Reflective Interfaces in Language-Model Systems: Sources, Interpretation, and Control

Utkarsh Upadhyay
UTET Partners B.V.
uu@utetpartners.nl

Abstract

Language-model "self-reflection" is attributed to systems that re-read visible text, retrieve earlier episodes, estimate correctness, inspect activation-derived representations, or query mechanistic tools. These systems differ along more than one axis. We propose a reporting schema that first declares the *reflective subject*—the model, a persistent agent, or a larger model–harness system—and then records what information about that subject is available, how it is interpreted, what interface is presented, who receives it, and what action follows. Within this schema, a source-access ladder groups visible context, persistent and retrieved memory, output distributions, execution state, and implementation state by the breadth or depth of system access required. The rungs are conventional access classes rather than a scalar measure of quality or causal proximity. The remaining fields distinguish self-reflective deployments from external monitoring, external control, and endogenous control interfaces that do not represent a property of the subject. We classify representative work in verbal reflection, uncertainty estimation, latent reasoning, and mechanistic interpretability, and derive evaluations of availability, informativeness, grounding, and use. Exogenous signals enter these evaluations as controls, not as self-reflective evidence. The taxonomy concerns operational information channels; it makes no claim about consciousness or subjective experience.

1 Introduction

"Self-reflection" is used for several different phenomena. In agent research it often means generating a textual critique and placing it back into context [Shinn et al., 2023, Madaan et al., 2023]. In uncertainty research it can mean estimating whether an answer is likely to be correct [Kadavath et al., 2022, Farquhar et al., 2024]. In interpretability research it can mean decoding, reporting, or controlling properties of hidden computation [Li et al., 2025, Lindsey, 2025]. These mechanisms may support similar decisions, but they expose different evidence.

The relevant unit is usually not a bare language model. It is a *language-model system*: a model together with prompt construction, storage, retrieval, learned monitors, tools, and decision policies. We call the system about which information is claimed the *reflective subject*. It may be one model invocation, a persistent agent assembled by a harness, or a larger model–harness system. The *interface recipient* is the human or component that receives a readout. A recipient that chooses an action is a *controller*; a passive researcher or evaluator is not. A self-reflection claim therefore requires the subject, recipient, and any controller to be named rather than inferred from first-person grammar.

The access ladder introduced informally in an earlier essay [Upadhyay, 2026] organizes one important question: how deeply does an architecture reach into the model's implementation? It does not by itself specify whether the evidence is about the declared subject, how it is interpreted, what representation reaches a recipient, or whether a controller uses it. For example, a learned monitor may consume thousands of hidden activations yet expose one probability, while a mechanistic service may inspect parameters and Jacobians but answer a natural-language query.

We therefore represent a reflective architecture as

$$\underbrace{\text{reflective subject}}_{\text{declared boundary}} \longrightarrow \text{source} \xrightarrow{\text{interpretation}} \text{interface} \longrightarrow \text{recipient} \xrightarrow{\text{if a controller}} \text{action}. \tag{1}$$

The subject is a reference boundary, not another processing stage. The source ladder summarizes the second term; the remaining terms describe what the architecture makes legible and actionable.

Our contributions are:

1. a role-explicit taxonomy separating reflective subject, information source, interpretation algorithm, presented interface, recipient, and optional controller/action;
2. a source-access ladder spanning visible context, persistent and retrieved memory, output distributions, execution state, and implementation state;
3. four evaluation questions concerning availability, informativeness, grounding, and use; and
4. an experimental agenda for testing availability, predictive informativeness, causal or evidential grounding, and closed-loop use.

2 A Reflective Pipeline

2.1 Subject, harness, and reflection points

Let episode i contain tokens $x_{i,1:T_i}$. A model with parameters θ defines

$$\pi_\theta(x_{i,t+1} \mid x_{i,\le t}) \tag{2}$$

and produces residual-stream states $h_{i,\ell,t}$. Let $\mathcal{D}_{<i}$ denote completed episodes and m_i persistent harness state available when episode i begins. Within episode i, an architecture computes reflective readouts at one or more *reflection points* $d = 1, \dots, D_i$. Reflection point d falls immediately after token $\tau_{i,d} \le T_i$ and fixes what evidence exists when the readout is computed; a readout computed at episode completion has $\tau_{i,d} = T_i$. The schedule of reflection points is part of the architecture: it may be fixed by the harness or chosen by a gating policy, in which case the choice to reflect is itself an action. This separates episode history, reflection time, and token-local computation.

Let B denote the declared reflective-subject boundary. If B is one model invocation, retrieved harness memory is external to that subject. If B is a persistent agent, the same memory can be part of its state. Neither choice is universal, but it must be explicit.

2.2 Operational components

At reflection point d of episode i, equation (1) instantiates as

$$\underbrace{B}_{\text{subject}} \xrightarrow{\mathcal{A}} \underbrace{s_{i,d}}_{\text{source}} \xrightarrow{\mathcal{I}} \underbrace{r_{i,d}}_{\text{readout}} \xrightarrow{\mathcal{P}} \underbrace{z_{i,d}}_{\text{interface}} \longrightarrow \underbrace{\mathcal{R}}_{\text{recipient}} \xrightarrow{\mathcal{C}} \underbrace{a_{i,d}}_{\text{action}}, \tag{3}$$

where the single interpretation arrow of equation (1) resolves into a readout $\mathcal{I}$ and its presentation $\mathcal{P}$, and the final arrow exists only when the recipient is a controller. Written out,

$$s_{i,d} = \mathcal{A}\big(B;\, x_{i,\le\tau_{i,d}},\, h_{i,\cdot,\le\tau_{i,d}},\, \theta,\, m_i,\, \mathcal{D}_{<i}\big)\,, \qquad \text{available source bundle,} \tag{4}$$

$$r_{i,d} = \mathcal{I}(s_{i,d}), \qquad \text{interpretation or readout,} \tag{5}$$

$$z_{i,d} = \mathcal{P}(r_{i,d}), \qquad \text{presented interface,} \tag{6}$$

$$z_{i,d} \longrightarrow \mathcal{R}, \qquad \text{interface recipient,} \tag{7}$$

$$a_{i,d} = \mathcal{C}(z_{i,d}; c_{i,d}), \qquad \text{optional controller and action.} \tag{8}$$

The bundle is what an architecture *could* read at d, not what it does read; $\mathcal{A}$ selects from it, and the ladder of section 3 classifies that selection. Gradients and Jacobians are functions of θ and the execution state rather than additional primitives. Here $c_{i,d}$ collects task context, budget, memory, and other state held fixed when the effect of $z_{i,d}$ is studied.

Two instantiations make the indices concrete. Reflexion has one reflection point per episode, at $\tau_{i,d} = T_i$, whose action writes the memory m_{i+1} consumed by a later episode. An uncertainty-gated delegator has a reflection point before finalization, at $\tau_{i,d} < T_i$, whose action is taken inside episode i.

Reflective subject. What system is the information about? This boundary B may enclose a model invocation, a persistent agent, or a larger model–harness system.

Source. What information about B is available at reflection point d, and through what acquisition process? Examples include subject-produced text in the visible prefix, a persistent summary, selected trajectories, output probabilities, hidden activations, attention patterns, parameters, gradients, and Jacobians. The containing source bundle may also include ordinary exogenous task context.

Interpretation algorithm. How is the acquired source transformed? The operation may be identity, entropy calculation, a learned probe, a sparse autoencoder, an auxiliary correctness predictor, circuit tracing, a Jacobian lens, or an LLM interpreting the response of a mechanistic tool. Retrieval is source selection; interpreting the returned records is a separate operation.

Presented interface. What representation reaches a recipient across a declared component boundary? Possibilities include discrete text, a category, a probability, a low-dimensional vector, a soft token passed into a later model step, sparse feature coordinates, an attribution graph, or a queryable API. Interface breadth is functional rather than dimensional: a fixed-purpose correctness score answers one preselected question, whereas a queryable mechanistic interface lets its recipient choose what to inspect.

Recipient, controller, and action. Who receives the interface, and does that recipient act? A recipient may be a passive researcher, an external monitor, a separate policy, the target model in a later pass, or the target model during the current request. A controller may answer, abstain, retrieve, delegate, allocate computation, alter an activation, or propose a parameter edit. If the recipient or controller is external to B, the architecture monitors or controls the subject from outside. If the channel is returned to B or one of its internal control components, it can participate in self-reflection.

This decomposition prevents implementation pedigree from determining the system's label. A tool may use high-bandwidth internal state while exposing a narrow interface; source richness, recipient legibility, and downstream use are separate properties.

2.3 Four evaluation questions

A reported architecture can be evaluated with four separable diagnostic questions:

1. **Availability.** Who can access the candidate source under the declared boundary, and at what implementation depth?
2. **Informativeness.** Does the interpretation map that information to the stated task-relevant variable?
3. **Grounding.** Does the readout depend on the purported subject-derived signal rather than an alternative cue? Stronger mechanistic claims additionally require causal attribution.
4. **Use.** Does a recipient interpret the interface, and does a controller use it to change a report, decision, or internal state?

These are evidential questions, not pipeline stages and not a maturity scale. Some tests presuppose others: causal use of a readout is uninformative if the readout does not predict its stated target. A probe can establish informativeness without privileged availability to the subject. An injected sentence can alter an action without reading the subject's state. A calibrated monitor can remain behaviorally irrelevant when no controller uses it.

2.4 Channel eligibility and interface breadth

The schema applies broadly to *candidate self-information architectures*. A particular deployment counts as self-reflective only when its channel represents a stated property of B, that channel value causally depends on B, and the resulting interface is available to B or one of its internal control components for subject-directed interpretation, reporting, or action. External probes and human-facing mechanistic tools are comparison

architectures unless their outputs are returned across that boundary. Ordinary hidden-state propagation is not self-reflection merely because endogenous state affects later computation; a subject-directed representation or inference must be identified.

A source bundle may also contain *exogenous* information supplied independently, such as task instructions or external feedback. Such context may influence the same controller, but that influence is a control condition or ordinary input rather than self-reflective evidence. Thus the sentence "I am uncertain" is not intrinsically introspective. Its provenance and deployment determine whether it presents a subject-derived estimate or an experimenter-selected intervention.

3 The Source-Access Ladder

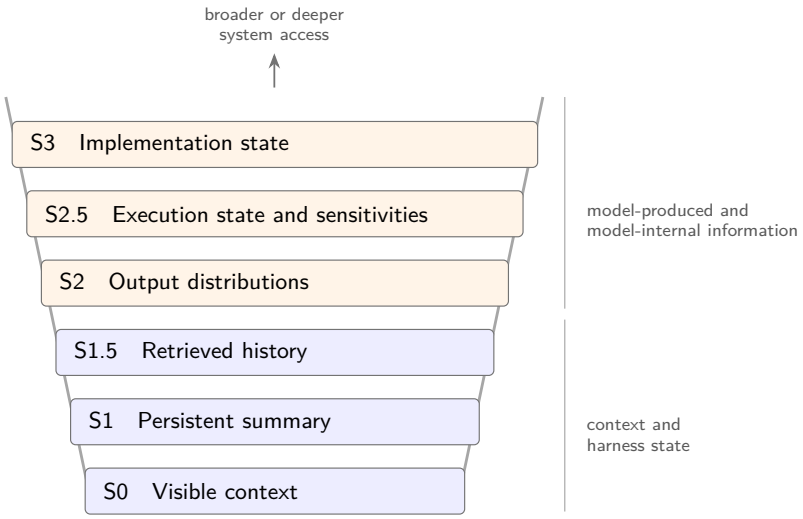

Figure 1: The source-access ladder. Height groups conventional classes that require broader or deeper system access, from context and harness records to model outputs, active execution state, and durable implementation state. Rung numbers do not denote quality or a strict scalar order.

The ladder in figure 1 is a compact set of *access classes*. Moving upward generally requires broader or deeper system access: from public or harness-mediated records, through model outputs, to active state and durable implementation state. The sequence is not a total mathematical order. In particular, a persistent summary and a retrieved archive differ by selection and retention; their plotted order records broader archive access, not deeper implementation access. Systems may expose several classes simultaneously. Comparisons must hold the declared subject boundary B fixed; the rung numbers do not license comparisons that silently switch from an invocation to a persistent agent.

3.1 S0: visible-context access

At S0, the architecture uses only evidence visible in the current request. The request can mix exogenous instructions with subject-derived evidence, such as a previous answer produced by B and placed back into context. Only the latter can carry self-information about B. An observer can be given the same public transcript; any advantage over that matched observer must therefore come from a source other than visible text. First-person grammar alone does not establish such an advantage.

3.2 S1: persistent-summary access

At S1, a harness maintains a compact summary m_i and exposes it during later episodes:

$$m_{i+1} = U(m_i, \mathcal{D}_i), \qquad y_{i+1} \sim \pi_\theta(\cdot \mid x_{i+1}, m_{i+1}). \tag{9}$$

Reflexion stores verbal feedback in episodic memory and conditions later attempts on it [Shinn et al., 2023]. Generative Agents synthesize reflections over observations and return them to a memory stream [Park et al., 2023]. A summary is internal only when B is declared to include the persistent harness; external feedback inside that summary remains exogenous evidence even then. Such systems support cross-episode adaptation without weight updates. Their bottleneck is selection: summaries can omit counterevidence, preserve stale conclusions, or turn transient behavior into a stable self-description.

3.3 S1.5: retrieved-history access

At S1.5, the system can revisit records that were not selected for a standing summary. Given query $q_{i,d}$, corpus $\mathcal{D}_{<i}$, and requested result count k,

$$s^{\text{ret}}_{i,d} = \text{Retrieve}(q_{i,d}, \mathcal{D}_{<i}, k)\,. \tag{10}$$

Retrieval must complete strictly before the reflection point it serves, so that the selected records are already in the bundle when the readout at d is computed; a retrieval issued at d itself could not inform that readout. Retrieval selects a source; a later interpretation maps the selected records to a task-relevant readout. As at S1, whether the archive lies inside B depends on whether B includes the persistent harness. This supports longitudinal audit and evidence-backed correction, but similarity does not imply relevance. Evaluations should include stale records, adversarially similar episodes, and tasks requiring reconstruction across several interactions.

3.4 S2: output-distribution access

At S2, the source includes quantities produced at the model's output boundary, such as token probabilities or samples from repeated generations. A probability from the target invocation causally depends on that invocation. Repeated-sample methods require B to include the sampling procedure or collection of invocations; if B is one invocation, the other samples are observer-accessible evidence about it rather than its internal state. Interpretation algorithms can turn output evidence into token entropy, sequence-level confidence, verbalized uncertainty, or semantic entropy [Kadavath et al., 2022, Lin et al., 2022, Kuhn et al., 2023, Farquhar et al., 2024]. These quantities differ in meaning and cost. Token entropy measures uncertainty over the next surface token; semantic entropy groups sampled generations by meaning.

A statistic becomes a candidate reflective interface only when a recipient receives it across a declared boundary. It becomes part of self-reflection only when returned to B or an internal control component under the eligibility rule above. Rendering a probability as text changes the presentation, not the source. Its usefulness must be evaluated under calibration, selective risk, and decision cost rather than raw accuracy alone.

3.5 S2.5: execution-state access

At S2.5, the source includes objects from the current forward or backward computation: hidden activations, attention patterns, gradients, or Jacobians. Many interpretation algorithms operate here. Unsupervised latent-knowledge probes, tuned lenses, Patchscopes, sparse feature dictionaries, and activation interventions expose different functions of hidden state [Burns et al., 2023, Belrose et al., 2023, Ghandeharioun et al., 2024, Bricken et al., 2023, Li et al., 2023, Turner et al., 2023, Zou et al., 2023].

Hidden-state access does not determine interface width. Semantic entropy probes estimate semantic uncertainty from a single generation's hidden states [Kossen et al., 2024]. Gnosis passively observes final-layer hidden-state trajectories and layer–head attention maps, processes them through learned encoders and a fusion predictor, and exposes a scalar correctness estimate while leaving the backbone frozen [Ghasemabadi and Niu, 2025]. Both consume execution state but produce fixed-purpose readouts; in their monitoring deployments, an external recipient rather than B receives the result.

Anthropic's interpretability sequence illustrates several other interpretation objects. Sparse autoencoders identify feature coordinates [Bricken et al., 2023]; Scaling Monosemanticity extends feature extraction to Claude 3 Sonnet [Templeton et al., 2024]; circuit tracing connects active features into prompt-specific

attribution graphs [Ameisen et al., 2025, Lindsey et al., 2025]. Human-facing, external-model, and target-model-facing deployments answer different availability and use questions even when they expose the same artifact.

3.5.1 Jacobians and J-space as interpretation methods

An activation records the state occupied by a computation. A derivative describes how a local change would propagate. For intermediate state $h_{i,\ell,t}$ and a later final-layer state $h_{i,L,t'}$, the Jacobian lens constructs an average downstream map

$$J_\ell = \mathbb{E}_{i,t,t' \geq t} \left[\frac{\partial h_{i,L,t'}}{\partial h_{i,\ell,t}} \right] \tag{11}$$

and reads the transported state through

$$\text{softmax}(W_U \, \text{norm}(J_\ell h_{i,\ell,t})) \,, \tag{12}$$

where W_U is the unembedding matrix [Gurnee et al., 2026]. J-space uses sparse non-negative combinations of token-associated J-lens directions. It is an interpretation method over execution state and derivative information.

The averaged map describes typical first-order downstream transport, not the exact local effect for every request. Prompt-local backward Jacobians also appear in circuit tracing, where they estimate influence between features [Ameisen et al., 2025]. The J-space work instead uses averaged transport to define readout coordinates. A self-information architecture might expose J-lens scores, sparse coordinates, local Jacobian-vector products, or a queryable service; the interface, recipient, and any controller must still be specified.

Recent work provides limited evidence that models can consume activation-derived signals about themselves. Models can learn through neurofeedback to report and modulate selected activation projections [Li et al., 2025]; concept-injection experiments show model- and context-dependent identification of injected activation patterns [Lindsey, 2025]; and models can describe dispositions introduced by fine-tuning [Betley et al., 2025].

3.6 S3: implementation-state access

At S3, the source includes parameters or durable functions of them. Raw tensors are not automatically a useful interface: representations are distributed and superposed [Elhage et al., 2022]. Practical systems are more likely to expose parameter tensors, low-rank parameter decompositions, parameter-derived diagnostics, or candidate weight edits [Dai et al., 2022, Meng et al., 2022]. Execution-conditioned gradients, feature activations, and prompt-specific circuit graphs remain at S2.5.

Reading implementation state is monitoring; applying an edit is control. Weight updating alone is ordinary learning. A stronger self-reflective deployment would require a parameter-derived diagnosis about B to be interpreted and used by an internal controller, which predicts the effect of a constrained edit and verifies both the intended change and side effects.

4 Classifying Existing Systems

Table 1 applies the schema to candidate self-information architectures. It separates the declared subject and information source from deployment status; the same method can support external monitoring, external control, or self-reflection under different boundaries.

Our framework is complementary to two recent accounts with broader aims. Liu et al. [2026] survey the wider literature on metacognition in LLMs, including its measurement, elicitation, improvement, and applications. Naphade et al. [2026] instead formalize policy and mechanistic introspection and introduce an empirical benchmark for those capabilities. The present paper focuses more narrowly on the information flow and deployment roles of reflective interfaces across model–harness systems. We refer readers to those works for broader reviews of LLM metacognition and introspection.

Soft Thinking is useful as a boundary case even though it is not primarily a self-reflection paper. It converts the next-token distribution into a probability-weighted mixture of token embeddings and feeds that continuous "soft token" into the same model's next reasoning step [Zhang et al., 2025]. It demonstrates an

A. Evidence pipeline

Method	Citation	Declared subject B	Available information ("what")	Transformation ("how")	Interface
Reflexion	Shinn et al., 2023	Persistent agent plus memory harness	Completed trajectories and feedback	Summary construction and record selection	Text memory
Semantic entropy	Farquhar et al., 2024	Sampling procedure / set of runs	Repeated output samples	Semantic clustering and entropy	Scalar uncertainty
Semantic-entropy probes	Kossen et al., 2024	Target invocation	Hidden states	Learned probe	Scalar uncertainty
Gnosis	Ghasemabadi and Niu, 2025	Target invocation	Hidden trajectories and attention	Learned encoders and fusion	Correctness probability
Sparse autoencoders	Bricken et al., 2023, Templeton et al., 2024	Target model	Hidden activations	Sparse dictionary learning	Feature activations / labels
Circuit tracing	Ameisen et al., 2025, Lindsey et al., 2025	Target invocation	Features and local derivatives	Attribution-graph construction	Directed feature graph
J-lens / J-space	Gurnee et al., 2026	Target model instantiated on the queried input	Activations and averaged Jacobians	Transport and sparse coding	Token readouts / coordinates
Soft Thinking	Zhang et al., 2025	Active invocation	Next-token distribution	Weighted embedding mixture	Continuous soft token

B. Deployment and status

Method	Recipient / controller	Scope shown	Action or use demonstrated	Status in cited setup
Reflexion	Same agent, later attempt	Retrospective	Text memory changes a later attempt	Self-reflective deployment
Semantic entropy	External monitor by default	Concurrent / post hoc	Monitoring unless connected to a controller	External monitoring
Semantic-entropy probes	External monitor by default	Concurrent	Monitoring unless connected to a controller	External monitoring
Gnosis	External monitor	Concurrent	Correctness prediction; no internal loop	External monitoring
Sparse autoencoders	Researcher in cited setup	Post hoc	Feature extraction and interpretation	External analysis
Circuit tracing	Researcher in cited setup	Post hoc	Graph construction and causal checks	External analysis
J-lens / J-space	Researcher in cited setup	Concurrent / post hoc	Readout and targeted modulation	External analysis / control
Soft Thinking	Same model's next step	Concurrent	Changes the next reasoning step	Endogenous control; reflection not established

Table 1: Representative candidate self-information architectures. Panel A reports subject, available information, transformation, and interface; panel B reports recipient, temporal scope, demonstrated use, and deployment status. Entries report the cited setup; where a cited study evaluates both analysis and intervention, both are named.

endogenous, continuous, intra-request control interface. It does not by itself establish self-reflection because no subject-directed property is interpreted.

Temporal scope is an independent reporting axis, and the reflection point makes it precise. *Retrospective* architectures draw the source at d from completed episodes $\mathcal{D}_{<i}$ to affect a later one, as in Reflexion. *Concurrent* architectures draw it from the active computation of episode i with $\tau_{i,d} < T_i$, as in Soft Thinking or an online uncertainty controller. *Prospective* architectures state a property $q(B)$ concerning behavior after $\tau_{i,d}$, such as whether an answer will be correct or how the model will respond [Binder et al., 2025, Naphade et al., 2026]. Each scope can pair with more than one source, interface, and deployment status.

5 An Experimental Agenda

Throughout this section we suppress the indices (i, d) where a statement holds at every reflection point, and restore them where an expectation is taken. The Δ quantities below are defined per reflection point and reported as averages over them.

5.1 Availability and privileged-access evaluation

A privileged-access claim is strongest when the reflective subject outperforms an observer with matched public evidence and no greater computational cost. For subject M, observer O, self-property q, and ground truth y, compare

$$\Delta_{\text{priv}} = \text{score}(M(q; s_M), y) - \text{score}(O(q; s_O), y), \tag{13}$$

where s_O includes the transcript, task data, and examples available to M but excludes the candidate privileged source. Cross-model self-prediction uses this logic [Binder et al., 2025]; cost-matched observer controls make the criterion stricter [Song et al., 2025].

Controls should match the source. Memory systems require observers with the same records. Output-distribution systems require matched samples or probabilities where possible. Execution-state systems require comparisons with another model's state, stale states, shuffled layers, or matched random projections. Model identity is a useful observer input because it supports inference from known model behavior, but identity alone does not expose the subject's current internal state. In the sampling-temperature task, apparent self-reporting did not outperform a cost-matched third party strongly enough to satisfy privileged self-access [Song et al., 2025].

This evaluation answers *availability*: who can access the candidate evidence under matched public information? Privilege is one comparative result, not the definition of availability.

5.2 Informativeness and readout evaluation

Given a stated property $q(B)$ and readout $r = \mathcal{I}(s)$, evaluate predictive validity on held-out examples before asking whether the readout is privileged or causally used. Report a proper scoring rule when r is probabilistic, calibration and selective risk when the interface supports abstention or escalation, and class-conditional metrics when the target is imbalanced. Compare against public-context baselines, simpler source summaries, and source-shuffled controls. Repeat under distribution shift to distinguish a stable readout from a dataset-specific correlate.

This evaluation answers *informativeness*: whether the interpretation maps the candidate source to its stated target. It does not by itself establish availability to B, grounding in the purported source, or downstream use.

5.3 Causal grounding

To test whether a readout is grounded in its purported source, intervene upstream on the claimed source s while holding alternative cues and the interpretation algorithm fixed. Let s' be a targeted, shuffled, stale, or counterfactual replacement, define $z(s) = \mathcal{P}(\mathcal{I}(s))$, and compare

$$\Delta_{\text{ground}} = D_z(z(s'), z(s)), \tag{14}$$

where D_z is chosen for the interface object. If the claim concerns a specific internal mechanism, intervene at that mechanism and measure its effect on the downstream readout. Specificity, dose response, and reversibility distinguish source grounding from generic disruption or a correlated alternative cue.

Natural grounding interventions include editing the subject-derived part of a summary, swapping retrieved episodes, perturbing output probabilities before interpretation, patching activations or Jacobians before J-space interpretation, and applying constrained parameter edits. Directly replacing a monitor output, J-space coordinate, or presented interface value does not test grounding; it tests downstream use.

This evaluation answers *grounding*: whether the readout depends on the purported subject-derived source rather than an alternative cue. Mechanistic attribution requires the stronger interventions appropriate to the mechanism claimed.

5.4 Monitoring-to-control transfer

Useful reflection should close a loop. Given interface value z and context c, a controller selects action $a = \mathcal{C}(z; c)$, such as answer, abstain, retrieve, call a tool, allocate compute, or request help. Evaluate

$$\Delta_{\text{use}} = D_G(G(\mathcal{C}(z'; c)), G(\mathcal{C}(z; c))), \tag{15}$$

where z' is a truthful, shuffled, stale, or counterfactual interface replacement, c is held fixed, G returns a defined action or behavioral statistic, and D_G is chosen for the range of G. Then evaluate outcome quality and cost:

$$U = \mathbb{E}_{i,d}[R(a_{i,d}, y_i) - \lambda \operatorname{cost}(a_{i,d})], \tag{16}$$

where the expectation runs over episodes and the reflection points within them, R captures task value, and cost captures inference, latency, or tool cost. Charging the cost inside the same expectation keeps value and expenditure on one population of reflection points, so a controller cannot look cheap by acting at fewer of them than it is scored on. For example, a local on-device model may delegate to a stronger remote oracle; the cost can include both money and response latency. This objective prevents trivial gains from always escalating.

For a concrete oracle-delegation task, let z be the local model's estimated probability that its answer is correct. The controller chooses $a \in \{\text{answer locally}, \text{delegate}\}$, and $G(a) = \mathbb{1}[a = \text{delegate}]$. Replacing z by a shuffled estimate z' makes Δ_{use} measure how much the interface changes the delegation decision. The reward $R(a, y)$ then scores the answer ultimately returned under that decision against ground truth y; the cost term charges for invoking the oracle. Thus Δ_{use} tests whether the controller responds to the interface, while U tests whether those responses improve task value enough to justify their cost.

A clean progression holds the task and controller fixed while changing the source or interface. Compare no signal, output probabilities, a fixed-purpose hidden-state readout, a richer activation representation, and a queryable mechanistic interface. The question is not whether the highest-dimensional source wins, but which pipeline yields calibrated utility under distribution shift.

This evaluation answers *use*: whether a recipient can interpret the interface and whether a controller changes action in a way that improves cost-sensitive outcomes.

5.5 Benchmark families

We suggest four benchmark families:

1. **Longitudinal contradiction.** Test whether retrieved records permit evidence-backed correction when current behavior conflicts with a persistent summary.
2. **Reasoning-state uncertainty.** Compare output-derived uncertainty with hidden-state correctness readouts before finalization, using selective accuracy and tool use as outcomes.
3. **State versus sensitivity.** Compare activation readouts with matched Jacobian-derived readouts on interventions whose effects depend on downstream sensitivity rather than activation magnitude.
4. **Mechanism-aware self-editing.** Ask a system to predict and verify the consequences of a constrained activation or parameter edit on held-out tasks.

All benchmarks should report subject boundary, source and provenance, interpretation algorithm, interface, recipient, controller if any, deployment status, temporal scope, observer controls, corrupted-channel controls, and compute cost.

6 Limitations and Risks

The source ladder describes practical classes of broader or deeper system access. That classification is useful for specifying what evidence an architecture can acquire; it should not be substituted for intelligence, maturity, causal relevance, or downstream utility. A calibrated fixed-purpose readout can outperform an open-ended interface that its recipient cannot interpret. More permissive access can also require greater bandwidth and stronger safeguards.

The reflective-subject boundary remains partly conventional. Different studies may choose one invocation, a persistent agent, or a larger model–harness system. The schema makes that choice visible but cannot choose it for every research question; within a comparison, however, B must remain fixed.

Internal readouts can fail under distribution shift, adversarial optimization, or representational drift. A model trained to consume a monitor may learn to influence it. Mechanistic access can improve diagnosis,

but it can also expose evaluation awareness or support monitor evasion [Gurnee et al., 2026]. Independent monitoring and causal validation remain necessary.

Human analogies can generate hypotheses, but they also import unsupported conclusions. This taxonomy concerns information interfaces and control loops. It neither establishes nor requires consciousness, subjective experience, or a human-like self-model.

7 Conclusion

In this paper, we have introduced a role-explicit framework for describing reflective interfaces in language-model systems. Its source-access ladder identifies how broadly or deeply an architecture reaches across context, harness records, model outputs, active state, or implementation state. The full schema declares the reflective subject and records which information is about and causally dependent on that subject, how it is interpreted, what interface is presented, which recipient receives it, whether a controller acts, and when the channel operates. Together, the ladder and schema make architectures comparable without collapsing access class, readout quality, deployment status, and downstream use into a single level.

They also make partial evidence legible. A probe may decode failure without making that readout available to the subject. A mechanistic tool may expose rich evidence without improving a decision. An endogenous control interface may influence later computation without representing a property of the subject. A controller may respond to a signal whose value is not grounded in the source claimed. Stronger evidence for self-reflection connects a declared subject, a subject-dependent and subject-directed channel, informative and grounded interpretation, and useful reporting or action under matched-observer, predictive, causal, and cost-sensitive evaluations.

Acknowledgments

This paper develops ideas first presented in the author's blog post, "Levels of self-reflection for LLMs" [Upadhyay, 2026].

References

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL `https://transformer-circuits.pub/2025/attribution-graphs/methods.html`.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023. doi: 10.48550/arXiv.2303.08112. URL `https://arxiv.org/abs/2303.08112`.

Jan Betley, Xuchan Bao, Martín Soto, Anna Sztyber-Betley, James Chua, and Owain Evans. Tell me about yourself: Llms are aware of their learned behaviors. In *International Conference on Learning Representations*, 2025.

Felix Jedidja Binder, James Chua, Tomek Korbak, Henry Sleight, John Hughes, Robert Long, Ethan Perez, Miles Turpin, and Owain Evans. Looking inward: Language models can learn about themselves by introspection. In *International Conference on Learning Representations*, 2025.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L. Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean,

Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *International Conference on Learning Representations*, 2023.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 8493–8502, 2022. doi: 10.18653/v1/2022.acl-long.581.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630:625–630, 2024. doi: 10.1038/s41586-024-07421-0.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 15466–15490. PMLR, 2024.

Amirhosein Ghasemabadi and Di Niu. Can LLMs predict their own failures? self-awareness via internal circuits. *arXiv preprint arXiv:2512.20578*, December 2025. doi: 10.48550/arXiv.2512.20578. URL https://arxiv.org/abs/2512.20578.

Wes Gurnee, Nicholas Sofroniew, Adam Pearce, Mateusz Piotrowski, Isaac Kauvar, Runjin Chen, Anna Soligo, Paul Bogdan, Euan Ong, Rowan Wang, Ben Thompson, David Abrahams, Subhash Kantamneni, Emmanuel Ameisen, Joshua Batson, and Jack Lindsey. Verbalizable representations form a global workspace in language models. *Transformer Circuits Thread*, jul 2026. doi: 10.48550/arXiv.2607.15495. URL https://transformer-circuits.pub/2026/workspace/index.html.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022. doi: 10.48550/arXiv.2207.05221. URL https://arxiv.org/abs/2207.05221.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. Semantic entropy probes: Robust and cheap hallucination detection in llms. *arXiv preprint arXiv:2406.15927*, 2024. doi: 10.48550/arXiv.2406.15927. URL https://arxiv.org/abs/2406.15927.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations*, 2023.

Ji-An Li, Huadong Xiong, Robert C. Wilson, Marcelo G. Mattar, and Marcus K. Benna. Language models are capable of metacognitive monitoring and control of their internal activations. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-2009.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, volume 36, pages 41451–41530, 2023. doi: 10.52202/075280-1797.

Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022.

Jack Lindsey. Emergent introspective awareness in large language models. *Transformer Circuits Thread*, oct 2025. doi: 10.48550/arXiv.2601.01828. URL https://transformer-circuits.pub/2025/introspection/index.html.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Gabrielle Kaili-May Liu, Areeb Gani, Jacqueline Lu, Jordan Thomas, Mark Steyvers, and Arman Cohan. Metacognition in llms: Foundations, progress, and opportunities. 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa 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*, volume 36, 2023. doi: 10.52202/075280-2019.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems*, volume 35, pages 17359–17372, 2022. doi: 10.52202/068431-1262.

Atharv Naphade, Samarth Bhargav, Sean Lim, and Mcnair Shah. Me, myself, and π: Evaluating and explaining llm introspection. In *ICLR 2026 Workshop on From Human Cognition to AI Reasoning: Models, Methods, and Applications*, 2026. doi: 10.48550/arXiv.2603.20276. URL https://arxiv.org/abs/2603.20276.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023. doi: 10.1145/3586183.3606763.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652, 2023. doi: 10.52202/075280-0377.

Siyuan Song, Harvey Lederman, Jennifer Hu, and Kyle Mahowald. Privileged self-access matters for introspection in ai. *arXiv preprint arXiv:2508.14802*, 2025. doi: 10.48550/arXiv.2508.14802. URL https://arxiv.org/abs/2508.14802.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L. Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023. doi: 10.48550/arXiv.2308.10248. URL https://arxiv.org/abs/2308.10248.

Utkarsh Upadhyay. Levels of self-reflection for llms. *Musically Yours*, April 2026. URL https://blog.musicallyut.xyz/2026/04/29/self-reflection.html. Original essay.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Wang. Soft thinking: Unlocking the reasoning potential of LLMs in continuous concept space. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5629. URL https://arxiv.org/abs/2505.15778.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023. doi: 10.48550/arXiv.2310.01405. URL https://arxiv.org/abs/2310.01405.