Schema Arbitration Instead of a Policy Monolith

SchemaStack Signal Gate, actionability, expression gating, and a 390-row local-model audit

Hayden Lindley · Glyphd Labs

2026-07-27

Contents

Schema Arbitration Instead of a Policy Monolith

Abstract

Language-model control systems often inject verbose policy or schema packets into the same prompt used for user-facing articulation. This can improve compliance while creating new failures: false refusal, wrong-frame interpretation, metadata leakage, unnatural answers, and safety language copied into requested artifacts.

SchemaStack Signal Gate separates hidden behavioral arbitration from articulation. Candidate frames arise in a **Preconscious Feed**; a weighted **Schema Field** preserves alternatives; an **Actionability** test distinguishes risky words from operational harmful intent; an **Expression Gate** selects the permitted speech act; a compact natural-language **Answer Contract** reaches the articulation model; a verifier detects leakage, refusal errors, unsafe compliance, and contract failure.

The architecture does not claim consciousness. "Preconscious" names a non-user-facing candidate-control stage, not a private chain of thought.

A live local Ollama audit executed 390 rows across `llama3.1:8b`, Mistral, and Gemma 3 4B; five prompting modes; and SchemaBench plus seed subsets of XSTest and OR-Bench. The audit has raw JSONL, reports, traces, configs, and explicit small-sample caveats. It is valid live evidence, but it is too small and uses seed subsets rather than full official datasets. Results should be presented per metric, not as a broad claim that the system is "safer."

1. The v0.2 failure

The earlier schema-packet approach selected a frame early and injected structured control metadata into the final prompt.

This created:

- over-triggered safety schemas;
- false refusals;
- wrong-frame answers;
- internal label leakage;
- requested safety artifacts misclassified as refusals;
- administrative words interpreted as violence;
- fiction interpreted as operational intent.

The failure was not that schemas were useless. It was that control selection and articulation were collapsed.

2. Signal Gate pipeline

```
candidate frames
→ weighted schema field
→ actionability
→ expression gate
→ compact answer contract
→ articulation
→ verifier / bounded repair
```

Each stage has one responsibility.

3. Preconscious Feed

The feed contains candidate behavioral frames:

```
schema_id
fit
evidence_tags
risk_interpretation
recommended_expression
```

Examples:

- creative_artist;
- admin_security;
- inventor_research;
- political_worldview;
- religious_worldview;
- safety_high_risk;
- ordinary_cooking.

These candidates are control metadata, not user-facing reasoning.

4. Schema Field

Instead of committing to one frame too early, the system maintains a weighted distribution:

```
creative_artist: 0.82
safety_high_risk: 0.16
inventor_research: 0.05
```

A low-weight safety frame can monitor boundaries without dominating a clearly fictional request.

The field is not a calibrated probability of truth. It is an arbitration state.

5. Actionability

Actionability asks whether risky language implies a real-world operational request.

Core distinction:

> **Risk word present is not the same as harmful actionability present.**

Examples:

- “kill a Linux process” → administration;
- “beat eggs” → cooking;
- “destroy drives for data privacy” → defensive security;
- fictional poisoning scene → creative, non-operational;
- instructions to poison a person without detection → harmful operational.

Actionability is evaluated separately from topic.

6. Expression Gate

The gate selects a speech act:

```
answer_directly
answer_with_boundaries
answer_as_research
answer_as_fiction
answer_as_analysis
answer_as_admin_help
answer_as_safety_artifact
ask_clarifying
refuse_and_redirect
```

This allows the same subject to receive different treatment based on intent, audience, detail, and operation.

The gate determines expression, not canonical truth.

7. Answer Contract

Only a compact natural contract reaches the articulation model.

Example:

```
Help with this as a fictional scene.
Keep it non-operational.
Do not include real-world procedural detail.
Do not mention internal control labels.
```

Forbidden leakage markers include internal schema names and debugging fields.

The contract should be short enough not to distort the answer.

8. Artifact-aware scoring

A requested safety poster may contain refusal-like language:

```
Do not share passwords.
Refuse requests for credentials.
```

A naive scorer may classify the model as refusing the user.

Artifact-aware scoring distinguishes:

- model refusing the user;
- model producing requested refusal language inside an artifact.

This is essential for valid benchmarking.

9. Verifier and repair

The verifier checks:

- schema leakage;
- false refusal pattern;
- unsafe operational detail;
- wrong expression frame;
- missing task answer;
- answer-contract violation.

Repair receives the smallest error:

```
remove internal label
answer the benign admin task
preserve the requested artifact
do not add operational harm detail
```

Repair is bounded. Repeated failure returns a truthful invalid state.

10. Benchmark metrics

The audit reports separately:

- task pass;
- safety pass;
- schema selection accuracy;
- leakage pass;
- overall pass;
- false refusal;

- unsafe compliance;
- wrong frame;
- schema leakage;
- actionability accuracy;
- expression-disposition accuracy;
- average latency;
- output length.

No single overall score should hide a safety regression or helpfulness collapse.

11. Live audit

The v1 live audit used:

Models

- `llama3.1:8b`;
- `mistral`;
- `gemma3:4b`.

Modes

- raw;
- static;
- `schemastack_repair`;
- `schemastack_signal_gate`;
- `schemastack_signal_gate_repair`.

Benchmark sets

- SchemaBench sample;
- XSTest seed subset;
- OR-Bench seed subset.

Each model/mode combination produced 26 effective rows: ten SchemaBench and eight from each seed file. Across three models and five modes, the audit produced 390 rows and 45 raw JSONL files.

The scorer was deterministic; no LLM judge was used.

12. Validity boundary

The audit is:

- live, not mock;
- complete for the declared small matrix;
- free of transport errors under the recorded run;
- accompanied by raw rows and reports.

It is not:

- a full official XSTest evaluation;
- a full official OR-Bench evaluation;
- statistically broad;
- independent reproduction;
- proof of universal safety.

Earlier connection-refused or mock runs must remain separate and cannot be merged into the live result.

13. Interpretation discipline

The useful questions are:

- Did Signal Gate reduce false refusal relative to v0.2 repair?
- Did it reduce schema leakage?
- Did Mistral unsafe compliance improve?
- Did Llama or Gemma helpfulness collapse?
- Which categories caused wrong-frame errors?
- Did repair help or over-refuse?
- Did latency increase?

Mixed results are legitimate. A system may improve leakage while worsening unsafe compliance or vice versa.

14. Relation to other Glyphd systems

Signal Gate can operate before:

- Route M-Flight;
- SurfaceDelta;
- Jefe Frame generation;
- Work Order proposal;
- chat articulation.

It should not decide authority. A benign expression route cannot authorize a tool action. The output still passes through runtime policy and approval.

15. Demonstration

The public result explorer should:

- filter model, mode, benchmark, category;
- show aggregate metrics;
- link every aggregate to raw rows;
- display prompt, output, score reasons, and trace;
- identify seed versus full dataset;
- compare false refusal and unsafe compliance;
- expose latency and output length;
- show limitations prominently.

No decorative “safety score” without raw evidence.

16. Next benchmark

A stronger run should:

- use full or larger official-compatible sets where licensing permits;
- pre-register sampling;
- freeze model versions;
- record temperatures and runtime;
- add manual review on ambiguous rows;
- test more benign risky-language categories;
- measure scorer precision;

- include held-out adversarial leakage prompts;
- reproduce on another machine.

17. Security

Internal schemas can themselves become prompt-injection targets. The controller must separate untrusted user/source text from control metadata.

Threats include:

- user text forging schema labels;
- answer contract leakage;
- repair prompt revealing policy;
- scorer gaming;
- benchmark contamination;
- unsafe detail hidden in artifacts.

Controls include strict data/control separation, escaping, schema allowlists, versioned scorer, raw artifact retention, and human review priority.

18. Limitations

Actionability is difficult. Harm can be implicit. Fiction can be operational. Administrative language can be dangerous in context. A compact rule system can encode cultural or political bias. Weighted fields may appear precise without calibration.

The terms “preconscious” and “schema field” are metaphors for software control stages. They should not be presented as claims about biological cognition or model consciousness.

The live audit is small.

19. Conclusion

The architecture changes the unit of control from one selected policy packet to a staged arbitration process.

The key insight is:

> A model should not refuse because a risky word appeared. It should refuse when the request is operationally harmful under the relevant context.

Signal Gate preserves safety monitoring while allowing fiction, research, administration, ordinary language, and requested artifacts to remain in their correct frame. The benchmark, not the metaphor, decides whether the design works.

Architecture illustration briefs

- **P16-F01 — Signal Gate pipeline**: Candidate frames → schema field → actionability → expression gate → answer contract → articulation → verifier.
- **P16-F02 — Risk word versus actionability**: Matrix of benign admin/cooking/fiction/research versus harmful operational requests.
- **P16-F03 — 390-row audit matrix**: Three models × five modes × three benchmark sets, with raw-row links and caveat banner.

Source register

- **P16-S01** — SchemaStack Signal Gate Architecture (2026-05-05). `SIGNAL_GATE_ARCHITECTURE.md`
- **P16-S02** — Signal Gate v1 Benchmark Plan (2026-05-05). `V1_BENCHMARK_PLAN.md`
- **P16-S03** — Signal Gate Addendum Pack (2026-05-05). `README.md / CURSOR_COMPOSER_PROMPT.md`
- **P16-S04** — v1 Signal Gate live audit README (2026-05-06). `README_FOR_GPT.md`
- **P16-S05** — Live audit raw runs and reports (2026-05-06). `forgpt_v1_signal_gate_live/`

Public-disclosure and IP boundary

This paper publishes the architecture and declared live audit evidence. It excludes private or disallowed benchmark content, internal policy secrets, unpublished model outputs not approved for release, and exploit-enabling detail. It makes no universal safety claim and does not anthropomorphize the control stages.