PRISM: An Observation Layer for Agentic and Long-Running Computational Systems

Park Junhu nller.park@snu.ac.kr

ABSTRACT

Long-running computational systems repeatedly expose nearly unchanged raw state. Language-model controllers make the resulting acquisition, representation, invocation, and reasoning costs visible. We introduce PRISM, a logically independent *Observation Layer* that projects process and log history into decision-facing semantic events before the expensive controller is invoked. The layer may run as an external daemon or as an independently stateful co-resident component; it requires neither model training nor weight changes.

Supporting theory states when this architectural boundary is safe and useful. It separates decision sufficiency from observation, representation, induced reasoning, and interaction resources. Under faithful decision-sufficient events, fixed polling period Δ, sparse semantic changes $K(T) = o(T)$, and explicit per-delivery inequalities, polling observations grow as $\Theta(T/\Delta)$ while event delivery grows with K. Shorter input alone does not imply shorter reasoning, and action equivalence cannot recover work already spent before a controller-side filter.

Our primary M4 Max study executes 720 episodes across eight live workloads, five paired repetitions, and nine architecture-matched controller checkpoints. All system outcomes succeed. Controller decision fidelity is 350/360 under polling and 360/360 under PRISM. Among the 350 pairs faithful in both arms, every checkpoint reduces aggregate tokens (51.9–84.5%), calls (28.0–54.0%), and delivered bytes (76.0–89.5%). An independent 360-pair replication spanning M3 Ultra live execution and M4 Max/Metal and RTX 5090/CUDA replays reproduces the aggregate direction in all nine model–execution blocks, establishing strong practical robustness over the tested scope.

A clean long-inference replay separates authoritative system outcome from controller self-report: ten fidelity-matched checkpoints reduce token totals, while Gemma 4 26B improves exact completion-identifier fidelity from 0/5 under polling to 5/5 under PRISM and is excluded from efficiency interpretation. Density, LLM-free, reasoning, and placement panels retain the boundaries: semantic delivery can increase notification count or bytes, shorter representations can induce longer generation, and co-residence preserves functional independence. PRISM therefore minimizes redundant decision-facing observation and controller activation conditionally, not information or reasoning universally.

1 Introduction

Long-running computational workflows—compilation, optimization, simulation, inference, downloads, tests, and CI/CD—must repeatedly determine whether an action-relevant state has changed. Language-model agents make the cost of this loop unusually explicit because they interleave environmental observation, reasoning, and tool-mediated action [10]. A conventional supervisor invokes `ps`, `tail`, or a status API on a fixed schedule. Most responses say only that the job is still running, yet each response is acquired, serialized, transferred, tokenized, inserted into context, and may trigger another inference.

PRISM—*Projection and Representation for Intelligent Semantic Monitoring*—treats this path as an architectural boundary. Its Observation Layer maintains state over raw history, applies a task-declared semantic projection, suppresses decision-equivalent repetition, and delivers a compact representation only when the declared semantic state changes. The expensive controller receives, for example, a verified `checkpoint_written` or `inference_complete(response_hash=...)` event rather than another raw log window. PRISM changes neither the workload, model weights, downstream tool contract, nor permission to act.

PRISM is not a better tool, scheduler, compiler, or workload runtime. It is an Observation Layer placed before the controller; downstream tool-contract and runtime improvements remain orthogonal and composable.

The central distinction is *where* the selection occurs. A model trained to ignore repeated raw polls may produce the same final action, but it cannot retroactively remove acquisition, input transfer, tokenization, context, or invocations already spent. Conversely, physical process separation is not necessary. An independently stateful observer attached at the model-call boundary can retain its own state and contract while sharing the controller's runtime, much as an organelle remains functionally distinct inside a host. We call both the external-daemon and co-resident forms Observation Layers; a filter evaluated only after the complete raw history has entered model context is not resource-equivalent.

This paper asks five research questions.

RQ1: Boundary and sufficiency. When is a logically independent Observation Layer information-theoretically and operationally meaningful? For raw history X, delivered history $Z = \Pi(X)$, and decision state Ω, the target is exact or approximate decision sufficiency together with a strict reduction in at least one pre-controller resource. Our required-event coverage, output fingerprints, identifier preservation, and independent system outcomes are operational probes; they do not estimate $I(\Omega; X \mid Z)$ or Bayes risk.

RQ2: Conditional resource effect. Conditional on system success and matched controller decision fidelity, how much do semantic delivery and compact representation change model invocations, delivered UTF-8 bytes, and observation-induced tokens relative to fixed 10-second polling?

RQ3: Practical robustness across controllers and platforms. Does the direction of the paired PRISM/polling effect recur under different controller checkpoints and execution architectures? Checkpoints and platforms are robustness blocks, not treatments or competitors. We never infer a scale law or hardware-speed rank-

ing from absolute cross-model token counts, latency, throughput, quality, or effect magnitude.

RQ4: Event density and reasoning. How does the effect change with elapsed time, density of decision-relevant changes, and the controller's reasoning response to a representation? This question separates information volume from inference behavior: fewer bytes or calls need not force fewer generated tokens.

RQ5: Placement. Can the same independently stateful observation function be attached co-resident at the model-call boundary without training or modifying the checkpoint? This asks about functional separation and feasibility, not whether an external daemon and an in-process component have identical IPC, scheduling, or failure-isolation costs.

We answer these questions with one architectural framework and its supporting analysis. The Observation Layer is the contribution; the formal resource model clarifies when its projection and delivery policy preserve decisions and when they reduce work. It orders observation, induced reasoning, and interaction resources without forcing arbitrary exchange rates, and connects the abstract boundary to bytes, tokens, context policy, and model computation. Its premises remain intentionally visible: semantic sufficiency, faithful delivery, event sparsity, and per-delivery inequalities are logically independent.

The primary empirical study executes on one 128-GB Apple M4 Max machine. Nine checkpoints form three matched robustness strata: Qwen3.5 9B, Llama 3.1 8B, and Ministral 3 8B are small dense controllers; Qwen3.6 27B, Gemma 4 31B, and Mistral Small 3.2 24B are medium dense controllers; Qwen3.6 35B-A3B, Gemma 4 26B-A4B, and GPT-OSS 20B-A3.6B are low-active-parameter MoE controllers. The matching prevents arbitrary capacity mixtures; it does not turn model size into an experimental treatment. Each checkpoint receives all eight workloads, both observation conditions, and five seeds.

All 720 primary system episodes succeed. Polling yields faithful terminal controller decisions in 350/360 pairs, while PRISM yields 360/360. We report resource effects only for the 350 pairs faithful in both arms. Within every checkpoint aggregate tokens, calls, and delivered bytes decrease; the observed ranges are 51.9–84.5%, 28.0–54.0%, and 76.0–89.5%, respectively. These are finite, within-checkpoint descriptive effects, not a pooled model benchmark or population claim.

The independent platform axis consists of the original live M3 Ultra matrix and canonical-trace replays on M4 Max/Metal and x86_64/RTX 5090/CUDA. Across three model sizes and the three execution settings, all nine model–execution blocks complete 80/80 system episodes and reproduce aggregate token, call, and byte reductions. Together with the nine-model primary matrix, this establishes strong practical robustness across the measured controller families, parameter regimes, dense/MoE architectures, workloads, repetitions, and Apple/Metal– NVIDIA/CUDA execution settings. The replay rows isolate persistence of the representation-to-reasoning effect; they are not live daemon-overhead or hardware-speed comparisons.

Long inference provides the sharpest protocol lesson. Earlier harness versions incorrectly conjoined authoritative workload state with a model's own `success` field, then attempted to satisfy that field by changing the PRISM payload. The final clean contract keeps the native semantic event, separates system success, controller decision fidelity, and optional success self-report, and excludes timing. Ten of eleven fidelity-matched checkpoints show token reductions; Gemma 4 26B instead improves identifier fidelity from 0/5 under polling to 5/5 under PRISM and is excluded from pure efficiency interpretation. Fresh live canaries reproduce all six selected replay conditions exactly after removing provider time fields.

Our primary contribution is the Observation Layer: a functional pre-controller boundary with external and co-resident realizations. Three supporting elements make that contribution testable. First, the conditional analysis states when the layer preserves decisions and reduces resources, with failure premises exposed rather than hidden. Second, PRISM-Bench separates authoritative system outcome, controller decision quality, observation representation, and induced reasoning. Third, a complete cross-family M4 Max matrix plus cross-architecture replication establishes strong practical robustness over realistic tested variation, while density, LLM-free, placement, and frozen-trace diagnostics expose explicit boundaries and counterexamples. The M3 Ultra/M4 Max/RTX 5090 replication remains separate from the primary estimand and is not treated as a hardware-speed comparison.

2 Background and Motivation

Agent interfaces materially affect behavior: ReAct couples reasoning and acting [10], while SWE-agent demonstrates that purpose-built agent–computer interfaces can improve software-engineering agents [9]. Existing agent benchmarks emphasize success across interactive environments [5, 8]. PRISM-Bench instead isolates a narrower systems question: what does it cost an already capable agent to learn that a long-running task changed state?

Most agent-stack optimizations begin after an observation already exists: reasoning consumes it, a tool call is formed and validated, and a runtime executes the accepted action. Input repair, schema validation, tool selection, scheduling, and execution optimization improve these later stages. PRISM asks an orthogonal question: *why was another observation needed at all?* Suppressing a semantically unchanged observation avoids downstream reasoning without changing how a later valid action is repaired, compiled, scheduled, or executed.

Polling couples observation frequency to wall-clock duration. Shorter polling intervals improve expected detection latency but create more repeated context; longer intervals save calls but delay reactions. Event delivery couples work to state changes, but introduces a different risk: a classifier can omit a decision-relevant event. Consequently, evaluation must jointly separate representation cost, misses, false wake-ups, authoritative system success, controller decision fidelity, and server overhead. Detection time may be a separate deployment endpoint; it is excluded from the present model-robustness comparison.

3 Observation-Layer Framework

The architectural question comes first: what work changes when an Observation Layer replaces repeated raw polling before the controller? This section gives the supporting analysis. Its coor-

dinates are not independent research programs; they distinguish what the layer must preserve, where it can save work, and where a cost may instead move downstream.

3.1 Supporting Resource Model

An episode starts a task, observes it through an Observation Layer, and reports its outcome. To evaluate that layer without hiding tradeoffs, we first model resources without choosing encodings, model providers, or prices. Let O, R, and I be partially ordered nonnegative resource spaces for observation, reasoning, and interaction.

Definition 1 (Agent Complexity)**.** For observer Π and horizon T, Agent Complexity is the product-space element

$$\mathcal{A}(\Pi;T) = \big(\mathcal{O}(\Pi;T), \mathcal{R}(\Pi;T), \mathcal{I}(\Pi;T)\big) \in \mathsf{O}\times\mathsf{R}\times\mathsf{I}. \quad (1)$$

The direct product is deliberate. Each factor is itself an ordered resource space, but its coordinates can have heterogeneous units and need not admit a canonical exchange rate. The product therefore carries the natural order

$$(o,r,i) \preceq (o',r',i') \iff o \preceq_O o',\ r \preceq_R r',\ i \preceq_I i'. \quad (2)$$

This order records Pareto improvement without pretending that a notification, a unit of model work, an environment action, and a unit of latency are commensurate. Thus one observer may use fewer deliveries but greater latency without either system being globally ordered. Any immediate scalar aggregate would silently impose deployment-specific substitution rates and could reverse under a different deployment. Product-space semantics retain the complete comparison until an explicit valuation is supplied; this is *delayed scalarization*.

These names describe supporting coordinates of the Observation Layer framework. Observation, Reason, and Interaction Complexity are the three peer factors of the resource product. Representation Complexity realizes the observation-representation coordinate; Token and Context Complexity are successive encoder and context-policy realizations that can induce Reason Complexity. Decision Complexity describes task demand and observer alignment, not another additive theory or resource:

$$\begin{aligned} \underbrace{\mathcal{A}}_{\text{resource product}} &= \mathcal{O}\times\mathcal{R}\times\mathcal{I}, \\ \mathcal{L}_\lambda \to \mathcal{T}_\tau \to \mathcal{C}_\chi &\to \mathcal{R}, \\ M(T) \quad &\text{indexes decision demand.} \end{aligned} \quad (3)$$

Interaction Complexity factors as

$$\mathcal{I} = (\mathcal{I}_{\text{task}}, \mathcal{I}_{\text{obs-induced}}), \quad (4)$$

separating actions required by the task from actions performed only to reduce state uncertainty. A deployment may enrich any factor with further resource coordinates. If it supplies a nonnegative dimension-matched linear functional w_A on the product space, and only then, the scalar deployment cost is

$$\begin{aligned} C_{w_A}(\Pi;T) &= \langle w_A, \mathcal{A}(\Pi;T)\rangle \\ &= C_{\text{obs}} + C_{\text{reason}} + C_{\text{task}} + C_{\text{obs-induced}}. \end{aligned} \quad (5)$$

The paper reports resource coordinates rather than selecting arbitrary prices.

Observer-service overhead is external to the agent-side product. Let $S_{\text{serve}}(\Pi;T) \in \mathsf{S}$ be its declared service-resource vector. It enters a total scalar comparison only through a separate nonnegative valuation w_s as $\langle w_s, S_{\text{serve}}\rangle$; it is never folded dimensionlessly into agent-side resources.

PRISM-Bench is one realization of these abstract spaces. Section 6 maps observation representation to encoded length, reasoning work to provider- reported model units, interaction to attributed tool records, and service resources to process measurements. Equal task-essential interaction is a paired-design premise checked by manifests and output fingerprints, not a consequence of Equation 1.

3.2 Resource Scaling at the Observation Layer

An Observation Layer is useful only if moving selection upstream changes a resource that the system actually pays. This section supplies the minimal resource account needed for that architectural claim: delivery count, representation volume, and detection delay. It then identifies the sparse-event condition under which semantic delivery can scale better than fixed polling.

3.2.1 Observation Cost Vector

Let Π be an observation mechanism, $T > 0$ a task horizon, and $Z^{\Pi}_{1:N_\Pi(T)}$ its delivered representations. Fix an abstract nonnegative representation functional $\lambda : \mathcal{Z} \to \mathbb{R}_+$ and a declared detection-delay functional d. Define

Definition 2 (Observation Complexity)**.**

$$\begin{aligned} \mathcal{O}_{\lambda,d}(\Pi;T) &= \big(N_\Pi(T), V_{\lambda,\Pi}(T), D_{d,\Pi}(T)\big), \\ V_{\lambda,\Pi} &= \sum_{i=1}^{N_\Pi(T)} \lambda(Z^{\Pi}_i). \end{aligned} \quad (6)$$

This is a mathematical resource vector, not a measurement prescription. λ may later be realized by an encoding length, energy cost, transfer cost, or another additive representation functional; d may be a mean, quantile, or deadline statistic. Comparisons require common choices of λ and d. Write $\mathcal{O}_A \preceq \mathcal{O}_P$ for componentwise non-increase, and $\mathcal{O}_A \prec \mathcal{O}_P$ when at least one component is additionally strict.

3.2.2 Comparing Coordinates Without Hiding Tradeoffs

For a nonnegative dimension-matched valuation w_O, define $C_{w_O}(\Pi) = \langle w_O, \mathcal{O}_{\lambda,d}(\Pi;T)\rangle$.

Product-order scalarization. If $\mathcal{O}_A \preceq \mathcal{O}_P$ and some strict component has positive weight, then $C_{w_O}(A) < C_{w_O}(P)$, because $C_{w_O}(P) - C_{w_O}(A)$ is a sum of nonnegative weighted coordinate differences with at least one positive term. *Depends on:* Equation 6 and the product order. If any component increases, this implication is inapplicable; a scalar comparison then requires disclosed weights.

3.2.3 Polling and Event Policies

A periodic policy with period $\Delta > 0$ delivers at $\Delta, 2\Delta, \ldots$, hence

$$N_{\text{poll}}(T) = \lceil T/\Delta \rceil. \quad (7)$$

If its positive finite mean representation value is v_p, then $V_{\lambda,\text{poll}} = v_p\lceil T/\Delta\rceil$. For fixed Δ, both coordinates are $\Theta(T/\Delta)$.

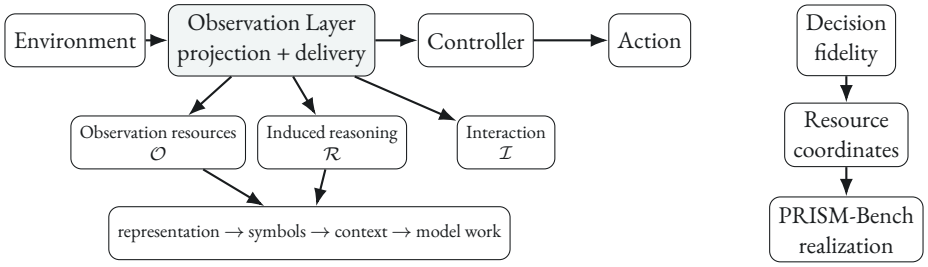


Figure 1. The Observation Layer is the architectural contribution. The resource coordinates, decision-fidelity condition, and PRISM-Bench measurements are supporting views used to evaluate it; they are not separate architectures.

Under an event time uniform between adjacent deliveries, $D_p \sim U[0, \Delta)$ and $\mathbb{E}D_p = \Delta/2$. These statements concern an abstract periodic policy; a benchmark later fixes an actual schedule and delay estimator.

Let $K(T)$ count delivered decision-relevant semantic changes. Under faithful one-event-per-change delivery,

$$N_{\text{event}} = K(T), \qquad V_{\lambda,\text{event}} = K(T)v_e, \tag{8}$$

for positive finite mean event value v_e. Thus both coordinates are $\Theta(K(T))$. Nothing implies $K < N_{\text{poll}}$.

3.2.4 Observation Sparsity and Finite Compression

Define

$$s_K(T) = K(T)/T, \qquad q_{A/P}(T) = K(T)/\lceil T/\Delta \rceil. \tag{9}$$

s_K is event density and $q_{A/P}$ is a dimensionless delivery-count ratio. Neither orders representation or reasoning resources. For fixed Δ, $K(T) = o(T)$ is equivalent to $s_K(T) \to 0$ and entails $q_{A/P}(T) \to 0$.

3.2.5 Online Notification Lower Bound

For a systems researcher, this bound answers whether an event observer could use still fewer messages without missing an action-relevant transition. The answer is conditional on deadlines and the absence of another information channel. Let $\tau_1 < \cdots < \tau_K$ be times at which the correct action changes. Assume each change must be exposed within ε, deadlines prevent batching, misses are forbidden, and no side channel exists.

Theorem 1 (Faithful online-notification lower bound)**.** *Every faithful observer uses $N_\Pi \geq K$ distinguishable notifications.*

Proof. If change i produces no distinguishing notification before its deadline, two trajectories can be identical to the agent through that deadline while requiring different actions afterward. The agent acts identically and fails on one. Each non-batchable change therefore requires a distinct notification. □

Depends on: causal observation, the declared deadlines, and the no-batching/no-side-channel premises. Prediction, batching, or alternate channels change the bound; it is not unconditional optimality.

3.3 Decision Sufficiency for Observation Layers

The resource argument is meaningful only if the Observation Layer preserves what the controller needs to decide. This section formalizes that safety condition. The central systems question is whether raw telemetry can be replaced at the controller boundary by a stateful semantic history without changing optimal task decisions.

3.3.1 Observation and Control

Let $E_t \in \mathcal{E}$ be environment state, $X_t \in \mathcal{X}$ raw telemetry, $Y_t \in \mathcal{Y}$ an observer output, and $A_t \in \mathcal{A}$ an action. A closed-loop agent is described by

$$X_t \sim O(\cdot \mid E_t), \qquad Y_t = \Pi_t(X_{0:t}), \tag{10}$$
$$A_t = \mu_t(Y_{0:t}), \qquad E_{t+1} \sim P(\cdot \mid E_t, A_t). \tag{11}$$

O acquires information, Π selects and represents it, μ reasons and chooses control, and P describes environment evolution. Observation does not change E_t; control does. Accordingly, Observation Complexity accounts for deliveries and their representations, while Interaction Complexity accounts for actions passed to P. PRISM changes Π, not μ or the task dynamics.

Definition 3 (Semantic observer)**.** A semantic observer is a causal observation mechanism logically upstream of the reasoning controller, $\Pi = (\pi, \sigma)$ comprising a semantic projection $\pi : \mathcal{X} \to \mathcal{Y}$ and a temporal selector $\sigma_t : \mathcal{Y}^{t+1} \to \mathcal{Y} \cup \{\perp\}$. It computes

$$Y_t = \pi(X_t), \qquad Z_t = \sigma_t(Y_{0:t}), \tag{12}$$

where $Z_t = \perp$ means no delivery. The reasoning policy consumes delivered Z_t, not raw X_t. Semantic observation is the upstream construction of Z_t; semantic reasoning is the controller's interpretation of Z_t.

This history-dependent definition contains the static notation $Y = \Pi(X)$ as the special case with no suppression. It also makes PRISM's two operations explicit: semantic projection followed by selection of novel decision-relevant state.

Observation 2 (Logical versus physical separation). The observer/controller boundary is defined by information flow, not process placement. An independently stateful observer may run as an external service or be attached co-resident at the model-call boundary. If it computes Π before the costly controller consumes input and suppresses $\perp$ deliveries, it remains an Observation Layer in either topology; no model-weight change is required. If full raw state X is first acquired, encoded, and supplied to the controller and Π is evaluated only afterward, the same action may be produced but upstream acquisition, representation, and invocation costs have already been incurred. Functional equivalence of actions therefore does not imply equality of observation resources.

3.3.2 When the Layer Preserves Decisions

The following result is the decision-quality gate for every later efficiency claim. It says when the layer may replace raw observation for an ideal controller; it does not assume that a deployed

projection satisfies the condition. For loss ℓ and latent decision state Ω, define

$$R^*(W) = \mathbb{E}\Big[\min_a \mathbb{E}[\ell(a, \Omega) \mid W]\Big]. \tag{13}$$

Call Π sufficient for the decision problem if

$$R^*(Z) = R^*(X), \tag{14}$$

where Z denotes the delivered semantic history available at the decision. Because Z is a (possibly history-dependent) function of X, coarsening cannot reduce Bayes risk; sufficiency requires equality, not merely a useful event vocabulary.

Theorem 3 (Decision-sufficient observation)**.** *If $Z = \Pi(X)$ and $\Omega \perp X \mid Z$, then $R^*(Z) = R^*(X)$ for every bounded decision loss, and $I(\Omega; Z) = I(\Omega; X)$.*

Proof. For every action, conditional independence makes posterior expected loss identical given X or Z. Their conditional minima coincide, so taking expectations proves the risk equality. Since Z is determined by X, the chain rule gives $I(\Omega; X) = I(\Omega; Z) + I(\Omega; X \mid Z) = I(\Omega; Z)$. □

Depends on: the Bayes-risk definition, the semantic observer map, exact conditional sufficiency, and the mutual-information chain rule. This theorem instantiates the exact-sufficiency results in Sections 3.4 and 3.9; it does not assert that a deployed observer satisfies the premise.

3.3.3 Conditional Notification Minimality

The online lower bound also characterizes when one notification per required decision change is minimal. Because this is a proof-level refinement of the delivery bound rather than a separate architectural contribution, its formal definition and corollary appear in Appendix B.1. The conclusion remains conditional: batching, prediction, or another information channel changes the bound.

3.3.4 Observer Quality and Cost

Observer quality and observer cost are separate objects. For a declared task, write

$$\mathcal{Q}(\Pi) = (R^*(Z), m_\Pi, f_\Pi), \qquad \mathcal{C}_{\text{obs}}(\Pi; T) = \mathcal{O}(\Pi; T), \tag{15}$$

where m_Π and f_Π are missed required changes and false notifications, and $\mathcal{O}$ is the Observation Complexity vector. Two observers may have equal $\mathcal{Q}$ but very different coordinates in O and R.

Equal-quality resource comparison. Let observers A and P satisfy equal decision risk and fidelity ($\mathcal{Q}(A) = \mathcal{Q}(P)$). If $\mathcal{O}(A; T) \preceq \mathcal{O}(P; T)$ and one component with positive deployment weight is strict, then $C_w(A) < C_w(P)$ without changing optimal decision quality: equal quality removes the decision-loss tradeoff, and the product-order scalarization in Section 3.2.2 applies to the nonnegative deployment weights.

3.3.5 Semantic Observer Substitution

The observer/controller analogy does not invoke the classical linear-control separation theorem. For a systems reader, the relevant question is narrower: if the Observation Layer is sufficient, can an optimal controller act exactly as it would on raw state? The following conditional substitution statement answers yes under common tie-breaking and coupled environment randomness. The formal corollary and proof appear in Appendix B.3. It is not a claim that arbitrary LLM policies are Bayes-optimal or that PRISM satisfies a classical control theorem.

3.4 Projection and Delivery Inside the Observation Layer

The preceding section states what the layer must preserve. This section states what PRISM actually does: it projects raw telemetry into a decision-facing state and suppresses delivery until that state changes. "Compression" here names these two architectural operations; it is not a claim that every encoded message, token sequence, or reasoning trace must be shorter. Let $X \in \mathcal{X}$ be complete raw telemetry and introduce the semantic projection

$$\pi : \mathcal{X} \to \mathcal{Y}, \qquad Y = \pi(X), \tag{16}$$

where Y is PRISM's semantic representation, Ω latent decision-relevant state, A an action, and $\ell(A, \Omega)$ decision loss. Typical Ω values are running, completed successfully, failed, or stalled. We target Ω, rather than A, because actions also depend on model, prompt, and policy.

Semantic compression and temporal compression are distinct. The former maps a raw state to a decision-oriented representation, $Y_t = \pi(X_t)$. A temporal selector then delivers Y_t only when the configured decision-relevant semantic state changes. Thus PRISM is a semantic projection followed by temporal event selection. The latter avoids retransmitting states when no relevant transition occurs. A system may have compact individual messages but send them redundantly, or send few large messages; PRISM is designed to address both axes.

The selection criterion is semantic, not syntactic. Relative to the last delivered semantic state Y_{t^-}, a delivery is novel only when the configured decision-relevant state changes; repeated or byte-dissimilar raw observations with the same semantic state are suppressed. Elapsed time alone does not create a notification. Consequently, the intended event count tracks novel decision-relevant information—operationalized by declared semantic transitions—rather than raw-byte repetition or wall-clock ticks.

Exact semantic sufficiency is the task-dependent condition

$$\Omega \perp X \mid Y. \tag{17}$$

Because $Y = \pi(X)$, it entails $P(\Omega \mid X) = P(\Omega \mid Y)$ almost surely. This decision-theoretic reading relates to comparison of statistical experiments [1, 4]. It is not assumed universally or estimated by the present panel.

3.5 Realizing Observation Cost in Bytes, Tokens, and Context

The Observation Layer produces a representation, but systems pay for concrete encodings and controller work. This section connects the architectural output to those measurable costs. Representation, Token, and Context Complexity are successive realizations of one Observation Layer output, not independent frameworks or interchangeable names:

$$\begin{aligned} \text{representation} \xrightarrow{\tau} \text{symbols} \xrightarrow{\chi} \text{context} \\ \xrightarrow{\Phi_{\mathcal{M}}} \text{model computation.} \end{aligned} \tag{18}$$

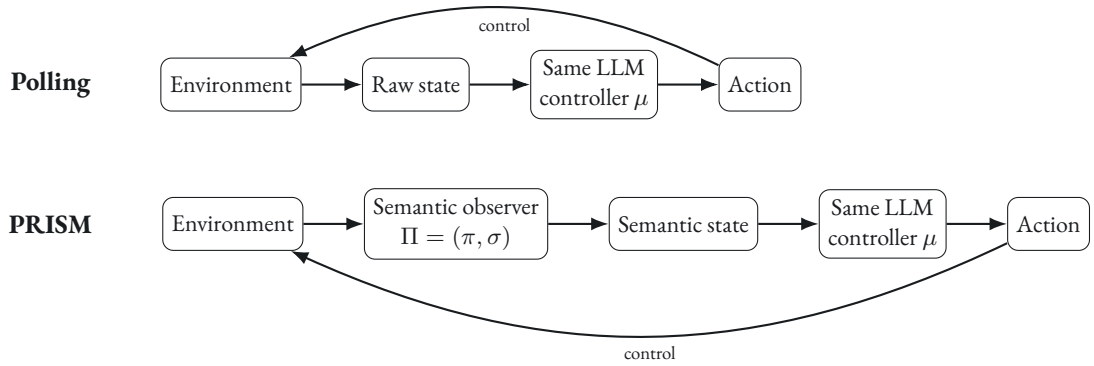


Figure 2. Closed-loop comparison. PRISM replaces the observation operator with a logically upstream semantic observer; the LLM controller and environment-control path are held fixed. The observer may be a separate service or an independently stateful component attached co-resident at the model-call boundary.

Here τ is an encoder/tokenizer, χ a context policy, and Φ a model-architecture cost functional. None is fixed by Observation Complexity.

3.5.1 Abstract Representation Coordinate

Definition 4 (Representation Complexity)**.** Let $\mathcal{Z}$ be a representation space and $\lambda : \mathcal{Z} \to \mathbb{R}_+$ a declared representation functional. For a delivered sequence $Z_{1:n}$ define

$$\mathcal{L}_\lambda(Z_{1:n}) = \sum_{i=1}^{n} \lambda(Z_i). \tag{19}$$

This is an abstract encoding/resource cost, not Shannon entropy, mutual information, or decision loss. A concrete byte encoding is one possible realization of λ, introduced only in Section 6.

Proposition 4 (Architecture invariance of exact encoded volume)**.** *Let $e : \mathcal{Z} \to \{0, \dots, 255\}^*$ be deterministic and define $\lambda_e(z) = |e(z)|$. If two executions deliver the same finite sequence $Z_{1:n}$, then*

$$\mathcal{L}_{\lambda_e}(Z_{1:n}) = \sum_{i=1}^{n} |e(Z_i)| \tag{20}$$

is identical under both executions, independently of processor, accelerator, operating system, model runtime, or model response.

Proof. Determinism fixes every encoded byte string $e(Z_i)$. The finite sum depends only on those strings and their delivery multiplicities, not on the machine that consumes them. □

Depends on: an identical delivered sequence and an identical deterministic encoder. If controllers stop after different prefixes, equality holds prefixwise; any aggregate difference is then a delivery-selection difference rather than an encoding-hardware difference. Length-prefixed hashes make sequence identity auditable but are not a premise of the proposition.

This proposition localizes a controlled cross-architecture replay. Exact byte volume belongs to the representation and selection stages. Token, context, invocation, and compute differences remain downstream empirical realizations; byte invariance alone does not order any of them.

3.5.2 Tokens as an Encoder Realization

Let $\tau : \mathcal{Z} \to \mathcal{V}^*$ be any deterministic symbol encoder and $n_\tau(z) = |\tau(z)|$. Define

$$\mathcal{T}_\tau(Z_{1:n}) = \sum_{i=1}^{n} n_\tau(Z_i). \tag{21}$$

Representation shortening alone need not order symbol counts. We expose the required regularity rather than assume proportionality [2].

The technical bridge from representation length to symbol count requires declared *Encoder-density bounds*; representation shortening alone is not enough. Appendix B.2 states the bounds and the *Representation-to-symbol implication* in full. Keeping that helper result outside the main flow makes the architectural point visible without weakening its condition.

Sparse-event token consequence. If per-delivery symbol counts have positive finite stable means, periodic delivery has $\mathcal{T}_\tau = \Theta(T/\Delta)$ and faithful one-event-per- change delivery has $\mathcal{T}_\tau = \Theta(K(T))$, obtained by multiplying the delivery orders in Section 3.2 by the corresponding stable means. *Depends on:* the polling/event delivery orders and stable encoder realizations. It asserts no finite-task dominance.

3.5.3 Context-Policy Realization

For observation-triggered call j, let p_j be processed input symbols, q_j newly retained symbols, and C_j maintained context before the call. For a declared context policy χ, define

$$\begin{aligned} \mathcal{P}_n &= \sum_{j=1}^{n} p_j, \qquad \mathcal{C}_n = \max_{j \le n} C_j, \\ C_{j+1} &\le C_j + q_j. \end{aligned} \tag{22}$$

$\mathcal{P}$ is cumulative processing and $\mathcal{C}$ peak Context Complexity. Under append-only history with positive finite stable increments, $\mathbb{E}C_n = \Theta(n)$. Hence

$$\mathbb{E}C_{\text{poll}} = \Theta(T/\Delta), \qquad \mathbb{E}C_{\text{event}} = \Theta(K(T)). \tag{23}$$

With stateless or uniformly bounded context, $\mathcal{P}_n = \Theta(n)$; with fully re-evaluated append-only history, $p_j = \Theta(j)$ and

$$\mathcal{P}_{\text{poll}} = \Theta((T/\Delta)^2), \qquad \mathcal{P}_{\text{event}} = \Theta(K(T)^2). \tag{24}$$

Other policies, including caching, truncation, and summarization, induce other orders. Context policy is therefore an explicit realization parameter.

3.5.4 Architecture-Neutral Compute Realization

Let $\Phi_{\mathcal{M}}(p, c)$ be any declared nonnegative computation functional for model architecture $\mathcal{M}$, input length p, and maintained state c. The abstract theory does not prescribe its form.

Monotone compute realization. If $\Phi_{\mathcal{M}}$ is coordinatewise nondecreasing and paired calls satisfy $(p_{A,j}, c_{A,j}) \preceq (p_{P,j}, c_{P,j})$, then summing over paired calls cannot increase realized compute; strict increase of $\Phi_{\mathcal{M}}$ on one strict pair yields a strict total inequality by applying monotonicity call by call and summing. *Depends on:* a declared architecture functional and paired calls. Unequal call counts require direct evaluation of the two sums.

The conditional cost chain is therefore

$$X \xrightarrow{\pi} Y \longrightarrow \mathcal{L}_\lambda \xrightarrow{\tau} \mathcal{T}_\tau \xrightarrow{\chi} (\mathcal{P}, \mathcal{C}) \longrightarrow \mathcal{R} \xrightarrow{\Phi_{\mathcal{M}}} C_{w_A}. \quad (25)$$

The first arrow preserves decision information only under sufficiency; each later arrow has its own declared realization premise. PRISM seeks lower representation cost subject to semantic fidelity, not lower information or Bayes risk.

3.6 Controller Work Induced by Observation

An Observation Layer can reduce deliveries yet still induce more controller work per delivery. This section isolates that downstream coordinate so that an observation saving is never silently equated with a reasoning saving. Reason Complexity is the supporting accounting term for this induced work. For a reasoning system $\mathcal{M}$, choose declared nonnegative accounting functionals for processed input P, produced output G, invocation count $N_{\mathcal{M}}$, and reasoning latency statistic L. Define

Definition 5 (Reason Complexity)**.**

$$\mathcal{R}(\Pi; T) = (P_\Pi, G_\Pi, N_{\mathcal{M},\Pi}, L_\Pi) \in \mathsf{R}. \quad (26)$$

These are abstract model-work coordinates. They need not be tokens, and the theory does not identify latency with neural computation. A realization must state its encoder, service boundary, caching policy, and latency definition.

Delivery-induced reason order. Suppose each delivery triggers exactly one reasoning invocation and the per-invocation coordinates in Equation 26 have positive finite stable means. Then periodic Reason Complexity components are $\Theta(T/\Delta)$, whereas faithful one-event-per-change components are $\Theta(K(T))$: $N_{\mathcal{M}} = N_{\mathrm{obs}}$ under the trigger premise, and multiplying the polling and event delivery orders by stable means preserves asymptotic order. *Depends on:* the observation-delivery orders in Section 3.2 and the stated trigger/stability premises. Batching, cached decisions, or multiple invocations per delivery require a different realization; finite dominance remains componentwise.

3.7 Decision Alignment at the Observation Boundary

The Observation Layer should deliver when the task's correct decision can change, not merely when bytes differ. This section supplies a supporting demand index for that alignment. Decision Complexity is therefore a property of the task/layer boundary, not another additive resource family. Let $M(T)$ be the number of predeclared decision changes with deadlines during a task. A decision change is a time at which the benchmark's correct action class changes; it is independent of observation frequency.

For delivery j, let Q_j contain the previously unreported decision changes that it first reveals before their deadlines. Define

Definition 6 (Decision-alignment complexity)**.**

$$U_\Pi = \sum_{j=1}^{N_\Pi} \mathbf{1}\{|Q_j| > 0\},$$
$$\eta_\Pi = U_\Pi / N_\Pi, \qquad r_\Pi = 1 - \eta_\Pi. \quad (27)$$

U counts informative deliveries, $\eta \in [0, 1]$ is Observation Efficiency, and $r \in [0, 1]$ is Observation Redundancy. This remains normalized when one delivery batches multiple changes. Under non-batching, deadline-separated faithful delivery, $U = M$, recovering $\eta = M/N_{\mathrm{obs}}$.

Sparse-decision alignment consequence. Fix $\Delta > 0$. Assume $M(T) = o(T)$, polling is faithful, deadlines are separated so one poll cannot first reveal two changes, and every change is revealed. Then $\eta_{\mathrm{poll}} \to 0$ and $r_{\mathrm{poll}} \to 1$. If PRISM emits exactly one informative delivery per decision, then $\eta_{\mathrm{PRISM}} = 1$ and $r_{\mathrm{PRISM}} = 0$ for every nonempty task. The polling premises give $U_{\mathrm{poll}} = M(T)$ and $N_{\mathrm{poll}} = \Theta(T)$, so $\eta_{\mathrm{poll}} \to 0$ and $r_{\mathrm{poll}} \to 1$. For PRISM, $U_{\mathrm{PRISM}} = N_{\mathrm{PRISM}} = M(T)$. *Depends on:* the polling delivery order, the informative- delivery definition in Equation 27, and one informative event per decision change. Thus, under the finite premises and $N_{\mathrm{poll}} > M > 0$, $r_{\mathrm{PRISM}} < r_{\mathrm{poll}}$. PRISM-Bench must preregister decision deadlines and record Q_j when experimentally evaluating this deadline-sensitive definition. The present cost study instead reports event-to-observation ratios and does not use deadline responsiveness as an outcome.

3.8 Temporal Redundancy in Observation Streams

The Observation Layer suppresses repeated decision-equivalent state. This section explains why that operation can matter even when individual snapshots are already compact: a slowly changing stream contains temporal redundancy that a stateless representation metric cannot see. Let $X_1, \ldots, X_n$ be the raw states delivered at observation times. For a stationary finite-alphabet process with $0 < H(X_1) < \infty$, define entropy rate and temporal information redundancy

$$h_X = \lim_{n\to\infty} \frac{1}{n} H(X_1^n), \qquad \rho_X = 1 - \frac{h_X}{H(X_1)}. \quad (28)$$

When the limit exists, $0 \le h_X \le H(X_1)$, so $\rho_X \in [0, 1]$. Equivalently, $H(X_n \mid X_1^{n-1})$ converges to h_X for stationary processes. High ρ_X means the past predicts much of the next snapshot; it does not mean the representation is decision-sufficient.

For a finite horizon and alphabet size $q \ge 2$, define capacity-normalized temporal redundancy

$$\rho_n^{\mathrm{cap}} = 1 - \frac{H(X_1^n)}{n \log q}. \quad (29)$$

This distinct normalization supports a sparse-transition finite-horizon bound.

For a systems reader, the next theorem connects a workload property—few state transitions—to an opportunity at the Observation Layer. It does not claim that every rare transition is safe to discard.

Theorem 5 (Sparse-transition temporal compressibility)**.** *Suppose X_1^n takes values in an alphabet of size q and every trajectory in its support has at most S_n transitions. If $S_n = o(n)$, then $H(X_1^n) = o(n)$ and $\rho_n^{\mathrm{cap}} \to 1$.*

Proof. The support has size at most

$$q \sum_{s=0}^{S_n} \binom{n-1}{s} (q-1)^s. \tag{30}$$

Entropy is at most the logarithm of support size. With $S_n/n \to 0$, the standard binomial entropy bound gives $\log \sum_{s \le S_n} \binom{n-1}{s} = o(n)$, while $S_n \log(q-1) = o(n)$. Hence $H(X_1^n) = o(n)$ and division by $n \log q$ proves the result. □

Depends on: finite-alphabet support counting, the declared sparse-transition support condition, and the standard binomial entropy bound. Thus periodic snapshots of a state with sparse transitions contain asymptotically vanishing new information per snapshot. If those transitions are exactly the decision changes and PRISM faithfully emits one event per change, its delivery count follows transitions rather than repeated states. This conclusion still requires semantic sufficiency; temporal compression alone may discard a rare but important change.

Delivered information as a diagnostic. For a finite observation mechanism Π, let $Z^{\Pi}_{1:N_\Pi(T)}$ denote its random sequence of delivered representations by horizon T. We define cumulative delivered observation information as

Definition 7 (Delivered Observation Information Volume)**.**

$$\mathcal{J}_{\mathrm{obs}}(\Pi; T) = H(Z^{\Pi}_{1:N_\Pi(T)}) = \sum_{i=1}^{N_\Pi(T)} H(Z_i^{\Pi} \mid Z_{<i}^{\Pi}). \tag{31}$$

The equality is the entropy chain rule. It measures novelty over the whole delivered process: a predictable repeated observation contributes little conditional entropy, while a state transition contributes an innovation term. It remains distinct from decision information $I(\Omega; Z^{\Pi}_{1:N})$. A silent observer can have zero $\mathcal{J}_{\mathrm{obs}}$ and still be useless, so information volume must always be reported with semantic fidelity.

No estimator is part of this definition. A realization must separately name its representation alphabet, sampling assumptions, estimator, and finite- sample interpretation. Section 6 gives the specific encoded-stream diagnostics used in this paper.

3.9 Information Preservation at the Observation Layer

The systems claim is not that PRISM delivers less "information" in every sense. It is that the Observation Layer should discard redundant representation while retaining what matters for the declared decision. This section supplies the information-theoretic support for that distinction. Recall Bayes risk from Equation 13. Theorem 3 establishes Bayes-risk equality under exact semantic sufficiency.

Since $Y = \pi(X)$, data processing gives $I(\Omega; Y) \le I(\Omega; X)$ [3]. Under exact sufficiency $I(\Omega; X \mid Y) = 0$, and because Y is determined by X, the chain rule yields

$$I(\Omega; X) = I(\Omega; Y) + I(\Omega; X \mid Y) = I(\Omega; Y). \tag{32}$$

This equality is Theorem 3 specialized to the memoryless projection. This preserves decision-relevant information, not necessarily entropy or encoded length. Section 3.5 formalizes that separation: $I(\Omega; Y) = I(\Omega; X)$ is a decision-information statement, whereas $\mathcal{L}_\lambda$, encoder-symbol count, context growth, and $\Phi_{\mathcal{M}}$ are representation and computation costs. Exact sufficiency does not order any of the latter.

For approximate sufficiency let $\delta = I(\Omega; X \mid Y)$, use natural logs, assume $0 \le \ell \le L_{\max}$, and define $\|p - q\|_{\mathrm{TV}} = \frac{1}{2}\|p - q\|_1$ and

$$\varepsilon_{\mathrm{TV}} = \mathbb{E}\|P(\Omega \mid X) - P(\Omega \mid Y)\|_{\mathrm{TV}}. \tag{33}$$

For a systems researcher, the following bound quantifies graceful degradation: if the layer loses only a small amount of decision-relevant information, the Bayes-risk penalty is bounded rather than left qualitative.

Theorem 6 (Approximate decision preservation)**.**

$$0 \le R^*(Y) - R^*(X) \le L_{\max} \varepsilon_{\mathrm{TV}} \le L_{\max} \sqrt{\delta/2}. \tag{34}$$

Proof. Coarsening cannot improve Bayes risk. Expectations of a $[0, L_{\max}]$ function under two posteriors differ by at most $L_{\max}$ times TV. Pinsker with natural-log KL [3], Jensen's inequality, and $\mathbb{E}\mathrm{KL}(P(\Omega \mid X) \| P(\Omega \mid Y)) = \delta$ prove the last inequality. □

Depends on: bounded nonnegative loss, the half-ℓ_1 total-variation convention, natural-log KL, Pinsker's inequality, and Jensen's inequality. The constant uses nonnegative bounded loss and half-ℓ_1 TV; a signed-loss or looser convention gives a factor 2. We do not estimate δ here.

PRISM can be interpreted, but is not trained, as a decision-constrained information bottleneck [7]:

$$\min_{\pi} \mathbb{E}[L(\pi(X))] \quad \text{s.t. } R^*(\pi(X)) - R^*(X) \le \epsilon, \tag{35}$$

$$\min_{\pi} I(X; Y) \quad \text{s.t. } I(\Omega; Y) \ge I(\Omega; X) - \delta. \tag{36}$$

PRISM-Bench measures representation cost; it does not numerically solve these objectives or estimate their mutual informations.

For finite episodes, let $\widehat{\Omega}_i$ be independently labeled ground truth and A_i the agent's terminal claim. The empirical policy loss

$$\widehat{R}_\pi = \frac{1}{n} \sum_{i=1}^{n} \ell(A_i, \widehat{\Omega}_i) \tag{37}$$

is measured through the task-specific controller decision-fidelity contract. Authoritative system success is recorded separately and never inferred from A_i. The empirical loss is not $R^*(X)$ or $R^*(Y)$ because neither the environment distribution nor the Bayes-optimal policy is known. Required-event coverage and identifier retention are likewise operational probes for discarded decision evidence, not estimators of $\delta = I(\Omega; X \mid Y)$.

3.10 Observation-Layer Guarantees and Boundaries

The preceding sections are supporting views of one architecture. They specify what the Observation Layer must preserve, how often it delivers, how its output is encoded, and what controller work those deliveries induce. This section now collects only the guarantees needed for the systems claim. Semantic sufficiency is a decision-quality premise, not a resource saving; resource dominance requires additional sparsity, stability, and equal-task premises. Keeping those premises explicit prevents a gain in one coordinate from hiding a failure in another.

3.10.1 Reading the Resource Coordinates

The product-order scalarization in Section 3.2.2 converts observation-vector dominance to every compatible observation price. The same direct-product monotonicity argument applies to the full $\mathcal{A} = \mathcal{O} \times \mathcal{R} \times \mathcal{I}$. Finite realized-symbol dominance instead requires the inequalities $Kp_e < N_{\text{poll}}p_p$ and $Kg_e < N_{\text{poll}}g_p$; semantic events do not make these inequalities automatic.

3.10.2 Asymptotic Dominance

Why should a systems researcher care about the asymptotic statement? It identifies the workload regime in which the architectural boundary can change scaling: elapsed time creates polls, whereas the Observation Layer follows decision-relevant change. The result is deliberately conditional.

Theorem 7 (Sparse-event asymptotic observation dominance)**.** *Fix $\Delta > 0$. If delivery is faithful, $K(T) = o(T)$, and corresponding per-delivery means are positive finite constants, then*

$$\frac{N_A}{N_P}, \frac{V_{\lambda,A}}{V_{\lambda,P}}, \frac{P_A}{P_P}, \frac{G_A}{G_P} \longrightarrow 0. \tag{38}$$

Proof. $N_P = \lceil T/\Delta \rceil = \Theta(T)$ whereas $N_A = K(T) = o(T)$, proving the count ratio. For the representation functional,

$$\frac{V_{\lambda,A}}{V_{\lambda,P}} = \frac{v_e}{v_p} \frac{K(T)}{\lceil T/\Delta \rceil} \to 0. \tag{39}$$

Replacing v_e/v_p with the finite constant ratios p_e/p_p and g_e/g_p proves the token statements. □

Depends on: the polling/event delivery orders, fixed Δ, sparse events, and stable per-delivery realization means. This is an asymptotic result for sparse-event long tasks, not a statement about every finite workload.

3.10.3 From Asymptotic Theory to Finite PRISM-Bench Episodes

The sparse-event theorem predicts a trend over increasing horizons; a finite episode cannot establish the limit $K(T)/T \to 0$. PRISM-Bench instead evaluates finite manifestations of the theorem by measuring K, N_{poll}, realized representation value, and induced model work, and by checking faithful delivery and equal task actions. Each workload therefore diagnoses whether the theorem's premises and component inequalities hold at its observed horizon. A favorable finite ratio is consistent with the predicted regime but is not proof of asymptotic behavior. Conversely, a finite count or token counterexample is expected when a premise fails and is informative about the boundary of the conditional claim rather than a contradiction of the theorem.

The next theorem is the paper's formal synthesis. It does not introduce a new theory; it states when the Observation Layer simultaneously preserves the declared decision and reduces the measured pre-controller resource chain.

Theorem 8 (Conditional Observation-Layer Dominance)**.** *Assume (i) $Y = \pi(X)$ and $\Omega \perp X \mid Y$; (ii) faithful delivery; (iii) $K(T) = o(T)$; (iv) fixed Δ; (v) positive finite stable event and poll representation/model-work means; (vi) identical task-essential interactions; (vii) one observation-triggered reasoning call per delivery with stable finite means; and (viii) separately measured daemon resources. Then*

$$R^*(Y) = R^*(X), \qquad I(\Omega; Y) = I(\Omega; X), \tag{40}$$

$$N_A/N_P \to 0, \qquad V_{\lambda,A}/V_{\lambda,P} \to 0, \tag{41}$$

and the prompt and generated token ratios also converge to zero.

Proof. Decision and information equalities are Theorem 3 and the chain rule argument in Section 3.9. The ratios follow from Theorem 7. Equal task interactions cancel in paired cost differences. □

Depends on: Theorem 3, Theorem 7, a stable reasoning-coordinate realization, and equal task-essential interaction.

3.10.4 Daemon Break-Even Condition

The Observation Layer may itself consume CPU, memory, and IPC resources. This final condition keeps the architecture honest by separating controller-side savings from total-system savings. Let $\Delta C_{\text{agent}}(T) = C_{\text{poll}}(T) - C_{\text{PRISM}}(T)$ and price the server vector by $C_{\text{serve}}(T; w_s) = \langle w_s, S_{\text{serve}}(T) \rangle$.

Corollary 9 (Priced total-system break-even)**.** *Under Theorem 8, total scalar system dominance holds only when the additional strict condition*

$$\Delta C_{\text{agent}}(T) > C_{\text{serve}}(T; w_s). \tag{42}$$

is satisfied.

Proof. Subtract the separately priced service cost from agent-side savings. □

Depends on: a declared agent valuation, a declared service valuation, and Theorem 8. Without deployment-specific w_s, priced total-system dominance is not established even when agent-side components fall. A concrete benchmark must measure S_{serve} separately and disclose any valuation.

3.10.5 Finite-Task Counterexamples

Table 1 makes the conditional structure concrete. Qwen long inference satisfies the observed count, byte, token, and fidelity premises. Gemma has lower raw resources but violates matched fidelity. Rust build has fewer calls and tokens while bytes increase. SGD reduces bytes while semantic count increases. A favorable coordinate neither proves the remaining coordinates nor compensates for decision loss.

For task j, let $\Delta_j = \mathcal{O}_{A,j} - \mathcal{O}_{P,j}$. Negative components in $\sum_j \Delta_j$ do not imply every $\Delta_j \preceq 0$; the measured rows provide explicit positive byte and count components.

The theory separates three claims: semantic sufficiency preserves decision quality; temporal alignment reduces redundant

Table 1. Measured premise diagnostics. P/A denotes polling/PRISM. Values are finite panel sums; the rows violate different implications and are not pooled.

Panel	Condition	Calls P/A	Bytes P/A	Tokens P/A	Decision P/A	Boundary
M4 clean replay	Qwen3.5 9B long inference	10/5	1,425/735	4,665/2,590	5/5	all measured premises hold
M4 clean replay	Gemma 4 26B long inference	10/5	1,425/735	20,015/3,082	0/5	fidelity mismatch; no efficiency claim
M4 primary	Rust build, 45 pairs	146/45	30,193/42,048	68,438/30,092	45/45	fewer calls but larger envelope
LLM-free	deterministic SGD, three pairs	15/24	2,010/1,509	–	3/3	dense events increase count

delivery; and compact event encodings can reduce representation and induced reasoning. None implies the other two without its stated premises.

4 System Design

4.1 Layered Agent Architecture

PRISM occupies one architectural boundary: the Observation Layer between the environment and reasoning. It decides when and in what representation state is delivered to the agent. The Reasoning Layer still chooses actions; the Tool Contract Layer still forms, validates, or repairs calls; the Tool Runtime Layer still dispatches them; and the operating system and execution engines still perform the work. PRISM changes none of those downstream semantics.

Layer independence is an interface claim, not a process-isolation requirement. The same independently stateful observer can be attached co-resident at the model-call boundary, performing projection and temporal selection before controller invocation without training or changing model weights. By contrast, invoking the full controller on every raw poll and asking it to ignore repetition places the filter after acquisition, serialization, tokenization, and wake-up; it may preserve decisions but is not resource-equivalent to pre-controller observation. The external `astop` service is the auditable reference realization, not a claim that every valid realization must be a separate daemon.

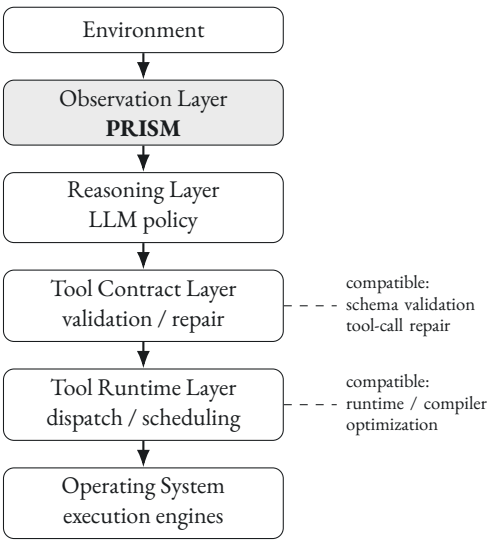


Figure 3. PRISM is an Observation Layer, implemented here by a Semantic Observer. Tool contracts, repair, runtimes, compilers, schedulers, and execution engines remain compatible downstream layers.

The layers compose rather than compete. One valid pipeline is Environment → PRISM → LLM → tool-contract repair → tool runtime → operating system. A better contract checker improves the calls that reach it; a better runtime improves their execution. PRISM reduces redundant observation-triggered passages through the whole downstream pipeline.

4.2 Observation Mechanisms

PRISM-Bench owns workload launch, PID lifetime, exit collection, and cleanup. A System Trans-Observation Projector (`astop`) is observation-only: registration never launches, kills, restarts, or modifies the target. A readiness barrier ensures astop observes the exact PID before the harness opens a start gate. Both conditions therefore execute the same command and trajectory.

The polling client receives only a raw state document containing process state, exit code when available, and a bounded log tail. It cannot call semantic wait or stream APIs. The PRISM client opens an astop event stream and remains silent between events. Canonical events retain audit fields internally; the agent projection removes envelope metadata while preserving decision fields. Ground-truth process status is stored separately and is never supplied to the PRISM agent.

A conceptual comparison follows the two information paths in Figure 2. Polling maps a wall-clock trigger to a raw snapshot and then to repeated LLM reasoning, even when the decision state is unchanged. PRISM maps raw telemetry through the semantic projection π, applies temporal event selection, and invokes LLM reasoning only on novel decision-relevant state changes. Raw byte repetition is not the selection rule, and elapsed time alone emits nothing. This pipeline is a conceptual illustration of the theory, not a measured performance figure.

5 Implementation

This implementation realizes only the Observation Layer in Figure 3. It neither constructs nor repairs tool calls, validates action schemas, schedules execution, optimizes compiled code, nor replaces the workload runtime. The harness owns workload launch, termination, and independent success ground truth; astop remains observation-only. astop retains full canonical events for audit, while its `profile=agent` wait and stream interfaces project each event to its decision-relevant fields before delivery to the LLM. PRISM-Bench does not filter those responses and measures their exact UTF-8 encoded size.

Jobs may install bounded regular-expression rules that map new log lines to named semantic events. astop incrementally maintains log cursors so old lines are not replayed. Built-in parsing emits canonical checkpoint events and the harness avoids registering a duplicate rule. On process disappearance, the engine drains the final available log batch *before* publishing the terminal event; otherwise subscribers permitted to close on exit could miss a same-tick completion record. Terminal events distinguish normal exit and exit code, signal termination, timeout, observation failure, and genuinely unknown state.

Task lifecycle and wrapper-process lifecycle use separate channels. A task-scoped semantic event may close the model-facing

subscription immediately; the harness does not invoke the model again merely to observe process exit. Independently, astop stores a durable exact-PID terminal notification. The harness blocks on `astop wait -until exit -json -ack`, checks the event against the child return code, and acknowledges it before accepting the episode. Campaign stages apply the same rule and additionally validate their artifact contracts; a successful child exit alone is not sufficient.

Every Ollama response stores `prompt_eval_count`, `eval_count`, their durations, and total duration verbatim. Thinking tokens, when produced, remain part of Ollama's `eval_count`. No character-based token estimate is used.

6 PRISM-Bench

PRISM-Bench is a concrete realization of the preceding framework, not part of its definitions. It evaluates Observation Layers while holding the controller- facing task fixed. Its workload families are ML training, Rust compilation, tests, benchmarks, model download, long inference, CI/CD replay, and scientific simulation. Deterministic fixtures support reproducibility; live profiles support measurement.

6.1 From Observation-Layer Model to Measurement

PRISM-Bench chooses the UTF-8 serialization $e : \mathcal{Z} \to \{0, \dots, 255\}^*$ and instantiates the abstract representation functional as

$$\lambda_B(z) = |e(z)|, \qquad B_\Pi = V_{\lambda_B,\Pi}. \tag{43}$$

Thus bytes measure one realization of the representation coordinate; they are not the abstract coordinate by definition. The benchmark chooses the Ollama tokenizer/service boundary for induced reasoning work, giving

$$\begin{aligned} N_\Pi &\leftrightarrow \text{observation-delivery records},\\ B_\Pi &\leftrightarrow \sum_i |e(Z_i)|,\\ P_\Pi &\leftrightarrow \sum \texttt{prompt_eval_count},\\ G_\Pi &\leftrightarrow \sum \texttt{eval_count},\\ N_{\mathcal{M},\Pi} &\leftrightarrow \text{Ollama request count},\\ L_\Pi &\leftrightarrow \sum \texttt{total_duration},\\ D_\Pi &\leftrightarrow \text{occurrence-to-delivery delay},\\ K &\leftrightarrow \text{required-event delivery count}. \end{aligned}$$

The latency coordinate is end-to-end model-service latency and may include loading; it is not identified with neural compute. Every request stores exact provider counts, including generated thinking when returned. No character- based token estimate is used.

The byte measurement is exact at the delivered-payload boundary. Every record stores the payload text and $|e(Z_i)|$; the normalized episode total is checked against their sum. Canonical-trace replay additionally hashes the complete source sequence after prefixing every UTF-8 string with its eight-byte length, and verifies that each delivered sequence is a source prefix. Consequently, cross-device byte equality for a common prefix instantiates Proposition 4; it is not inferred from token counts. HTTP framing, transport packets, and latent semantic information are outside B_Π.

With nonnegative deployment prices, the earlier scalar notation is instantiated as

$$C_{\text{obs}} = \alpha N_{\text{obs}} + \beta B_{\text{obs}} + \gamma T_{\text{obs}}, \tag{44}$$

$$C_{\text{reason}} = \eta P + \theta G + \kappa N_{\text{LLM}}, \tag{45}$$

but the paper reports coordinates and selects no numerical weights. Interaction Complexity is realized by task-essential, observation, and observation-induced tool-call attribution. Service resources are realized by daemon CPU seconds and peak RSS and remain separately priced.

6.2 Measurement, Derived Metric, and Estimator

PRISM-Bench keeps four levels distinct:

$$\begin{aligned} &\text{complexity object} \longrightarrow \text{measurement map}\\ &\qquad\qquad \longrightarrow \text{derived metric} \longrightarrow \text{estimator}. \end{aligned} \tag{46}$$

For example, exact payload bytes are measurements under λ_B; token counts are measurements under the fixed tokenizer/service; prompt totals and peak context are derived from ordered request records. The context realization is

$$\mathcal{P} = \sum_j p_j, \qquad \mathcal{C} = \max_j p_j. \tag{47}$$

For the measured dense Transformer deployment only, PRISM-Bench additionally reports

$$A_2 = \sum_j p_j^2, \tag{48}$$

an architecture-specific token-squared diagnostic motivated by dense-attention arithmetic. It is not an abstract complexity coordinate, hardware FLOPs, runtime, energy, or a theorem about future architectures. KV caching, kernels, batching, sparse attention, and hardware require different realizations of $\Phi_{\mathcal{M}}$.

For the encoded observation stream, PRISM-Bench frames each UTF-8 payload with a four-byte length and resets transitions at episode boundaries. From empirical byte and adjacent-pair frequencies it computes

$$\widehat{H}_0 = -\sum_b \widehat{p}(b) \log_2 \widehat{p}(b), \tag{49}$$

$$\widehat{h}_1 = -\sum_{b,c} \widehat{p}(b,c) \log_2 \widehat{p}(c \mid b), \tag{50}$$

$$\widehat{\mathcal{J}}_{\text{obs}} = m\widehat{h}_1, \qquad m = B_{\text{obs}} + 4N_{\text{obs}}. \tag{51}$$

This first-order plug-in quantity estimates an encoded-stream diagnostic, not the theoretical entropy rate, $H(Z_{1:N})$, latent-state entropy, or mutual information. Its ratio decomposes into framed-length and estimated entropy- density ratios. PRISM-Bench also reports covered required-event types per estimated kbit as an operational efficiency diagnostic.

The fixed-codec diagnostic

$$g_C = 1 - \frac{L_C(\text{frame}(x_1, \dots, x_n))}{\sum_i L_C(x_i)} \tag{52}$$

uses zlib level 9. It measures syntactic reuse under that codec and is neither entropy nor a property used by PRISM's semantic selector.

6.3 Operational Fidelity Is Not Sufficiency

For theoretical observer quality, PRISM-Bench uses empirical task loss $\widehat{R}_\pi$ as an operational probe, not as Bayes risk $R^*(Z)$. It maps m_Π to missed required changes and f_Π to false notifications. It checks required-event coverage, independent task success, false terminal claims, identifier retention, mode-invariant actions, and paired output fingerprints. These constitute *operational semantic fidelity*. They do not prove $\Omega \perp X \mid Y$, estimate $I(\Omega; X \mid Y)$, or identify Bayes risk. This distinction is maintained in every empirical interpretation.

The benchmark additionally separates outcome authority. System task success is computed from workload and observation evidence, never from a controller's own success claim. Controller decision fidelity is a second endpoint defined by the task contract (for example, a terminal decision plus exact completion-identifier preservation). A self-reported success field is retained as an optional diagnostic. Resource effects are labeled efficiency effects only when system success and controller fidelity hold in both paired arms; unequal-fidelity rows remain decision-quality results rather than being discarded or counted as cheap successes.

The finite theory diagnostics are $s_K = K/T$ and $q_{A/P} = K/\lceil T/\Delta \rceil$, computed from recorded timelines and the preregistered schedule. Deadline-sensitive decision alignment would require decision IDs, deadlines, and delivery attribution; responsiveness is not an outcome in the present cost study.

For every episode, raw artifacts include the manifest, workload log and timeline, observation/event JSONL, complete model requests, tool calls, independent process result, and normalized result. The download profile records cold-cache state, and CI/CD uses deterministic replay to avoid external drift.

7 Experimental Setup

7.1 Primary M4 Max Matrix

The primary matrix ran on a Mac Studio with a 16-core Apple M4 Max, 128 GB of unified memory, macOS 26.5.2, Ollama 0.32.1, and astop 0.4.1. The active Ethernet link was 1 Gb/s. Experiments were serial: unrelated Ollama models were unloaded between blocks, while workload-internal models were permitted only when declared by the task. Provider durations remain in raw audit records, but model latency, throughput, wall-clock time, and hardware speed are not outcomes.

The matrix crosses nine controller checkpoints, eight workloads, two observation conditions, and five seeds (1001–1005):

$$9 \times 8 \times 2 \times 5 = 720 \text{ episodes.} \tag{53}$$

The workloads are ML training, Rust build, Rust test, Rust benchmark, model download, long inference, deterministic CI/CD replay, and scientific simulation. All use live local task processes. The download task removes and re-pulls Qwen2.5 0.5B; long inference uses Qwen3.5 4B as the supervised workload, not as a controller condition.

Polling observes raw process state and a bounded log tail every 10 seconds. astop is registered for independent ground truth but its event stream is not exposed to the polling controller. The PRISM arm blocks on astop's semantic stream and receives only events selected by the same task-specific rule set. Each exact PID is registered before a launch gate releases the workload. Process termination is received through astop's durable terminal queue, matched to the child return code, and acknowledged. The system and task prompt, workload command, tool permissions, decoding configuration, and seed are held fixed within every polling/PRISM pair.

Ollama decoding uses temperature zero, top-$p = 1$, top-$k = 1$, no presence penalty, and a 32,768-token requested context. The request does not override the checkpoint's thinking mode; Ollama's `eval_count` therefore includes hidden reasoning when returned. No observer-model request has a fixed wall-clock client timeout: censoring slow generations would select favorable rows on a quantity outside the observation-layer estimand. Workload hard limits remain task contracts and are independent of controller inference time.

7.2 Why These Controller Checkpoints

The nine checkpoints form three capacity- and architecture-matched robustness strata:

The small stratum supplies contemporary sub-10B dense controllers from three families. The medium stratum is deliberately dense: GPT-OSS 20B is not placed there because it is MoE. The low-active stratum matches active rather than total parameters. Gemma 4 therefore appears twice as two different architecture-matched checkpoints. Qwen3.6 has no qualifying sub-10B release, and no large checkpoint is substituted into the small stratum.

The checkpoints are blocks. The valid estimand is the paired observation effect within a fixed checkpoint, task, and seed. We do not pool raw tokens, success rates, latency, or effect magnitudes across checkpoints; rank models; or interpret differences between strata as a parameter scaling law. A stratum summary may count only whether independently computed within-checkpoint effect directions recur. Qwen3.5 9B is reused from the exact M4 reproduction block and is not executed twice. Qwen3.5 27B and 35B remain same-family controls outside the nine-model primary estimand.

7.3 Outcome Separation and Efficiency Eligibility

The primary analysis separates authoritative system outcome from controller self-report. For the seven ordinary workloads, `system_task_success` requires workload exit zero and complete required-event coverage. For clean long inference it requires the preserved workload result plus decision- and evaluation-event coverage. A model's `success` claim cannot make the workload succeed or fail.

`controller_decision_fidelity` is a separate task-specific endpoint. For ordinary tasks it requires the declared terminal `done` and success decision. For clean long inference it requires `done=true` and exact preservation of the supplied SHA-256 completion identifier anywhere in the returned JSON; an optional success claim is recorded but non-gating. System success and controller fidelity are both reported. A paired token, call, or byte effect is eligible as an efficiency comparison only if both arms have system success and controller fidelity. Unequal-fidelity pairs remain accepted decision-quality outcomes but cannot become cheap "wins."

Legacy raw artifacts are immutable. The primary analysis projects their conjoined historical `task_success` field into the two outcomes above, retains the original value for audit, and replaces only the long-inference rows with the clean replay described

Table 2. Controller robustness strata. Parameter matching limits arbitrary capacity variation; rows are not performance tiers or a model-size treatment.

Stratum	Checkpoints	Match
Small dense	Qwen3.5 9B; Llama 3.1 8B; Ministral 3 8B	8–9.7B total, Q4 class
Medium dense	Qwen3.6 27B; Gemma 4 31B; Mistral Small 3.2 24B	24–31B total, Q4 class
Low-active MoE	Qwen3.6 35B-A3B; Gemma 4 26B-A4B; GPT-OSS 20B-A3.6B	3–4B active

next.

7.4 Clean Long-Inference Contract

Long inference revealed that a terminal task event and a wrapper-process exit are different lifecycle boundaries. The final `task-terminal-clean-v6` contract asks every controller only for `done` and the completion identifier. Polling receives two frozen raw observations: running, then exited with exit code zero and a response hash. PRISM receives the native two-field `inference_complete` astop event containing the same hash. No synthetic `completed` or `success` field is injected, and no thinking override or timing endpoint is introduced.

The corrected experiment replays model calls over immutable task traces from the aligned live panel rather than rerunning the long workload. Every replay is bound to its source manifest, normalized result, and length-prefixed payload SHA-256. It spans eleven checkpoints, two arms, and five seeds (110 episodes, 55 pairs), including the two same-family Qwen controls outside the primary matrix. Before admission, fresh live polling and PRISM conditions for Qwen3.5 9B, Llama 3.1 8B, and Ministral 3 8B must exactly match replay payloads, full model requests, provider-time-excluded responses, token counts, and decision metrics. All six conditions pass this equivalence gate.

Versions v2 through v5 are retained as protocol diagnostics, not silently overwritten. V3 changed both the prompt and semantic representation; v4 paired the legacy process-exit instruction with an earlier task-terminal event; v5 made the evidence compatible by injecting success fields but thereby changed the representation and expanded Qwen3.5 9B reasoning. Clean-v6 removes both confounds by separating outcome authority from model self-report.

7.5 Co-Resident Placement Panel

RQ5 uses Qwen3.5 9B, all eight workloads, five repetitions, and two pre-controller placements (80 episodes). The external arm consumes astop's event stream and projects its service envelope to the shared decision schema. The co-resident arm maintains an independently stateful observer inside the harness immediately before model invocation and does not register the workload with astop. Both use the same rule set, canonical decision schema, checkpoint, prompt, quantization, and seed; neither trains or changes model weights.

Each placement pair records system success, controller fidelity, required-event coverage, a length-prefixed hash of ordered canonical records, and a second hash of the sorted record multiset. The two arms are separate live workload executions. Exact information equality is therefore a diagnostic, not an identified placement effect: nondeterministic benchmark or progress facts may differ even when both observer implementations are faithful.

7.6 Secondary Boundary Panels

Three secondary panels remain logically separate from the 720-episode primary matrix. First, the Qwen reproduction panel includes controlled 30, 60, 120, and 300 second horizons with a fixed four-event semantic trajectory; it tests the finite direction predicted when polling count grows with elapsed time. Second, an 18-episode LLM-free panel observes deterministic SGD, Rust compilation, and numerical simulation. It records exact payload bytes, event coverage, and output fingerprints with $N_{\mathrm{LLM}} = 0$. Third, a five-pair, four-worker PyTorch panel fixes both observation arms to one terminal Qwen3.5 9B call and crosses native versus additional canonical compression with model-default versus disabled thinking. It isolates the fact that shorter representation can induce longer or incomplete generation.

The platform-generality panel, reported in Appendix A, combines the original live M3 Ultra matrix with canonical-trace replays on M4 Max/Metal and x86_64/RTX 5090/CUDA. It crosses three model sizes with three execution settings and 40 workload pairs per block: 360 paired comparisons and 720 system episodes. This panel tests whether the direction of the representation-to-reasoning effect survives the measured platform changes. It is analyzed under its own protocol, never pooled with the current M4 primary matrix, and never treated as a hardware-speed measurement.

7.7 Analysis

Every checkpoint has five paired repetitions per workload. We report exact ratio-of-sums reductions and deterministic paired-bootstrap 95% intervals within each checkpoint; the intervals summarize the finite paired block and are not population confidence claims. Cross-checkpoint strata count only effect direction. All raw Ollama requests and responses, observation traces, system results, model digests, manifests, and astop terminal-event acknowledgements remain in the replication directory. Provider time fields are retained for audit but excluded from every paper comparison.

8 Results

8.1 System Outcome Is Not Model Self-Report

The M4 primary matrix contains all preregistered 720 episodes. Every workload exited successfully and every required observation set was covered, giving 720/720 authoritative system successes. Controller behavior is a separate result. Polling produces 350/360 faithful terminal decisions, whereas PRISM produces 360/360. All ten unequal pairs are therefore accepted system outcomes and decision-quality contrasts, not harness failures.

The unequal cells are localized. In Rust benchmark, polling fidelity is 1/5 for Ministral 3 8B and 4/5 for Qwen3.6 35B, while PRISM is 5/5 in both. In long inference, Gemma 4 26B polling is 0/5 and PRISM is 5/5 because polling corrupts the terminal response hash. These ten pairs are excluded from pure resource-efficiency effects. The resulting primary efficiency set contains 350/360 pairs. This gate is symmetric: neither a cheap incorrect answer nor an expensive correct answer is selected as favorable

after inspecting token count.

This separation changes the interpretation of historical `task_success`. Five ordinary workload rows and five clean long-inference rows had successful systems but unfaithful polling-controller decisions. Conjoining the two would have called only 710/720 episodes successful and obscured the fact that all failures were downstream decisions.

8.2 RQ2–RQ3: Conditional Resource Effects Across Controllers

Table 3 reports only fidelity-matched pairs. Within every checkpoint, the ratio-of-sums for tokens, calls, and delivered UTF-8 bytes is below one. Token reductions range from 51.9% (Qwen3.6 27B) to 84.5% (Ministral 3 8B); call reductions range from 28.0% to 54.0%; and byte reductions range from 76.0% to 89.5%. All nine checkpoints therefore reproduce the direction of all three within-model aggregate effects.

The result demonstrates strong practical controller robustness, not a model ranking. Effect magnitudes are not monotone in total or active parameters. The medium dense row spans token reductions of 51.9%, 69.6%, and 81.4%, while the low-active MoE row spans 56.9%, 76.2%, and 74.4%. Architecture, tokenizer, training, and default reasoning policy change together across checkpoints, so no causal scale effect is identified. Deterministic paired-bootstrap intervals retained in the artifact summarize within-checkpoint variability; notably, Qwen3.6 35B's token interval is wide and crosses zero even though its 39 eligible-pair total falls 56.9%. We therefore treat the table as finite descriptive replication, not population inference.

Pairwise behavior is less uniform than model totals. Of 350 eligible pairs, PRISM uses fewer tokens in 347, fewer calls in 318, and fewer model-facing bytes in 305. Rust benchmark accounts for the three token increases and many call ties or increases; ML training also contains dense semantic trajectories. The aggregate direction cannot be promoted to universal componentwise dominance.

8.3 RQ3: Practical Robustness Across Platforms

The independent platform panel adds the original M3 Ultra live closed loop and canonical-trace replays on M4 Max/Metal and RTX 5090/CUDA (Table 8). It contains three model sizes, three execution settings, and 40 paired workloads per model–execution block, for 360 paired comparisons. Every block completes 80/80 system episodes. All nine reproduce aggregate token, call, and delivered-byte reductions: token reductions range from 58.8% to 84.4%, call reductions from 23.1% to 57.0%, and byte reductions from 88.1% to 94.1%.

The M3 rows establish the direction in a live Apple unified-memory execution; the M4 and RTX rows establish that the frozen representation-to-reasoning effect persists under Metal and x86_64/CUDA controller execution. Combined with the nine-model M4 primary matrix, the evidence establishes strong practical robustness across the tested workloads, model strata, and execution architectures. Because the replay rows hold source histories fixed, they do not estimate platform effects on workload behavior, daemon overhead, latency, or hardware speed, and they are not pooled with the primary estimand.

8.4 Clean Long-Inference Decision Contract

The clean-v6 panel contains 110 replay episodes and 55 paired comparisons. System success is 110/110. Controller fidelity is 50/55 for polling and 55/55 for PRISM. Among the ten checkpoints with 5/5 fidelity in both arms, every within-checkpoint token total decreases. Pooling only these eligible blocks for accounting, not model comparison, gives 51,039 polling tokens versus 28,050 PRISM tokens (45.0% fewer), 100 versus 50 model calls (50.0% fewer), and 14,250 versus 7,350 delivered bytes (48.4% fewer).

Gemma 4 26B is deliberately outside that efficiency aggregate. The polling controller returns `done=true` and valid JSON in every repetition but copies a 64-character SHA-256 completion identifier as a 60-character string. Its final baseline response deterministically omits four hexadecimal characters under greedy decoding. PRISM supplies the same identifier in one native semantic event and preserves it in 5/5 runs. The raw 20,015 versus 3,082 token difference remains auditable, but unequal decision fidelity makes "84.6% cheaper" an invalid efficiency claim. It is instead evidence that representation can affect decision quality.

The historical Qwen3.5 9B reversal is also resolved. A prior diagnostic injected `completed=true, success=true` into the PRISM payload to satisfy a model-gated success rule; this deterministically expanded hidden reasoning. Clean-v6 keeps native astop evidence and removes success self-report from system authority. Qwen3.5 9B then falls from 4,665 to 2,590 tokens (44.5%) while retaining 5/5 fidelity in both arms. Six fresh live conditions for Qwen3.5 9B, Llama 3.1 8B, and Ministral 3 8B exactly match their replay payloads, complete requests, provider-time-excluded responses, token counts, and decision metrics. The correction is therefore not an artifact of offline parsing.

8.5 Information Volume and Reasoning Are Independent Coordinates

The results reject either direction of a simple byte–reasoning equivalence. In clean long inference, every checkpoint receives the same aggregate observation-volume change, 1,425 to 735 bytes (48.4%), but eligible token reductions range from 33.7% to 53.2%. The mapping from a representation to generated reasoning is checkpoint-specific even when payload bytes and call counts are fixed.

The converse boundary appears in Rust build. In all 45 primary pairs, PRISM reduces observation calls to one while its native service-facing event is larger than the aggregate model-facing raw polls: polling transmits 625–813 bytes and PRISM 930–951 bytes depending on the checkpoint's observed prefix. All 45 token totals nevertheless decrease. This selector–envelope reversal shows that fewer deliveries do not imply fewer bytes, while the token result shows that more bytes do not imply more reasoning. Semantic selection, serialization envelope, and controller inference are three distinct optimization coordinates.

This distinction is precisely why PRISM-Bench reports calls, bytes, prompt tokens, completion tokens, and fidelity separately. No scalar "information efficiency" score is formed without deployment-specific weights, and encoded length is never presented as mutual information.

Table 3. M4 Max primary matrix. Models are robustness blocks, not competitors. Resource reductions are ASTOP relative to 10-second polling and use only pairs with system success and controller decision fidelity in both arms. All 720 system episodes succeeded. 'Eligible' reports fidelity-matched pairs; 'Decision P/A' reports polling/ASTOP controller fidelity out of 40.

Stratum	Controller	Eligible	Token red. (%)	Call red. (%)	Byte red. (%)	Decision P/A
Small dense	Qwen3.5 9B	40/40	81.8	54.0	89.5	40/40
Small dense	Llama 3.1 8B	40/40	75.5	41.9	78.4	40/40
Small dense	Ministral 3 8B	36/40	84.5	49.0	86.0	36/40
Medium dense	Qwen3.6 27B	40/40	51.9	28.0	76.0	40/40
Medium dense	Gemma 4 31B	40/40	69.6	40.6	80.8	40/40
Medium dense	Mistral Small 3.2 24B	40/40	81.4	44.8	83.3	40/40
Low-active MoE	Qwen3.6 35B	39/40	56.9	51.5	87.7	39/40
Low-active MoE	Gemma 4 26B	35/40	76.2	44.4	84.0	35/40
Low-active MoE	GPT-OSS 20B	40/40	74.4	48.3	85.8	40/40

Table 4. Clean-v6 long-inference replay. P/A values are polling/ASTOP sums over five frozen-trace repetitions. Token reduction is reported only with 5/5 decision fidelity in both arms. Gemma 4 26B's raw token difference is retained but not interpreted as efficiency because polling corrupted the completion identifier in all five runs. † Historical same-family controls lie outside the nine-model primary matrix.

Controller	Tokens P/A	Token red. (%)	Calls P/A	Bytes P/A	Decision P/A
Qwen3.5 9B	4,665/2,590	44.5	10/5	1,425/735	5/5
Llama 3.1 8B	2,635/1,280	51.4	10/5	1,425/735	5/5
Ministral 3 8B	2,845/1,850	35.0	10/5	1,425/735	5/5
Qwen3.5 27B†	5,435/2,545	53.2	10/5	1,425/735	5/5
Qwen3.6 27B	8,380/4,515	46.1	10/5	1,425/735	5/5
Gemma 4 31B	5,755/2,840	50.7	10/5	1,425/735	5/5
Mistral Small 3.2 24B	2,790/1,850	33.7	10/5	1,425/735	5/5
Qwen3.5 35B†	5,560/2,635	52.6	10/5	1,425/735	5/5
Qwen3.6 35B	8,335/5,485	34.2	10/5	1,425/735	5/5
Gemma 4 26B	20,015/3,082	–	10/5	1,425/735	0/5
GPT-OSS 20B	4,639/2,460	47.0	10/5	1,425/735	5/5

8.6 RQ4: Event Density and Reasoning Boundaries

The controlled-duration reproduction holds three semantic deliveries per episode fixed while increasing the task horizon from 30 to 300 seconds. Across five repetitions per checkpoint, PRISM remains at 15 calls and 860 bytes per horizon. Polling grows from 20 calls and 1,360 bytes at 30 seconds to 135–155 calls and 8,050–9,370 bytes at 300 seconds, depending on the checkpoint's valid stopping prefix. This four-point trend is consistent with sparse-event dominance; it does not establish an asymptotic limit.

The LLM-free panel supplies the complementary dense-event counterexample. All 18 systems and all nine paired output fingerprints succeed. Rust compilation and simulation reduce both deliveries and bytes, but deterministic SGD emits 24 faithful semantic notifications versus 15 polls while reducing bytes from 2,010 to 1,509. Architectural applicability does not imply count dominance: when semantic state changes more often than the polling grid samples it, a faithful event observer should deliver more.

The four-worker training panel isolates downstream reasoning. Both native arms make exactly one terminal model call and preserve all five paired final-model hashes. Native PRISM reduces delivered bytes by 67.7% and prompt tokens by 74.3%, but default completion tokens rise 4.53×, leaving total tokens nearly tied. Re-serializing the same native PRISM trace as chronological text retains 5/5 fidelity and the long-generation effect. More aggressive canonical compression produces smaller rows but omits required facts, and disabling thinking makes all canonical conditions fail. These rows locate a fidelity–compression boundary rather than a counterexample to the measured native representation reduction.

8.7 RQ5: Functional Independence Under Co-Residence

Both external astop and the independently stateful co-resident observer achieve 40/40 system successes, complete required-event coverage, and equal controller fidelity. No model weight is updated, and the co-resident implementation never registers its workload with the daemon. This demonstrates that physical daemon separation is not necessary for the Observation Layer abstraction.

Exact canonical information multisets match in only 23/40 live pairs. The mismatches are concentrated in ML training, Rust benchmark, simulation, and two Rust-test repetitions, where separate executions produce different progress, benchmark, or terminal facts. Consequently this panel establishes functional feasibility and matched operational fidelity, not byte-for-byte equivalence of placement under a controlled common trace. A frozen common-trace placement experiment would be required to identify implementation equivalence; native integration inside the Ollama server would be a further systems experiment.

The resource totals also cannot rank placements because the delivered histories differ in 17 pairs. External execution records 80 calls, 6,380 decision-facing bytes, and 32,595 tokens; co-residence records 85 calls, 6,665 bytes, and 36,873 tokens. These are descriptive audit totals, not a causal overhead or quality comparison. The valid conclusion is narrower: an observation-only, stateful component can be attached before model invocation without being learned into the checkpoint.

8.8 Answers to the Research Questions

RQ1. The layer is meaningful when its delivered history is an operationally faithful decision representation and it suppresses or shortens work before the expensive controller boundary. The experiments establish this for declared tasks, not universal information sufficiency.

RQ2. Under matched fidelity, all nine controller blocks reduce

Table 5. LLM-free Observation Layer panel on real deterministic workloads (three repetitions per condition).

Workload	Poll calls	PRISM calls	Call ratio	Poll bytes	PRISM bytes	Byte ratio	Success
ml sgd	15	24	1.600	2,010	1,509	0.751	yes
rust compile	60	3	0.050	11,463	240	0.021	yes
scientific simulation	12	9	0.750	4,881	831	0.170	yes

Condition	Complete report	Prompt tok.	Completion tok.	Total tok.
Native observer output (primary live comparison; text row is same-trace replay)				
Native polling JSON/default	5/5	2208.0 ± 0.0	460.0 ± 8.9	2668.0 ± 8.9
Native PRISM JSON/default	5/5	568.4 ± 99.3	2083.0 ± 750.7	2651.4 ± 708.3
Native PRISM text/default	5/5	329.0 ± 6.7	2133.2 ± 558.4	2462.2 ± 562.1
Additional canonical compression (boundary ablation)				
Canonical polling JSON/default	5/5	420.0 ± 0.0	2255.0 ± 0.0	2675.0 ± 0.0
Canonical polling text/default	5/5	404.0 ± 0.0	3198.0 ± 0.0	3602.0 ± 0.0
Canonical PRISM JSON/default	0/5	294.6 ± 5.4	390.4 ± 7.2	685.0 ± 11.7
Canonical PRISM text/default	3/5	263.2 ± 4.0	2147.8 ± 322.5	2411.0 ± 320.3
Canonical polling JSON/no-think	0/5	422.0 ± 0.0	117.0 ± 0.0	539.0 ± 0.0
Canonical polling text/no-think	0/5	406.0 ± 0.0	123.0 ± 0.0	529.0 ± 0.0
Canonical PRISM JSON/no-think	0/5	296.6 ± 5.4	109.0 ± 0.0	405.6 ± 5.4
Canonical PRISM text/no-think	0/5	265.2 ± 4.0	114.0 ± 0.0	379.2 ± 4.0

Table 6. Native-observer result and five-seed compression-boundary ablation. The primary comparison uses unmodified polling and PRISM outputs and one terminal LLM call. Canonical rows add a separate lossy-risk compression intervention; low-token rows that fail the complete-report gate are not cost wins. Time-derived outcomes are excluded.

Table 7. External-daemon versus independently stateful co-resident observer. Separate live executions make exact information equality a diagnostic rather than a controlled placement effect. Both placements succeeded in all 40 episodes.

Task	Pairs	Info eq.	Coverage eq.	Decision eq.
ML training	5	0/5	5/5	5/5
Rust build	5	5/5	5/5	5/5
Rust test	5	3/5	5/5	5/5
Rust benchmark	5	0/5	5/5	5/5
Model download	5	5/5	5/5	5/5
Long inference	5	5/5	5/5	5/5
CI/CD	5	5/5	5/5	5/5
Simulation	5	0/5	5/5	5/5
Total	40	23/40	40/40	40/40

their aggregate tokens, calls, and bytes; magnitudes remain task- and checkpoint-dependent.

RQ3. Effect direction recurs across three small dense, three medium dense, and three low-active MoE checkpoints, and across the M3 Ultra live, M4 Max/Metal replay, and RTX 5090/CUDA replay blocks. This establishes strong practical robustness across realistic tested variation in controller family, parameter regime, dense/MoE architecture, workload, repetition, and execution platform, but no model-size, model-quality, or hardware-speed ranking.

RQ4. Sparse long horizons favor semantic delivery; dense meaningful change can increase notifications; and shorter representations can induce longer reasoning. Information volume and inference efficiency are empirically and formally distinct.

RQ5. Co-resident pre-controller observation is feasible without weight changes and preserves operational fidelity in this panel. Exact placement equivalence and native model-runtime integration remain unproven.

9 Discussion

9.1 What Makes an Observation Layer Independent?

Independence here is a boundary of state, responsibility, and information flow. The observer owns a stateful map from raw history to a decision-facing representation and may suppress delivery altogether. The controller consumes that representation and chooses an action. Neither component must occupy a different machine or address space. An external astop daemon is valuable for auditability and failure isolation; a co-resident observer can avoid that transport boundary while remaining independently stateful and observation-only.

This definition excludes a superficially similar design: invoking the full controller on every raw poll and asking it to ignore repetition. Such a model may internalize the same semantic rule and produce an equivalent final action, but it operates after the raw observation has been acquired, serialized, tokenized, and inserted into context. It cannot recover those resources or the controller invocation. Learned semantic understanding and upstream observation selection can compose, but they are not resource-equivalent interventions.

The organelle analogy is therefore architectural rather than biological. A co-resident observer shares a host runtime while retaining its own state, contract, and lifecycle. The placement panel demonstrates feasibility at the harness/model-call boundary; it does not yet place the observer inside the Ollama server or establish that every deployment should prefer co-residence.

9.2 Observation Is Not Compression Alone

PRISM performs three separable operations: semantic projection determines which facts remain; temporal selection determines when a representation is delivered; and encoding determines its model-facing envelope. Treating all three as "compression" hides important counterexamples.

The Rust-build reversal is the clearest example. A single semantic notification can be larger than several terse polls because a durable event envelope contains command, path, timestamp, or notification metadata. The selector still removes wake-ups, but

the encoder has not reduced the byte coordinate. A production agent profile should therefore minimize decision-facing envelope fields while preserving a complete audit record out of band. Post-hoc projection is useful as a design diagnostic, but it cannot replace measured primary payloads.

Conversely, clean long inference fixes the byte reduction across checkpoints while token effects vary. Tokenization, context policy, hidden reasoning, and generation interact with semantics rather than byte length alone. The four-worker training panel goes further: dramatically shorter prompt input induces substantially longer completion reasoning. These observations support the paper's product-space model. Calls, bytes, prompt symbols, generated symbols, and decision loss are distinct coordinates; none is an adequate proxy for all the others.

9.3 Decision Quality Before Efficiency

The clean-v6 outcome contract is not only a harness repair; it is a general benchmarking principle. The environment or workload owns authoritative system state. The controller owns an interpretation and action. Conjoining a model's self-report with system success makes an evaluator unable to distinguish task failure, observation failure, and decision failure. Worse, changing the observation payload merely to elicit the required self-report changes the treatment being measured.

PRISM-Bench now reports three quantities separately: system task success, controller decision fidelity, and optional controller success claim. Resource effects require matched fidelity. This rule retains useful failures rather than discarding them. Gemma 4 26B's polling hash corruption is a decision-quality result; Ministral and Qwen benchmark misses are likewise controller results. None causes a successful workload to be mislabeled, and none can be counted as a token saving.

The strict hash criterion is intentional. A completion identifier is an integrity reference, not an approximate natural-language fact. A 60-character near-copy of a 64-character SHA-256 value cannot identify the same result. Relaxing the metric after observing Gemma's output would turn a precise decision contract into favorable post-hoc scoring.

9.4 Why Several Models and Why These Sizes?

The experiment does not ask whether a larger model is faster, cheaper, more accurate, or more compatible with PRISM. It asks whether a within-controller observation effect survives changes in the controller realization. Testing one checkpoint would confound an observation-layer result with one tokenizer, reasoning policy, and training recipe.

At the same time, arbitrarily mixing an 8B dense checkpoint with a 120B MoE checkpoint would make robustness difficult to interpret. The three strata therefore constrain controller envelopes. Small and medium rows match dense total parameters; the MoE row matches active parameters. GPT-OSS 20B belongs only in the latter. Gemma 4 supplies distinct dense and MoE realizations. The strata are blocking variables, not causal scale interventions, and effect magnitudes are not averaged across them.

The nine-model primary result and the three-platform replication test two independent axes of generality. The first shows that the aggregate direction is not unique to Qwen3.5, one tokenizer, or one architecture family. The second shows that it recurs in the original M3 Ultra live closed loop and in canonical-trace controller execution on M4 Max/Metal and RTX 5090/CUDA. This is strong practical robustness across realistic variation in controller family, parameter regime, dense/MoE architecture, workload, repetition, and execution platform, not merely a single-host observation.

The practical-robustness conclusion is the empirical headline. Its scope qualification is narrower: the study does not make a probability claim about an open-ended population, establish monotonicity or a model-size law, rank hardware speed, or predict an arbitrary unseen checkpoint. Indeed, the nonmonotone reductions are evidence against casual scale storytelling. The M4 and RTX replays specifically hold source histories fixed, so their contribution is portability of the representation-to-reasoning effect rather than a live comparison of daemon or workload cost.

9.5 What the Formal Guarantees Establish

Observation Complexity explains the duration result: with fixed Δ, polling count grows with elapsed time, while a faithful event observer tracks semantic change. The four measured horizons show the predicted direction when the event trajectory is held fixed. They cannot prove $K(T) = o(T)$.

The Observation Layer's decision-sufficiency analysis constrains what may be removed. Required-event coverage, independent outputs, final hashes, and exact completion identifiers are finite operational tests. They do not prove the conditional independence $\Omega \perp X \mid Z$ or estimate residual information $I(\Omega; X \mid Z)$. The empirical contribution is to expose these premises and their violations rather than claiming universal sufficiency from task success.

The LLM-free SGD panel locates one failure of count dominance: a faithful observer may emit more events than a coarse polling schedule. The training reasoning panel locates a different failure: shorter input may trigger longer generation. Rust build locates a representation-envelope failure, and Gemma 4 26B locates a fidelity mismatch. None contradicts the conditional theorems; each violates a different premise or component inequality.

9.6 Protocol Evolution as Evidence

The long-inference corrections illustrate why benchmark provenance matters. V3 changed prompt and payload together, so its Qwen reversal could not identify the cause. V4 preserved native payloads but gave the controller a process-exit instruction while closing the semantic stream on task completion. V5 aligned the evidence by injecting success fields, yet that injection changed the representation and hidden reasoning. Clean-v6 finally changes the outcome contract rather than the treatment: native evidence remains native, system authority is external, and controller fidelity is measured separately.

All versions remain immutable and auditable. The primary overlay is admitted only after source hashes, request options, payload order, call cardinality, and six fresh live equivalence conditions pass. This history should not be read as four independent confirmatory experiments. It is a transparent sequence of diagnostic failures leading to one clean estimand.

The automation follows the same principle. An exit-zero process is necessary but not sufficient for a completed campaign stage. astop's durable terminal event must be received and acknowledged, and the resulting artifact must pass its protocol, cardinality, success, fidelity, and canary contracts. Resume revalidates both

rather than trusting a stale state file.

9.7 Relationship to Downstream Agent Optimizations

PRISM is not a tool-call repair algorithm, scheduler, compiler, or workload runtime. Those components operate after an observation has reached the controller or after an action has been proposed. One valid stack is environment → Observation Layer → reasoning → tool-contract validation → runtime. Improvements at each layer compose. This paper distinguishes all of these boundaries but studies only the first as its intervention target.

This distinction also explains why PRISM is not restricted to LLMs. The reasoning factor may be a language model, a conventional program, or zero in the LLM-free panel. What remains is the cost and fidelity of transforming environment history into a decision-facing representation. LLM controllers make this cost conspicuous, but they do not define the architectural boundary.

10 Limitations

The primary result is a finite local systems experiment, not a population estimate. It uses one 128-GB M4 Max host, Ollama, quantized local checkpoints, eight declared workloads, and five repetitions. The independent M3 Ultra live and M4 Max/RTX 5090 replay panel materially broadens external validity: jointly, the studies establish strong practical robustness across realistic tested variation in controllers, workloads, and execution platforms. This is the main empirical conclusion, not a tentative single-host trend. The remaining limit is formal extrapolation: deterministic paired-bootstrap intervals do not license probability inference to an open-ended model, workload, or hardware population. The checkpoints and platforms are not random samples, a quality or speed benchmark, or a controlled parameter-scaling study.

The observer rules are task-specific and partly heuristic. Operational fidelity is checked through required events, process outcomes, output fingerprints, report facts, and completion identifiers. These checks neither prove exact semantic sufficiency nor estimate $I(\Omega; X \mid Z)$, Bayes risk, or the value of information omitted from the declared task contract. A new task requires a new projection and independent fidelity tests.

Resource effects are conditional on controller fidelity. Ten of 360 primary pairs are excluded from efficiency comparisons: five Gemma 4 26B long-inference pairs, four Ministral 3 8B Rust-benchmark pairs, and one Qwen3.6 35B Rust-benchmark pair. They remain reported decision-quality outcomes. This gate prevents incorrect cheap responses from being counted as wins, but it also means the per-model resource tables have 35–40 eligible pairs rather than a common 40 in every row.

Clean long inference is a validated model-only replay over frozen live traces, not a second execution of every long-running workload. Source manifests, normalized results, payload order, and request options are hashed and checked. Fresh live equivalence is established for six conditions across three small controllers, not all 22 controller–arm cells. The exact 6/6 match supports replay validity for the tested boundary; it cannot prove that every checkpoint would reproduce under arbitrary future provider or model revisions.

The placement panel runs the external and co-resident arms as separate live workloads. Both implementations achieve 40/40 system success, coverage, and decision-fidelity equality, but canonical information matches in only 23/40 pairs because several workloads produce execution-specific progress or benchmark facts. The panel therefore demonstrates functional co-resident feasibility, not exact implementation equivalence or comparative overhead. A common frozen-trace placement experiment and native integration inside the Ollama server remain future work.

Time-derived model outcomes are intentionally excluded. Provider duration, wall-clock, throughput, and latency would mostly measure checkpoint and runtime behavior rather than the observation mechanism, and fixed client timeouts would censor slow models. Raw durations remain available for diagnosing stalls. We also do not measure energy, memory traffic, KV-cache behavior, network packets, or hardware FLOPs. The token-squared dense-attention quantity is a declared proxy, not measured computation.

The byte coordinate ends at the model-facing UTF-8 payload. It excludes HTTP framing and network packets, while external astop transport bytes are stored separately. Rust build demonstrates that a native semantic envelope can be larger than several raw polls. A more compact production agent schema could improve that result, but substituting it after observing the data would change the measured treatment.

The secondary duration, LLM-free, training, M3 Ultra, and RTX 5090 panels use different estimands and protocols and are never pooled into the nine-model M4 primary result. The M3/M4/RTX panel nevertheless supplies the independent platform-generality evidence reported for RQ3; separation of estimands does not erase that replication. Earlier long-inference v2–v5 panels are protocol diagnostics, not repeated evidence for clean-v6.

Finally, a logically independent layer introduces engineering obligations: state ownership, schema versioning, backpressure, durable terminal delivery, crash recovery, security boundaries, and audit retention. astop implements one reference design, but the theorems do not guarantee that its service cost is smaller than agent-side savings. Total-system dominance requires deployment-specific weights over daemon CPU, memory, IPC, and failure-isolation costs.

11 Related Work

ReAct formalized interleaved language reasoning and environment actions [10]. PRISM does not replace that control loop; it changes the representation and timing of observations entering it. SWE-agent shows that agent-specific computer interfaces affect software-task performance [9]; PRISM focuses specifically on long-running observation and provides a component cost model.

Tool-input repair, schema validation, structured tool calling, scheduling, compiler optimization, and execution runtimes address downstream stages after an observation has entered agent reasoning and an action is being formed or executed. PRISM is orthogonal: it asks why another observation was delivered at all. This is an architectural distinction, not a claim that downstream work is unnecessary. A semantic Observation Layer can feed the same LLM, contract repair system, scheduler, and runtime, so improvements at those layers compose naturally with PRISM.

AgentBench evaluates agents across diverse interactive environments [5], and OSWorld supplies execution-based evaluation

in real computer environments [8]. PRISM-Bench is complementary: it holds the task trajectory fixed and attributes observation calls, bytes, tokens, misses, decision fidelity, and observer overhead. Classical event-driven monitoring motivates blocking notification, but our object of study is the decision-facing representation and reasoning induced by observation. The present model-robustness study excludes model latency and hardware speed.

The semantic-compression analysis uses classical entropy and coding concepts originating with Shannon [6] and the standard data-processing, conditional-mutual-information, and Pinsker results summarized by Cover and Thomas [3]. These results justify the information-preservation and approximate-risk bounds; they do not imply that PRISM's encoding is shorter. PRISM-Bench evaluates that separate empirical property through delivered bytes and Ollama token counts. Blackwell's comparison of experiments [1] and Le Cam's statistical decision framework [4] motivate our separation of observation representation from attainable decision risk. The information bottleneck method [7] motivates the relevance–compression tradeoff. PRISM is a hand-engineered instance of this view; it does not solve the information-bottleneck optimization numerically.

Classical systems and control commonly distinguish an observer that estimates state from a controller that acts on it. Our use is an architectural analogy: PRISM externalizes semantic observation while leaving the LLM policy fixed. We do not invoke the linear observer–controller separation theorem, assume linear dynamics, or claim equivalence to a classical state estimator. The sufficient-observer substitution corollary is instead a decision-theoretic result under explicit posterior-sufficiency and Bayes-optimality assumptions.

12 Conclusion

PRISM makes observation an explicit systems layer between environment history and expensive reasoning. Its Semantic Observer performs stateful projection, temporal selection, and representation construction before the controller is invoked. The layer may be an external daemon or a co-resident component with its own state and contract; it need not be trained into model weights. A model that filters after consuming every raw poll may preserve actions, but it cannot recover the pre-controller resources already spent.

The theory states the boundary condition precisely. Under faithful decision-sufficient events, sparse semantic change, fixed polling period, and explicit per-delivery inequalities, polling cost grows with elapsed time while event cost grows with decision-relevant change. Semantic sufficiency preserves decision quality; it does not imply shorter encoding or reasoning. Conversely, shorter encoding does not prove sufficiency. The supporting resource model keeps observation calls, representation, prompt, generation, interaction, and service resources as separate coordinates until a deployment supplies valid weights.

The primary M4 Max matrix completes 720/720 system episodes across nine controllers, eight workloads, two observation mechanisms, and five seeds. PRISM yields 360/360 faithful controller decisions versus 350/360 for polling. Among the 350 pairs faithful in both arms, every checkpoint reduces aggregate tokens, calls, and delivered bytes; the observed ranges are 51.9–84.5%, 28.0–54.0%, and 76.0–89.5%. Because effects are computed only within a fixed checkpoint, these results establish cross-controller robustness of direction, not a model ranking or scale law.

The independent platform replication adds 360 paired comparisons over the original M3 Ultra live closed loop and canonical-trace controller execution on M4 Max/Metal and RTX 5090/CUDA. All nine model–execution blocks complete 80/80 system episodes and reduce aggregate tokens, calls, and bytes. Taken together, the two studies establish strong practical robustness across realistic tested variation in controller family, parameter regime, dense/MoE architecture, workload, repetition, and Apple/Metal–NVIDIA/CUDA execution platform. This is the empirical headline rather than a tentative single-host trend. The studies do not rank hardware speed: the replay rows deliberately freeze source histories and exclude time.

Clean long inference shows why the outcome contract matters. Authoritative system success, controller decision fidelity, and optional model self-report must remain separate. Ten fidelity-matched checkpoints all reduce tokens, while Gemma 4 26B improves completion-identifier fidelity from 0/5 to 5/5 and is excluded from pure efficiency interpretation. Fresh live canaries match the frozen replay exactly in all six tested conditions. Earlier contradictory panels remain visible as protocol diagnostics rather than being averaged into the final result.

The counterexamples are as important as the reductions. Dense SGD state change produces more semantic notifications than polling; shorter native training traces induce longer model reasoning; Rust-build semantic envelopes are larger despite fewer calls; and separate co-resident executions do not always deliver byte-identical histories. These outcomes show that event selection, representation volume, reasoning efficiency, decision quality, and placement are related but independent design variables.

PRISM therefore does not claim to minimize information, reasoning, or total system cost universally. It seeks to preserve declared decision evidence while removing redundant pre-controller representation and activation. PRISM-Bench makes that conditional claim falsifiable by measuring system outcome, controller fidelity, calls, bytes, and tokens separately. It is a benchmark for Observation Layers, not a better tool, scheduler, compiler, runtime, or learned controller; those improvements remain downstream and composable.

A Secondary Replications and Protocol Provenance

A.1 Cross-Architecture Generality Replication

The original study used a live M3 Ultra Qwen3.5 9B/27B/35B matrix. Two canonical-trace replays subsequently generated new controller responses on an M4 Max/Metal system and an x86_64/RTX 5090/CUDA system while preserving the source observation sequences and model digests. Table 8 reports those within-execution effects. The panel contains 360 paired comparisons and 720 successful system episodes across its nine model–execution blocks.

These rows supply the platform axis of RQ3. They show that the aggregate representation-to-reasoning effect persists across the measured Apple unified-memory and x86_64/CUDA execution architectures. Together with the nine-model M4 matrix in Table 3, they establish strong practical robustness across the tested

Table 8. Cross-architecture observation effects. Reductions are PRISM relative to polling over 40 workload pairs per model. M3 Ultra is the original live closed loop; M4 Max and RTX 5090 replay the exact preserved source traces with new model responses. Time and latency are excluded.

Model	Execution	Token red. (%)	Call red. (%)	Byte red. (%)	Success
9B	M3 Ultra live	69.2	30.1	90.0	80/80
9B	M4 Max replay	84.4	57.0	94.1	80/80
9B	RTX 5090 replay	84.1	57.0	94.1	80/80
27B	M3 Ultra live	58.8	23.1	88.1	80/80
27B	M4 Max replay	71.4	43.8	91.4	80/80
27B	RTX 5090 replay	69.4	45.6	91.7	80/80
35B	M3 Ultra live	67.8	28.6	90.0	80/80
35B	M4 Max replay	79.1	47.8	92.7	80/80
35B	RTX 5090 replay	78.6	49.5	93.0	80/80

controllers and platforms.

The evidence remains stratified by estimand. M3 Ultra is a live closed loop; M4 Max and RTX 5090 replay fixed source histories with new model responses. The replay rows therefore do not rerun live workload or daemon behavior, compare hardware speed, or justify numerical pooling across machines. Those limits qualify the mechanism attributed to each block, not the replicated direction of effect.

A.2 Long-Inference Protocol Ledger

The following versions are retained as immutable diagnostics:

No version is selected because it produces a favorable token result. Each was invalidated by a contract mismatch observable before or independently of the headline effect. Clean-v6 admits all system-success rows, reports unequal decision fidelity, and applies the same fidelity eligibility rule to both arms.

A.3 Automation and Artifact Contracts

The M4 campaign is staged under astop supervision. For a stage to receive `validated_success`, the supervisor must receive and acknowledge the durable `process_exited` event, its exit code must match the child, and the stage-specific artifact validator must pass. Validators check protocol and policy identifiers, exact cardinality, source hashes, system-success coverage, controller fidelity, and live-canary equivalence where applicable. Resume repeats artifact validation before skipping a stage.

The clean replay contains 110 normalized episodes and 165 raw Ollama requests: two requests for each polling episode and one for each PRISM episode. None overrides model thinking. Source traces, complete request bodies, responses, provider token metadata, and output decisions remain available under `experiment/replications/m4_max_live_20260729`. Provider time fields remain audit metadata and are excluded from analysis.

B Supporting Observation-Layer Proof Details

This appendix retains proof-level refinements that support the Observation Layer guarantees but are not needed to follow the main architectural argument.

B.1 Conditional Notification Minimality

Fix decision changes $\tau_1 < \cdots < \tau_M$ and latency budget ε; here M is the lower-bound section's K specialized to genuine decision changes. Let $\mathfrak{F}_\varepsilon$ be the class of causal observers that reveal every required change before its deadline, cannot batch deadline-separated changes, and have no alternate side channel.

Definition 8 (Observer minimality)**.** A faithful observer $\Pi^* \in \mathfrak{F}_\varepsilon$ is notification-minimal if $N_{\Pi^*} \le N_\Pi$ for every $\Pi \in \mathfrak{F}_\varepsilon$.

Corollary 10 (Conditional minimal semantic observer)**.** *Under the faithful online-notification assumptions of Theorem 1, every $\Pi \in \mathfrak{F}_\varepsilon$ satisfies $N_\Pi \ge M$. A semantic observer that emits exactly one distinguishing notification per decision change has $N = M$ and is notification-minimal in $\mathfrak{F}_\varepsilon$.*

Proof. The lower bound follows by the same indistinguishability construction: without a distinguishing notification, two histories visible to the controller are identical but require different post-change actions. One notification per change attains the bound. □

Depends on: observer minimality and Theorem 1. No second lower-bound argument is needed. The conclusion changes if deadlines permit batching, a predictive controller acts before observation, or another channel reveals the change. PRISM is minimal only when its configured events coincide with these required decision changes and the lower-bound assumptions hold.

B.2 Representation-to-Symbol Helper

The following technical condition makes explicit when a representation-length comparison is strong enough to order encoded symbol counts.

Definition 9 (Encoder-density bounds)**.** On $\mathcal{S} \subseteq \mathcal{Z}$, (λ, τ) has constants $0 < a \le b < \infty$ and $c \ge 0$ if

$$a\lambda(z) - c \le n_\tau(z) \le b\lambda(z) + c \quad \text{for every } z \in \mathcal{S}. \tag{54}$$

Representation-to-symbol implication. Let $X_{1:n}, Y_{1:m} \subseteq \mathcal{S}$ use the same encoder. If

$$b\,\mathbb{E}\mathcal{L}_\lambda(Y_{1:m}) + cm < a\,\mathbb{E}\mathcal{L}_\lambda(X_{1:n}) - cn, \tag{55}$$

then $\mathbb{E}\mathcal{T}_\tau(Y_{1:m}) < \mathbb{E}\mathcal{T}_\tau(X_{1:n})$. This follows by summing the upper encoder bound for Y, applying Equation 55, and then summing the lower bound for X. *Depends on:* Representation Complexity and the declared encoder-density bounds. The condition is sufficient, not necessary.

Table 9. Long-inference protocol evolution. Only clean-v6 contributes primary efficiency evidence.

Version	Intended correction	Why it is diagnostic or primary
v2	Original adaptive harness	Conflated task and process termination and allowed duplicate reasoning.
v3	Close reasoning on logical task completion	Changed prompt and semantic payload together; cannot identify the cause of the Qwen reversal.
v4	Preserve legacy prompt and native event	Prompt required process-exit evidence while the model-facing stream closed on `inference_complete`.
v5	Align projected evidence with the legacy prompt	Injected `completed` and `success`; representation change expanded Qwen3.5 9B hidden reasoning.
clean-v6	Separate system outcome, decision fidelity, and self-report	Preserves native evidence, removes model self-report from system authority, excludes timing, and passes 6/6 live–replay equivalence. Primary.

B.3 Sufficient-Observer Substitution

Corollary 11 (Sufficient-observer substitution)**.** *Assume $Z = \Pi(X)$, $\Omega \perp X \mid Z$, a common measurable tie-breaking rule, and a Bayes-optimal controller. Then there exists an optimal semantic controller μ_Z^* whose action equals the raw-observation optimum μ_X^* almost surely. Under coupled environment randomness, the resulting control-action and state-transition laws are identical.*

Proof. The posterior expected loss of each action is the same given X and Z. Common tie-breaking therefore selects the same minimizer. Identical actions passed to the same kernel P induce identical coupled transitions. □

Depends on: Theorem 3, a Bayes-optimal controller, common tie-breaking, and the common transition kernel in Equation 11.

References

[1] David Blackwell. Equivalent comparisons of experiments. *The Annals of Mathematical Statistics*, pages 265–272, 1953.

[2] Kaj Bostrom and Greg Durrett. Byte pair encoding is suboptimal for language model pretraining, 2020.

[3] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2006.

[4] Lucien Le Cam. *Asymptotic Methods in Statistical Decision Theory*. Springer Science & Business Media, 2012.

[5] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, 2024.

[6] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[7] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method, 2000.

[8] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Tuo Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, 2024.

[9] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024.

[10] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.