For research into computational writing, however, the difference is essential because the trajectory is the experimental evidence.

The pipeline should therefore resist the temptation to become merely a more elaborate content-production machine. Its value lies in preserving distinctions that high-throughput generation tends to erase: plan versus realization, defect versus diagnosis, diagnosis versus warrant, repair versus replacement, transient capability versus persistent representation, model competence versus system competence, and output accumulation versus capability diffusion.

Those distinctions provide the basis for a more general conception of artificial intelligence as one component of cumulative technical culture. The strongest model need not contain the entire process. It can contribute bounded transformations to an environment whose history, interfaces, and constraints outlive it. Intelligence becomes useful not only when it produces answers but when its useful consequences can enter structures that others can inspect, contest, repair, and extend.

16.1 Formalization: Diffusion of Competence Through Persistent Representations

Let agent or model A possess intrinsic capability set

$$\mathcal{C}_A.$$

Let environment E contain persistent representations

$$\mathcal{Z}_E = \{Z_1, \ldots, Z_n\}.$$

The effective reachable set of agent A in environment E is

$$\mathcal{R}(A, E).$$

In general,

$$\mathcal{R}(A, E) \neq \mathcal{C}_A,$$

because reachable transformations depend upon both intrinsic competence and environmental information.

Suppose stronger model B produces representation

$$Z_B = \sigma(B, X)$$

from task state X.

The environment changes from

$$E$$

to

$$E' = E \cup \{Z_B\}.$$

If

$$\mathcal{R}(A, E') \supsetneq \mathcal{R}(A, E),$$

then the externalized representation has increased the effective capability of the composite system without altering the parameters of A.

This motivates the definition

$$\Delta\mathcal{R}_A(Z_B) = \mathcal{R}(A, E \cup \{Z_B\}) \setminus \mathcal{R}(A, E).$$

The quantity

$$|\Delta\mathcal{R}_A(Z_B)|$$

is a conceptual measure of the additional reachable region induced by the representation.

Complete competence transfer would require

$$\mathcal{R}(A, E \cup \{Z_B\}) \supseteq \mathcal{R}(B, E)$$

for the relevant task family.

There is no reason to expect this generally.

Instead, externalization usually produces partial diffusion:

$$\mathcal{R}(A, E) \subsetneq \mathcal{R}(A, E \cup \{Z_B\}) \subsetneq \mathcal{R}(B, E).$$

The first strict inclusion means that the representation helps the weaker component. The second means that the representation does not fully substitute for the stronger component.

Theorem 16.1 (Externalized information can enlarge effective reachability without increasing intrinsic model capacity)**.** *Let model A have fixed parameters and let task state X admit target transformation Y such that*

$$Y \notin \mathcal{R}(A, E).$$

Suppose another model B produces persistent representation Z for which

$$Y \in \mathcal{R}(A, E \cup \{Z\}).$$

Then the effective reachable set of the composite system has increased while the intrinsic parameters of A remain unchanged.

Proof. By assumption,

$$Y \notin \mathcal{R}(A, E)$$

and

$$Y \in \mathcal{R}(A, E \cup \{Z\}).$$

Therefore

$$\mathcal{R}(A, E) \subsetneq \mathcal{R}(A, E \cup \{Z\})$$

with respect to at least target Y. Since the parameters of A are unchanged, the enlargement cannot be attributed to intrinsic model modification. It results from the changed informational environment. □ □

This theorem captures the narrow sense in which a strong review can diffuse useful competence into a weaker revision stage.

The value of a representation depends upon recipient capability. Let

$$V(Z, A, E)$$

denote the increase in expected task utility caused by supplying Z to A:

$$V(Z, A, E) = \mathbb{E}[Q(A(X \mid E, Z))] - \mathbb{E}[Q(A(X \mid E))].$$

There is no general identity

$$V(Z, A, E) = V(Z, A', E).$$

Thus representation value is relational.

A representation can satisfy

$$V(Z, A_1, E) > 0,$$

$$V(Z, A_2, E) \approx 0,$$

and even

$$V(Z, A_3, E) < 0$$

if it confuses or misdirects the recipient.

This yields a recipient-compatibility condition for diffusion:

$$D(Z, A) = \mathbf{1}[V(Z, A, E) > 0].$$

More generally, let

$$\eta_D(Z, A) = \frac{V(Z, A, E)}{C(Z)}$$

measure useful downstream gain relative to the cost of producing and transmitting Z.

The strongest representation is therefore not necessarily the longest or most sophisticated. It is the one that produces the greatest useful enlargement of downstream reachability relative to cost.

Now consider repeated reuse. Suppose representation Z is produced once at cost

$$C_p(Z)$$

and subsequently used by agents

$$A_1, \ldots, A_m.$$

The cumulative downstream value is

$$V_{\text{cum}}(Z) = \sum_{i=1}^{m} V(Z, A_i, E_i) - C_p(Z) - \sum_{i=1}^{m} C_{r,i}(Z),$$

where $C_{r,i}$ is retrieval and interpretation cost.

A representation becomes a form of productive persistent structure when

$$V_{\text{cum}}(Z) > 0$$

and continues to contribute across multiple future transformations.

This motivates a formal definition of representational capital.

Definition 16.2 (Representational capital)**.** *A persistent representation Z is representational capital relative to task family $\mathcal{T}$ when its continued availability increases expected future reachable utility across more than one transformation:*

$$\sum_{t \in \mathcal{T}} V(Z, A_t, E_t) > 0.$$

A prompt template, validated procedure, diagnostic test, mathematical definition, or reusable review protocol can therefore function as representational capital even though it is inexpensive to copy.

Its scarcity lies primarily in the cost of discovering a useful representation:

$$C_{\text{discover}}(Z) \gg C_{\text{copy}}(Z)$$

for many textual and software artifacts.

This inequality has consequences for diffusion. Once discovered,

$$C_{\text{copy}}(Z) \to 0$$

approximately, while exclusion can maintain restricted access artificially.

Let population of potential recipients be

$$\mathcal{P} = \{A_1, \ldots, A_N\}.$$

If representation Z is openly available, potential aggregate reachability gain is

$$G_{\text{open}} = \sum_{i=1}^{N} \max(0, V(Z, A_i, E_i)).$$

If access is restricted to subset

$$\mathcal{P}_r \subset \mathcal{P},$$

then

$$G_{\text{restricted}} = \sum_{A_i \in \mathcal{P}_r} \max(0, V(Z, A_i, E_i)).$$

Provided at least one excluded recipient would benefit,

$$G_{\text{restricted}} < G_{\text{open}}.$$

This does not establish that unrestricted publication is always normatively preferable, because safety, privacy, incentives, and other constraints may alter the admissible policy set. It establishes only that restriction and diffusion are structurally different modes of capability distribution.

The distinction between service diffusion and competence diffusion can also be formalized.

Let system B provide output

$$y = B(x)$$

to user A.

Under pure service access, the user receives

$$y$$

but no reusable representation Z sufficient to alter future independent reachability:

$$\mathcal{R}(A, E \cup \{y\}) \approx \mathcal{R}(A, E)$$

outside the immediate task.

Under competence diffusion, the interaction yields representation Z such that

$$\mathcal{R}(A, E \cup \{Z\}) \supsetneq \mathcal{R}(A, E)$$

for some future task family.

The immediate outputs can have equal utility while their cumulative consequences differ:

$$U^{\text{service}}_{\text{immediate}} \approx U^{\text{diffusion}}_{\text{immediate}},$$

yet

$$U^{\text{diffusion}}_{\text{future}} > U^{\text{service}}_{\text{future}}.$$

The difference is the persistence of transferable structure.

Now consider component replacement. Let system

$$S = (E, M)$$

contain persistent environment E and model M.

Replace M by M':

$$S' = (E, M').$$

If the project's identity-bearing state is contained primarily in E, then replacement can preserve substantial continuity:

$$I(S, S') \approx I(E, E)$$

despite

$$M \neq M'.$$

If instead project state is encoded primarily in inaccessible model-specific context, replacement produces larger discontinuity.

Define replacement resilience

$$\mathcal{K}(S, M \to M') = \frac{|\mathcal{I}_{\text{preserved}}|}{|\mathcal{I}_{\text{required}}|},$$

where $\mathcal{I}_{\text{required}}$ is the set of identity-bearing constraints required for continuation.

Architectures that externalize those constraints seek

$$\mathcal{K} \to 1.$$

This produces a general design principle:

Persist the project; replace the processor.

The processor may be a language model, human collaborator, compiler, symbolic engine, retrieval service, or other computational component. What matters for continuation is that the history and constraints required by future work are not unnecessarily trapped inside the transient component.

Finally, consider cumulative competence across time. Let

$$E_t$$

be the persistent environment after experiment t, and let useful artifact Z_t be admitted when it survives evaluation.

Then

$$E_{t+1} = E_t \cup \{Z_t\}$$

for accepted representations, while superseded representations remain historically recoverable even if no longer active.

Effective system competence is

$$C_t^{\text{eff}} = \Phi(M_t, E_t).$$

Even if model capacity remains fixed,

$$M_{t+1} = M_t,$$

the system can improve when

$$\Phi(M_t, E_{t+1}) > \Phi(M_t, E_t).$$

Likewise, when a stronger model becomes available,

$$M_{t+1} > M_t,$$

its contribution compounds with the accumulated environment rather than replacing it.

This yields two independent sources of progress:

$$\Delta C = \Delta C_{\text{model}} + \Delta C_{\text{environment}} + \Delta C_{\text{interaction}}.$$

Contemporary discussion concentrates heavily upon

$$\Delta C_{\text{model}}.$$

The pipeline is designed to investigate

$$\Delta C_{\text{environment}},$$

the increase in effective competence produced by accumulating useful external representations.

The interaction term may ultimately be the most consequential:

$$\Delta C_{\text{interaction}} = \Phi(M', E') - \Phi(M', E) - \Phi(M, E') + \Phi(M, E).$$

It measures the possibility that stronger processors become disproportionately useful when embedded within better accumulated environments, and that better environments become disproportionately useful when interpreted by stronger processors.

The final theoretical claim of the experiment can therefore be stated without assuming that model capability is unimportant:

Technological capability becomes cumulative competence only when enough of its useful consequences survi

A larger model is a temporary concentration of computational capability. A repository can convert selected consequences of that capability into persistent structure. A review can become a repair constraint. A successful prompt can become a reusable procedure. A failed experiment can become diagnostic evidence. A definition can become an invariant. A script can turn a manually discovered sequence into an executable method.

In each case, the important transition is the same:

$$\text{capability} \longrightarrow \text{externalized representation} \longrightarrow \text{future reachability}.$$

The essay pipeline is a deliberately small environment in which this transition can be observed. Its significance lies not in claiming that four shell-mediated model calls constitute intelligence, but in showing how intelligence, wherever it is temporarily available, can leave behind structures through which later processes need not begin again from nothing.

17 Conclusion: The Repository as the Continuing Object

This project began with a deliberately modest engineering question: given two locally available language models of unequal capacity, could they be arranged into a useful essay-writing process without constructing a large agent framework, depending upon a proprietary orchestration service, or treating the strongest available model as the appropriate component for every operation? The resulting implementation used ordinary tools. Bash coordinated the stages. Python performed limited prompt rendering. Markdown stored the intermediate representations. Ollama supplied local inference through Granite 4.1 3B and Granite 4.1 8B. Git preserved changes. The filesystem supplied the memory through which one transient computation could become the input to another. None of these components was individually unusual. The experiment became interesting because of the relations established among them.

The initial architecture separated outlining, drafting, reviewing, and revision. That separation immediately changed what could be observed. Instead of receiving a single terminal essay and attempting to infer how it had been produced, the investigator could inspect the conceptual plan from which the draft developed, compare that plan with its prose realization, examine a stronger model's explicit diagnosis of the resulting weaknesses, and then determine whether a subsequent reviser actually performed the requested repairs. Operations that would ordinarily disappear inside one generation became persistent transitions between identifiable states.

This simple decomposition exposed distinctions that are easy to collapse when language-model performance is evaluated only at the terminal output. Generation is not diagnosis. Diagnosis is not repair. Detection of a defect does not itself warrant an intervention. Knowledge of a repair

does not guarantee the capacity to execute it. A fluent revision does not necessarily preserve the distinctions that gave the original artifact its identity. A larger model's superior judgment does not automatically become useful to a smaller model merely because the judgment is supplied as text. These are empirical questions about transitions between representations.

The pipeline consequently became less interesting as an automated writing application than as a small experimental apparatus for studying those transitions. The essay itself ceased to be the sole object of analysis. The relevant object became the trajectory

$$T \longrightarrow O \longrightarrow D \longrightarrow R \longrightarrow F,$$

together with the prompts, scripts, model assignments, intermediate files, failures, and historical revisions through which that trajectory was produced.

Once the trajectory is treated as the object, several familiar assumptions about artificial intelligence become less obvious. The strongest available component need not perform every transformation. The most valuable output of an inference need not be a terminal answer. A smaller model can sometimes become more effective without any change to its parameters because another process has altered its informational environment. A failed output can remain useful when its provenance permits the failure to be localized. A repository can accumulate competence even though none of its individual language-model invocations possesses persistent memory of the entire project.

These observations support a distinction between model capability and system competence. Model capability is located in the transformations a particular model can perform under specified conditions. System competence depends additionally upon orchestration, memory, representation, verification, repair, and the ability to reuse the consequences of previous work. Improving the model can improve the system, but it is not the only way to do so.

The distinction matters particularly for local models. A 3B model and an 8B model are both limited systems. Neither should be treated as an oracle. Their limitations are not merely obstacles to be concealed behind repeated generation. They can instead motivate architectural differentiation. If one model is inexpensive and generative while another is slower but diagnostically stronger, assigning them different roles can become a testable resource-allocation strategy. If the weaker model cannot execute the stronger model's repairs, that failure reveals a boundary of the architecture rather than merely producing another disappointing essay.

The experiments therefore suggest that heterogeneity should be treated as a resource rather than automatically normalized away. Different models, programs, and human interventions need not possess equivalent capabilities in order to participate in the same process. What matters is whether their outputs can cross interfaces in forms usable by the next component. A

Bash script does not understand an argument, but it can reliably preserve the file containing one. Git does not judge whether a revision is philosophically sound, but it can preserve exactly what changed. A language model cannot provide the historical guarantees of Git, but it can interpret semantic relations that would be extremely difficult to encode procedurally. The architecture becomes stronger by refusing to demand that every component perform every kind of work.

This principle places the language model inside a larger computational ecology. The model is not the system. It is a transformation available to the system. Once this is recognized, many operations currently absorbed into prompts can be reconsidered. Deterministic operations can remain deterministic. Persistent state can remain external. Verification can be assigned to tools capable of stronger guarantees. Human judgment can intervene at boundaries where warrant is underdetermined. Language models can be concentrated where flexible semantic transformation is actually required.

The architecture also changes the temporal interpretation of inference. A model invocation is temporary. Its output need not be. When an outline, diagnosis, definition, or procedure is written into a persistent artifact, the consequences of that inference can influence processes that occur after the model has stopped running. This is the elementary mechanism by which the system acquires a history.

History, however, is useful only if it remains discriminating. Accumulating every generated token indiscriminately would produce storage without memory in the stronger functional sense. The repository becomes useful because artifacts occupy identifiable roles and retain relations to the transformations that produced them. The distinction between outline and draft matters. The distinction between draft and review matters. The distinction between review and final revision matters. Git history matters because successive states remain distinguishable rather than being merged into an undifferentiated archive.

Persistence therefore requires structure. A repository containing thousands of unlabeled outputs may preserve bytes while losing the practical ability to recover the relevant trajectory. The future development of the system will consequently require increasingly sophisticated selection as the archive grows. The problem of memory becomes the problem of determining which inherited representations remain relevant to the next transformation.

This limitation does not weaken the central result. It clarifies it. Persistence is necessary for cumulative competence but is not sufficient. A useful continuing system requires persistence, selection, interpretation, and repair. Information must not merely survive; it must remain reachable in forms that can influence future action.

The same qualification applies to externalization. Writing a stronger model's criticism into a

file does not transfer the stronger model into the weaker one. The experiment instead tests a narrower and more interesting possibility: whether some useful consequence of stronger computation can be compressed into an artifact that changes what a weaker process can subsequently accomplish. Where this succeeds, capability has diffused through representation. Where it fails, the failure identifies a boundary between possessing information about a transformation and possessing the competence required to execute it.

This boundary should become one of the principal objects of future experiments. The review–revision interface provides a controlled location at which it can be measured. A frozen draft can receive reviews from different models. A frozen review can be supplied to different revisers. Review formats can be varied while the underlying draft remains fixed. Repair success can be measured against explicit defects and protected invariants. The architecture thereby converts an intuitive claim about "stronger models helping weaker models" into a family of experimentally separable hypotheses.

The same apparatus can test whether stronger models should be concentrated elsewhere. Perhaps the original allocation is wrong. Perhaps the larger model produces greater marginal value during outlining, where early structural decisions propagate through the entire essay. Perhaps revision requires greater capacity than review because recognizing a defect is easier than repairing it without collateral damage. Perhaps a homogeneous 8B pipeline is sufficiently better that asymmetric allocation is not worthwhile. Perhaps a monolithic 8B prompt outperforms the staged system entirely when terminal essay quality is the only criterion.

None of these results would invalidate the experimental method. They would answer the question the architecture was built to expose. The pipeline should therefore not be protected from falsification by its theoretical motivation. Its model assignments, prompts, stage boundaries, and repair policies are hypotheses embodied in software.

This is one reason version control belongs conceptually inside the experiment rather than merely around it. The pipeline itself can undergo the same process of diagnosis and repair as the essays it generates. A prompt can fail. A stage can prove unnecessary. A model assignment can be inefficient. A review representation can lose important information. Each correction can be committed without erasing the configuration that preceded it. The apparatus therefore develops as a trajectory alongside the artifacts it produces.

There is a useful asymmetry here. The current state of the repository is revisable, but its historical states need not be destroyed in order for revision to occur. Current truth claims can remain non-monotonic while historical existence remains monotonic. A claim can be rejected without pretending that it was never made. A script can be replaced without losing the experiment that revealed its weakness. This distinction allows correction and continuity to coexist.

The resulting architecture offers a concrete interpretation of the principle that repair preserves difference. Repair cannot be evaluated merely by comparing a defective state with a later state and observing that some loss function decreased. It matters which distinctions survived the transformation. In essay revision, the relevant differences include thesis, terminology, argumentative dependencies, uncertainty, and intentionally nonstandard concepts. A system that removes every difficult distinction can produce extremely smooth prose while failing as a repair mechanism.

For this reason, optimization alone provides an incomplete description of the writing process studied here. The target is not fixed independently of the trajectory. Definitions introduced during drafting constrain later uses. Objections can force the thesis to change. Reviews can reveal that earlier assumptions were inadequate. Human intervention can protect or revise invariants. The admissible region changes as the artifact develops.

Writing is therefore better represented as constrained continuation through a changing space of possible successors. At every stage, many continuations are linguistically possible. Fewer preserve the commitments already established. Fewer still satisfy new evidence or criticism. Revision changes the trajectory without necessarily returning to its beginning. The objective is not to reach a globally optimal string but to continue constructing an artifact whose transformations remain warranted and whose relevant identity survives correction.

The importance of this interpretation extends beyond essays. Software development, mathematical proof, scientific modeling, technical design, and institutional decision-making all involve histories in which intermediate representations create constraints upon future transformations. In each case, the final artifact alone provides an incomplete account of how error was diagnosed, what was preserved, and why a particular repair was chosen.

The local essay pipeline is useful because it makes this structure unusually easy to observe. Its representations are ordinary files. Its transitions are ordinary processes. Its history can be inspected with ordinary version-control commands. There is little infrastructure obscuring the conceptual relations. The experiment can therefore remain small enough to understand while still expressing problems that appear in much larger AI systems.

This simplicity should be preserved as long as possible. There is no need to introduce an agent framework merely because the system contains multiple model calls. There is no need for a database while directories provide adequate state. There is no need for a complex message bus while files provide sufficient interfaces. Infrastructure should be added when an experimental requirement demands it, not because complexity is assumed to be a sign of sophistication.

The use of local models strengthens this methodological advantage. The system can be run, interrupted, modified, repeated, and inspected without depending upon a remote conversational

interface. Model identity can be controlled more directly. Prompts exist as files rather than disappearing into an application. Generated artifacts remain under the investigator's ordinary filesystem. The entire procedure is therefore closer to an executable laboratory notebook than to a sequence of conversations with an opaque service.

The laboratory metaphor should nevertheless remain qualified. The current experiments do not establish objective essay quality, factual reliability, or comparative efficiency. Those claims require external evaluation, repeated sampling, instrumentation, and appropriate controls. The contribution at this stage is architectural and methodological. The system demonstrates how such questions can be made experimentally accessible.

Its most important output may consequently be the set of distinctions that became visible during construction. The experiment began with model size and task allocation. It revealed generation versus criticism, criticism versus repair, detection versus warrant, local repair versus upstream escalation, artifact persistence versus active memory, model capability versus effective reachability, immediate service versus competence diffusion, and terminal quality versus trajectory quality. These distinctions now suggest experiments that were difficult to formulate when the task was represented simply as "ask a model to write an essay."

This is characteristic of useful experimental apparatus. An instrument does not merely provide answers. It makes previously conflated quantities separately observable. Once they can be observed separately, they can be manipulated separately. Once they can be manipulated separately, causal hypotheses become possible.

The broader motivation can therefore be stated in terms of technological inheritance. Computational systems are increasingly capable of producing useful intellectual work, but the social and technical consequences of that capability depend upon what survives each act of computation. If only terminal services survive, capability remains concentrated in the systems that provide them. If methods, diagnoses, representations, procedures, and histories also survive, some fraction of the useful competence can enter the environment from which future work begins.

The essay pipeline does not solve this larger problem. It instantiates it at a scale small enough to manipulate. A stronger local model writes criticism into a file. A weaker model attempts to use it. A human can inspect the boundary. The result can be committed. Another model can later replace either component. Nothing about this process requires the original inference to remain alive.

The persistent object is therefore not the model.

It is not even the essay.

The persistent object is the evolving repository of representations, constraints, failures, repairs, and procedures through which later work becomes reachable from earlier work.

This inversion provides the final conceptual result of the experiment. Intelligence can be transient while intellectual structure persists. Models can be replaceable while projects continue. Errors can be corrected without deleting their history. Stronger capability can sometimes alter weaker processes through externalized representations. A simple filesystem can become a substrate upon which otherwise independent acts of computation acquire continuity.

The practical implementation remains almost embarrassingly ordinary:

$$\text{Bash} + \text{Markdown} + \text{Python} + \text{Ollama} + \text{Git}.$$

Yet this ordinariness is part of the result. Persistent computational inquiry does not necessarily require a monolithic intelligent agent with a unified internal memory. It can emerge from transient heterogeneous processes operating upon durable shared representations.

The resulting design principle is correspondingly simple:

Do not require the intelligence to persist; require the consequences of useful intelligence to remain available f

The experiment began as a way to make several local models write essays together. It ends by suggesting a more general architecture for cumulative machine-assisted inquiry. The important question is not only what a model can produce now, but what structure its work leaves behind from which the next act of reasoning can continue.

17.1 Formal Conclusion: Continuation Under Persistent Externalization

Let transient computational agents be

$$M_1, M_2, \ldots, M_n,$$

and let persistent environment at time t be

$$E_t.$$

Each computational operation produces candidate artifact

$$Z_t = M_t(X_t, E_t).$$

An admission operator

$$\mathcal{P}$$

determines whether and how that artifact enters persistent state:

$$E_{t+1} = \mathcal{P}(E_t, Z_t).$$

The simplest append-only form is

$$E_{t+1} = E_t \cup \{Z_t\},$$

although operationally active state may designate some artifacts as superseded.

Future computation depends upon a selection operator

$$\Gamma_t(E_t)$$

such that

$$X_{t+1} = F(M_{t+1}, \Gamma_t(E_t)).$$

Thus continuity between independent model invocations does not require

$$M_{t+1} = M_t$$

or persistent internal state inside either model.

It requires that sufficient information cross the temporal boundary:

$$I_{\text{required}} \subseteq \Gamma_t(E_t).$$

Define continuation viability

$$\mathcal{V}_t = \mathbf{1}\left[\exists X_{t+1} \in \mathcal{A}_{t+1} \text{ reachable from } \Gamma_t(E_t)\right].$$

A persistent architecture succeeds at continuation when

$$\mathcal{V}_t = 1$$

despite replacement, termination, or failure of individual components.

Let intrinsic model competence be

$$C(M_t)$$

and effective system competence be

$$C_{\text{sys}}(t) = \Phi(C(M_t), E_t, \Gamma_t, \mathcal{T}),$$

where $\mathcal{T}$ denotes the available transformation architecture.

It is therefore possible that

$$C(M_{t+1}) = C(M_t)$$

while

$$C_{\text{sys}}(t+1) > C_{\text{sys}}(t),$$

because useful representations have accumulated in E.

Likewise,

$$C(M_{t+1}) < C(M_t)$$

does not necessarily imply

$$C_{\text{sys}}(t+1) < C_{\text{sys}}(t)$$

if the weaker model operates within a sufficiently improved representational environment.

This does not imply unlimited substitution between environment and intrinsic capacity. There exists, for a given task, a reachability boundary beyond which environmental information cannot compensate for missing transformational competence.

Let

$$Y$$

be required target state.

For model M, define maximal environmentally augmented reachability

$$\mathcal{R}_{\max}(M) = \bigcup_{E \in \mathfrak{E}} \mathcal{R}(M, E),$$

over admissible environments $\mathfrak{E}$.

If

$$Y \notin \mathcal{R}_{\max}(M),$$

then no admissible external representation is sufficient to make M perform the required transformation.

Thus externalization complements rather than abolishes intrinsic capability.

Now define persistent contribution of inference I_t as

$$P(I_t) = \sum_{k>t} \omega_{tk} \Delta Q_k(Z_t),$$

where $\Delta Q_k(Z_t)$ is the marginal improvement in later operation k attributable to retained artifact Z_t.

An inference whose output is immediately useful but never reused can satisfy

$$U_{\text{immediate}}(I_t) > 0$$

while

$$P(I_t) \approx 0.$$

An inference producing a reusable procedure may instead satisfy

$$P(I_t) \gg U_{\text{immediate}}(I_t).$$

This distinction motivates cumulative system value

$$U_{\text{cum}} = \sum_t U_{\text{immediate}}(I_t) + \lambda \sum_t P(I_t).$$

A system optimized solely for terminal responses effectively emphasizes the first term. A persistent research architecture attempts to increase the second without sacrificing the first unnecessarily.

Repair enters through transition

$$\rho : X_t \to X_{t+1}.$$

Let identity-bearing invariant set be

$$I_t.$$

A repair is continuation-preserving when

$$X_{t+1} \in \mathcal{A}_{t+1}$$

and

$$I_t^{\text{required}} \subseteq I_{t+1}.$$

If the repair removes the defect by deleting all relevant identity-bearing structure, then admissibility under a narrow local criterion may improve while continuation fails.

Define continuation-preserving repair as

$$\rho^* = \arg\min_{\rho} C(\rho)$$

subject to

$$\rho(X_t) \in \mathcal{A}_{t+1}$$

and

$$I_t^{\text{required}} \subseteq I(\rho(X_t)).$$

The null intervention remains available:

$$\rho_0(X_t) = X_t.$$

Therefore intervention is justified only when

$$W(\rho^*) > W(\rho_0).$$

The entire essay architecture can finally be represented as a persistent transition system

$$\mathfrak{W} = (E, \mathcal{M}, \mathcal{T}, \mathcal{A}, \mathcal{P}, \Gamma),$$

where E is persistent state, $\mathcal{M}$ is the set of available computational components, $\mathcal{T}$ is the set of transformations, $\mathcal{A}$ specifies admissibility conditions, $\mathcal{P}$ governs persistence, and Γ governs retrieval into active context.

No single element is sufficient for cumulative competence.

Without capable components,

$$\mathcal{T}$$

cannot reach useful new states.

Without persistence,

$$E_{t+1}$$

cannot inherit useful consequences of earlier computation.

Without retrieval,

$$E$$

becomes an archive disconnected from action.

Without admissibility,

$$\mathcal{T}$$

has no principled distinction between continuation and degradation.

Without repair,

$$\mathfrak{W}$$

cannot adapt when inherited representations fail.

The cumulative architecture therefore requires their interaction:

$$\boxed{\text{Capability} + \text{Persistence} + \text{Selection} + \text{Admissibility} + \text{Repair} \longrightarrow \text{Continuation}.}$$

The arrow is not an identity and not a universal sufficiency theorem. It represents the architectural hypothesis exposed by the experiment: durable progress requires more than increasingly powerful transient computation.

Theorem 17.1 (Persistence of useful consequences is necessary for direct cumulative influence across independent inference events)**.** *Let I_t and I_{t+k} be independent inference events with no shared mutable internal model state. Let information Z_t be produced uniquely during I_t and be relevant to the optimal transformation performed at I_{t+k}. If Z_t is neither persistently represented nor reconstructible from information available at I_{t+k}, then Z_t cannot directly condition the later inference. If Z_t is persistently represented and successfully retrieved, direct cumulative influence becomes possible.*

Proof. By assumption, I_t and I_{t+k} share no mutable internal state. Therefore any information transmitted from the former to the latter must cross an external channel. Suppose Z_t is neither persisted nor reconstructible. Then

$$Z_t \notin \Gamma_{t+k}(E_{t+k})$$

and no equivalent representation of Z_t is available. Consequently the conditional distribution governing the later transformation cannot condition upon the information uniquely supplied by Z_t.

If instead Z_t is persisted and retrieved,

$$Z_t \in \Gamma_{t+k}(E_{t+k}),$$

so the later transformation can be conditioned upon it:

$$P(X_{t+k+1} \mid X_{t+k}, Z_t).$$

Persistence does not guarantee improvement because Z_t may be false, irrelevant, or unusable, but it establishes the channel required for direct cumulative influence. □ □

The theorem is intentionally modest. It does not claim that repositories create intelligence or that external memory substitutes for stronger models. It identifies a necessary architectural condition for a particular kind of accumulation. Independent acts of intelligence cannot contribute their unique information directly to later acts unless some representation bridges the interval between them.

The repository is such a bridge.

Its files are imperfect representations. Its prompts are provisional. Its models are replaceable. Its reviews can be wrong. Its revisions can fail. Its history can become difficult to navigate. Yet because these states persist, their defects can themselves become objects of later inquiry.

The system therefore need not be correct at every moment in order to continue learning from its own construction. It requires something both weaker and more demanding: that enough of what happened remain distinguishable for subsequent processes to determine what should be preserved, what should be rejected, and what can still be repaired.

In this sense, the terminal essay is not the end of the experiment. It is another committed state in a longer trajectory.

The experiment continues wherever that state leaves behind enough structure for the next transformation to begin from somewhere other than zero.

References

[1] J. L. Austin, *How to Do Things with Words*, Clarendon Press, Oxford, 1962.

[2] Gregory Bateson, *Steps to an Ecology of Mind*, Chandler Publishing Company, San Francisco, 1972.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei, "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[4] Andy Clark, *Being There: Putting Brain, Body, and World Together Again*, MIT Press, Cambridge, Massachusetts, 1997.

[5] Andy Clark and David J. Chalmers, "The Extended Mind," *Analysis*, vol. 58, no. 1, 1998, pp. 7–19.

[6] Peter J. Denning, "Computing Is a Natural Science," *Communications of the ACM*, vol. 50, no. 7, 2007, pp. 13–18.

[7] Douglas C. Engelbart, *Augmenting Human Intellect: A Conceptual Framework*, Stanford Research Institute, Menlo Park, California, 1962.

[8] Martin Fowler, *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, Reading, Massachusetts, 1999.

[9] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, Massachusetts, 1994.

[10] Emden R. Gansner and Stephen C. North, "An Open Graph Visualization System and Its Applications to Software Engineering," *Software: Practice and Experience*, vol. 30, no. 11, 2000, pp. 1203–1233.

[11] H. P. Grice, *Studies in the Way of Words*, Harvard University Press, Cambridge, Massachusetts, 1989.

[12] Edwin Hutchins, *Cognition in the Wild*, MIT Press, Cambridge, Massachusetts, 1995.

[13] Daniel Kahneman, *Thinking, Fast and Slow*, Farrar, Straus and Giroux, New York, 2011.

[14] Donald E. Knuth, "Literate Programming," *The Computer Journal*, vol. 27, no. 2, 1984, pp. 97–111.

[15] David Lewis, *Counterfactuals*, Harvard University Press, Cambridge, Massachusetts, 1973.

[16] John McCarthy, "Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I," *Communications of the ACM*, vol. 3, no. 4, 1960, pp. 184–195.

[17] Robert K. Merton, *The Sociology of Science: Theoretical and Empirical Investigations*, University of Chicago Press, Chicago, 1973.

[18] Allen Newell and Herbert A. Simon, *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

[19] Elinor Ostrom, *Governing the Commons: The Evolution of Institutions for Collective Action*, Cambridge University Press, Cambridge, 1990.

[20] Nick Pearce, Martin A. Stodden, and collaborators, "Reproducible Research and Computational Experimentation," in the broader literature on reproducible computational science, 2012.

[21] Roger D. Peng, "Reproducible Research in Computational Science," *Science*, vol. 334, no. 6060, 2011, pp. 1226–1227.

[22] Karl R. Popper, *The Logic of Scientific Discovery*, Hutchinson, London, 1959.

[23] Karl R. Popper, *Conjectures and Refutations: The Growth of Scientific Knowledge*, Routledge & Kegan Paul, London, 1963.

[24] Eric S. Raymond, *The Art of Unix Programming*, Addison-Wesley, Boston, 2003.

[25] Dennis M. Ritchie and Ken Thompson, "The UNIX Time-Sharing System," *Communications of the ACM*, vol. 17, no. 7, 1974, pp. 365–375.

[26] Douglas C. Schmidt, "Model-Driven Engineering," *Computer*, vol. 39, no. 2, 2006, pp. 25–31.

[27] Herbert A. Simon, "A Behavioral Model of Rational Choice," *The Quarterly Journal of Economics*, vol. 69, no. 1, 1955, pp. 99–118.

[28] Herbert A. Simon, *The Sciences of the Artificial*, MIT Press, Cambridge, Massachusetts, 1969.

[29] Herbert A. Simon, *The Sciences of the Artificial*, 3rd ed., MIT Press, Cambridge, Massachusetts, 1996.

[30] Diomidis Spinellis, "Version Control Systems," *IEEE Software*, vol. 22, no. 5, 2005,

pp. 108–109.

[31] Victoria Stodden, Marcia McNutt, David H. Bailey, Ewa Deelman, Yolanda Gil, Brooks Hanson, Michael A. Heroux, John P. A. Ioannidis, and Michela Taufer, "Enhancing Reproducibility for Computational Methods," *Science*, vol. 354, no. 6317, 2016, pp. 1240–1241.

[32] Alan M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, no. 236, 1950, pp. 433–460.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[34] Joseph Weizenbaum, *Computer Power and Human Reason: From Judgment to Calculation*, W. H. Freeman, San Francisco, 1976.

[35] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J. G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A. C. 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons, "The FAIR Guiding Principles for Scientific Data Management and Stewardship," *Scientific Data*, vol. 3, article 160018, 2016.

[36] Niklaus Wirth, "Program Development by Stepwise Refinement," *Communications of the ACM*, vol. 14, no. 4, 1971, pp. 221–227.

[37] 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 *International Conference on Learning Representations*, 2023.

[38] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 24824–24837.

[39] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bod-

hisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark, "Self-Refine: Iterative Refinement with Self-Feedback," in *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[40] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R. Narasimhan, and Shunyu Yao, "Reflexion: Language Agents with Verbal Reinforcement Learning," in *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[41] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe, "Training Language Models to Follow Instructions with Human Feedback," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 27730–27744.

[42] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosiute, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan, "Constitutional AI: Harmlessness from AI Feedback," arXiv preprint arXiv:2212.08073, 2022.

[43] Git Project, *Git Reference Manual and Documentation*, software documentation, accessed 2026.

[44] Ollama, *Ollama Documentation*, software documentation, accessed 2026.

[45] Free Software Foundation, *GNU Bash Reference Manual*, GNU Project, accessed 2026.

[46] Python Software Foundation, *Python 3 Documentation*, accessed 2026.