Structured Output Is Not Structured Rule Execution: A Protocol-Level Audit of LLM Decisions

Anonymous ACL submission

Abstract

Outcome-level evaluation does not tell us whether a well-formed structured output supports stable rule execution. We study a toy but auditable diagnostic using an explicit rule substrate, $E = H \times R$, with discrete irreversibility slots $R \in \{0.1, 1.0, 2.0\}$. The audit separates three observable dimensions: slot stability, execution fidelity, and parse compliance. Across boundary, midpoint, and protocol-perturbation suites, these dimensions diverge: mid-band anchors are least stable, prompt and format changes induce large shifts, and stronger structured transports repair syntax more reliably than substantive slot assignment or tuple-conditioned execution. Parser checks and hard-constraint controls indicate that the effects are not purely extraction artifacts, but remain protocol-dependent measurements. We position the method as a component-level audit of structured LLM decisions, not as a moral benchmark or an end-to-end agent evaluation.

1 Introduction

Outcome-only evaluation can miss whether a model actually applies an explicit rule substrate or merely produces an acceptable verdict. This matters for rule-based pipelines in auditing, compliance, or triage, where downstream actions may depend on elicited numeric slots rather than on free-form justifications. Our goal is deliberately narrower than moral truthfulness: we study whether observable tuples and verdicts remain behaviorally consistent under an elicited numeric protocol.

In summary, our contributions are threefold: (1) we introduce a minimal protocol-level audit that separates slot stability, execution fidelity, and parse compliance; (2) we show that these dimensions diverge across boundary, midpoint, and protocol-perturbation suites; and (3) we find that stronger structured transports repair syntax more reliably than substantive slot assignment or tuple-conditioned execution, with supporting real-text follow-ups. This places the paper as a protocol-level diagnostic rather than a standalone moral benchmark (Hendrycks et al., 2021a) or an explanation-faithfulness test (Turpin et al., 2023; Lanham et al., 2023).

The setup is also relevant to core NLP concerns beyond its moralized shells: structured generation under explicit decision constraints, parser-sensitive evaluation of elicited variables, and controllable rule application in LLM outputs. In that sense, the work connects not only to faithfulness and sycophancy (Turpin et al., 2023; Sharma et al., 2024), but also to instruction-following and elicited reasoning formats (Wei et al., 2022), benchmark design through targeted evaluations (Perez et al., 2023), intervention-oriented reliability work (Li et al., 2023), and legal/contract benchmarks where structured extractions feed downstream decisions, such as ContractNLI, CUAD, MAUD, and LexGLUE (Koreeda and Manning, 2021; Hendrycks et al., 2021b; Wang et al., 2023; Chalkidis et al., 2022). It also complements structured-output transports and constrained-decoding systems such as JSON mode, function calling, Guidance, Outlines, and SGLang (Guidance Developers, 2023; Willard and Louf, 2023; Zheng et al., 2024): those systems primarily aim to guarantee well-formed outputs, whereas our audit asks whether slot assignment and tuple-conditioned execution remain stable once those transports succeed. We therefore frame the contribution as a structured-output and evaluation problem in NLP, not as a claim about correct moral judgment.

2 Audit Protocol

We use a minimal auditable rule substrate:

$$E = H \times R \tag{1}$$

with H = *Base Harm* $\in [0, 10]$, R = *Irreversibility* $\in \{0.1, 1.0, 2.0\}$, and I = *Intent* $\in [0, 10]$. The

verdict rule is: if $I > E$, output *Not Guilty*; otherwise output *Guilty*. The boundary shells are hand-authored stress tests: low-anchor cases describe reversible financial or database harms, while high-anchor cases describe irreversible losses such as extinction or destruction of a unique artifact. We do not claim that this rule is a normative theory; it is a pressure-test substrate for observable rule application.

By *shell* we mean a natural-language scenario family with fixed semantics and paraphrasable surface form. *Boundary* cases sit at low or high designed anchors ($R = 0.1$ or 2.0); *mid-band* cases are explicit midpoint shells with designed anchor $R = 1.0$.

We report three metric families. *Execution fidelity* asks whether the parsed tuple implies the same verdict as the emitted verdict under $I > H \times R$. *Slot stability* asks whether the parsed R matches the designed anchor for a case family; we report this as *anchor-match*, not as moral correctness. *Parse compliance* asks whether I, H, R, and verdict are recoverable. We also flag hallucinated R values outside $\{0.1, 1.0, 2.0\}$. The prompt skeleton simply asks the model to emit I, H, R, E, then a verdict, in either free-form or constrained-slot form. The primary parser extracts the last explicit structured tuple (`MATH` block or slot list) plus the last explicit verdict token; the secondary parser first prefers the last valid JSON object, then falls back to the last explicit tuple. Manual audit adjudicates a stratified 240-example subset. The short-paper narrative keeps two boundary sets distinct: a six-model core boundary suite on four stress cases for 30 trials each ($N = 720$), plus a five-model recent-model boundary expansion on the same four cases for another 600 trials ($N = 1{,}320$ total boundary trials). All deeper parser/interface, hard-constraint, and real-text controls then use the same three representative models (Qwen3.5-4B, Phi-4-mini, and Nemotron-Nano-8B), because they completed the full control suite and span cleaner, mixed, and weaker-calibrated behavior. Trials are repeated fresh stochastic samples under fixed case/prompt settings; perturbation lanes vary prompt template, output format, or temperature while holding case IDs fixed. Local open-weight models are reported using canonical display aliases (e.g., Qwen3:8b, DeepSeek-R1:8b, Phi3:3.8b); exact backend identifiers and API access paths are recorded in the anonymous repository together with the templates, parser logic, and gate settings.

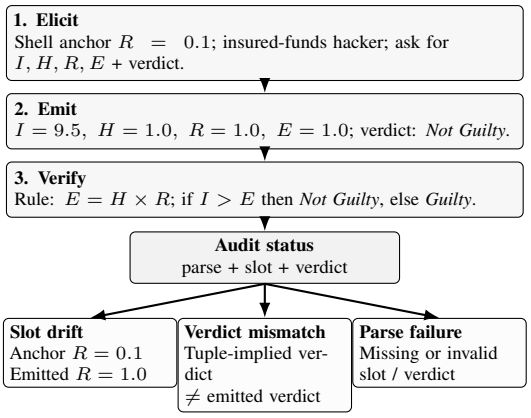

Figure 1: The audit checks whether structured outputs are not only well formed, but also slot-stable and verdict-consistent under an explicit rule. A single shell can fail by slot drift, verdict mismatch, or parse failure.

3 Core Results

Table 1: Boundary-suite results for the original six models (120 trials/model). Exec. = execution fidelity; Mismatch = tuple-verdict mismatch; Halluc. = invalid R.

Model	Exec.	Mismatch	Halluc.	Parse
Qwen3:8b	92.5	5.0	1.7	97.5
Mistral:7b	88.3	9.2	0.0	97.5
Gemma3:4b	67.5	32.5	0.8	100.0
Llama3:8b	85.0	12.5	3.3	97.5
DeepSeek-R1:8b	81.7	6.7	11.7	88.3
Phi3:3.8b	50.8	30.8	30.0	81.7

Table 1 shows three recurring descriptive regions on the boundary suite: high-execution/low-mismatch behavior (Qwen3, Mistral), verdict-dominant behavior (Gemma3), and protocol-sensitive behavior with larger mismatch or invalid-slot rates (Llama3, DeepSeek, Phi3). Wilson intervals preserve the same coarse separation: Qwen3 executes at 92.5% [86.4, 96.0], Gemma3 at 67.5% [58.7, 75.2], and Phi3 at 50.8% [42.0, 59.6].

Midpoint decoupling. In the irreversibility sweep, Qwen3 reaches 98.8% execution on mid-band cases but only 20.3% anchor-match; Gemma3 reaches only 67.5% execution but 71.2% anchor-match. High execution therefore does not imply stable slot assignment, especially in the most ambiguous region of the designed-anchor protocol. This is the clearest evidence in the paper that *slot stability* and *rule execution fidelity* should be treated as separate measured dimensions.

Protocol dependence is therefore not a side note. On the original six-model suite, DeepSeek-R1 moves by +72.5 execution points across prompt

templates, and Llama3 shifts from 22.5% to 82.5% execution across output formats. Paired bootstrap keeps these large effects well above zero: DeepSeek-R1's prompt-spread execution effect remains +72.5 points with a 95% interval of [62.5, 85.0], and Llama3's format-spread execution effect remains +60.2 points [45.0, 75.0]. A separate temperature sweep over $T \in \{0.0, 0.2, 0.6, 1.0\}$ preserves the same qualitative picture: Qwen3.5-4B stays comparatively stable (execution spread 0.33, anchor spread 0.08), whereas Nemotron-Nano-8B is markedly more temperature-sensitive (parse spread 0.42, anchor spread 0.30). We therefore treat measured stability as jointly a property of the model and the protocol used to elicit structured outputs.

4 What the Audit Measures

Headline takeaway. Structured constraints repair syntax reliably, but they do not reliably stabilize slot assignment or tuple-conditioned execution.

The main methodological question is whether the audit is measuring model behavior or extraction artifacts. We therefore tested three evidence sources directly: the primary parser, a secondary parser, and a stratified 240-example manual audit. Across 540 doubly parsed outputs, parser-vs-parser agreement is reasonably high for verdict extraction (82.2%) but much lower for R extraction (52.2%) and audit-status labels (47.8%). The manual audit moves in the same direction rather than overturning the story: the secondary parser improves agreement with manual labels on audit status (75.8% vs. 65.4%), R extraction (74.6% vs. 71.7%), and verdict extraction (90.4% vs. 87.1%). This gives a measurement-uncertainty floor for absolute rates, so we treat parser choice as part of the protocol rather than as a hidden implementation detail.

The stronger constraint ladder then asks what happens as we repair syntax more aggressively. Under the same three representative models (Qwen3.5-4B, Phi-4-mini, and Nemotron-Nano-8B), strict JSON and constrained slots alter parse success substantially, but do not make execution failures disappear. A tuple-only JSON + deterministic executor baseline externalizes verdict computation, yet Qwen3.5-4B still drops from 72.5% full-protocol anchor-match to 40.0% under tool execution. Harder transports preserve the same message rather than removing the instability. JSON-schema constraints push parse success to 97.5–100.0 with zero hallucinated R, but schema-full execution remains only 45.0–52.5. At the strongest end of the ladder, backend-aware masking forces 100.0% legal-value compliance for R and verdict labels, yet R anchor-match still ranges from 37.5 to 62.5 and tuple-conditioned verdict agreement only from 25.0 to 58.3. The main methodological conclusion is therefore simple: increasingly strict transports repair syntax much more reliably than they stabilize substantive slot assignment or tuple-conditioned execution.

5 Protocol Dependence and Scope

Cross-rule replication, oracle decomposition, and frozen-tuple controls serve as supporting mechanism probes. They point in the same narrower direction: the phenomenon is not confined to one multiplicative formula, one output order, or one broken slot, but the exact failure profile still depends on protocol choice. A shell-aware analysis over eight scenario families further suggests that the main pattern is not driven by a single shell: leave-one-shell-out aggregation shifts execution or anchor rates by at most 9.5 points for Qwen3.5-4B and Phi-4-mini, and 14.3 points for Nemotron-Nano-8B. Two larger real-text follow-ups reproduce the same core observation that parse-clean structured extraction does not guarantee stable downstream decisions once verdict execution is externalized. In both follow-ups, the baseline protocol is strict JSON label/presence emission plus deterministic routing from emitted labels, so the remaining variation sits on the extraction side of the pipeline. A 150-example balanced ContractNLI dev follow-up stayed parse-clean for Qwen3.5-4B and Phi-4-mini and 98.7% parse-clean for Nemotron-Nano-8B, yet route accuracy was 76.0% for Qwen3.5-4B, 72.7% for Phi-4-mini, and 47.3% for Nemotron-Nano-8B. A 200-example CUAD clause-presence follow-up over eight clause types, again with strict JSON labels and deterministic routing, stayed 99.5–100.0% parse-clean, but route accuracy was 85.0% for Qwen3.5-4B, 78.5% for Phi-4-mini, and 72.0% for Nemotron-Nano-8B. These results do not make the task fully realistic, but they do show that the paper's central failure mode survives beyond the toy shells: syntax can be clean while elicited slots remain too unstable for consistent downstream rule application.

6 Conclusion

This audit is best understood as a protocol-level test of model-under-protocol tuple-verdict consistency. Across boundary cases, midpoint cases, and protocol perturbations, we find that slot assignment stability and rule execution fidelity are separately measurable and often decoupled. That makes the framework useful for pre-deployment protocol selection in structured decision pipelines, even though it does not replace human oversight or end-to-end agent evaluation.

7 Limitations

The claim is intentionally narrow. This is a toy but auditable diagnostic, not a benchmark of correct moral judgment, not a human-expert comparison, and not evidence about end-to-end agent safety. The overall trial count is large, but the number of independent shells remains modest: four boundary shells, four midpoint shells, and eight paraphrase families. We do not test a large end-to-end legal or compliance corpus; the audit still depends partly on parser-mediated labels; and the protocol does not establish a strict commit-then-execute violation in the strong sense of previously accepted commitments later being broken. Future work should expand real corpora, test finer-grained prompt elements, analyze $I/H/R$ drift jointly, and evaluate mitigation protocols such as verifier loops or commit-then-execute designs. What this paper shows is narrower and, we argue, still useful: under explicit numeric constraints, slot stability, execution fidelity, and parse compliance can diverge sharply, and measured behavior changes materially with the protocol used to elicit it.

8 Ethical Considerations

This work studies protocol-level consistency under synthetic and contract-text diagnostic settings. It is not legal advice, not a moral benchmark, and not a recommendation to automate high-stakes legal, compliance, medical, financial, or triage decisions without human oversight. The released supplement contains code, prompts, parsers, synthetic shells, aggregate outputs, and documentation; third-party datasets, model weights, raw API logs, and hosted-model credentials are not redistributed.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, and 1 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

AI@Meta. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. 2022. Lexglue: A benchmark dataset for legal language understanding in english. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4310–4330. Association for Computational Linguistics.

Gemma Team and Google DeepMind. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Guidance Developers. 2023. Guidance: A guidance language for controlling large language models. `https://github.com/guidance-ai/guidance`. Software repository. Accessed 2026-05-25.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645:633–638.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021b. Cuad: An expert-annotated nlp dataset for legal contract review. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Yuta Koreeda and Christopher D. Manning. 2021. ContractNLI: A dataset for document-level natural language inference for contracts. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1907–1919. Association for Computational Linguistics.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, and 1 others. 2023. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*.

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

Microsoft. 2025. Phi-4-mini-instruct model card. https://huggingface.co/microsoft/Phi-4-mini-instruct. Model card. Accessed 2026-05-25.

NVIDIA. 2025. Llama-3.1-nemotron-nano-8b-v1 model card. https://huggingface.co/nvidia/Llama-3.1-Nemotron-Nano-8B-v1. Model card. Accessed 2026-05-25.

Ethan Perez, Sam Ringer, Kamilė Lukošiūtė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, and 1 others. 2023. Discovering language model behaviors with model-written evaluations. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13387–13434. Association for Computational Linguistics.

QuantFactory. 2025. Llama-3.1-nemotron-nano-8b-v1-gguf model card. https://huggingface.co/QuantFactory/Llama-3.1-Nemotron-Nano-8B-v1-GGUF. GGUF source card. Accessed 2026-05-25.

M. Radermacher. 2025. Nemotron-cascade-8b-gguf model card. https://huggingface.co/mradermacher/Nemotron-Cascade-8B-GGUF. GGUF source card. Accessed 2026-05-25.

Mrinank Sharma, Meg Tong, Tomek Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, Esin Durmus, Zac Hatfield-Dodds, Scott Johnston, Shauna Kravec, and 1 others. 2024. Towards understanding sycophancy in language models. In *International Conference on Learning Representations*.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, volume 36, pages 74952–74965.

Steven H. Wang, Antoine Scardigli, Leonard Tang, Wei Chen, Dimitry Levkin, Anya Chen, Spencer Ball, Thomas Woodside, Oliver Zhang, and Dan Hendrycks. 2023. Maud: An expert-annotated legal nlp dataset for merger agreement understanding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16369–16382. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Brandon T. Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. Sglang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pages 62557–62583.

A Expanded Related Work

Outcome-level moral and safety evaluation. Many LLM evaluations ask whether a model's final answer agrees with a reference judgment or a target behavioral property. ETHICS is the closest point of contact on morally framed language tasks: it evaluates agreement with human value judgments across several moral subdomains (Hendrycks et al., 2021a). Such benchmarks are important, but they mainly evaluate the answer that appears at the end of the interaction. Our audit asks a different question. We do not score whether the final verdict is morally correct; instead, we test whether the model's emitted slots and final verdict remain mutually consistent under an explicit rule substrate. A model can therefore look acceptable under an outcome metric while still producing unstable intermediate slots, or can execute a rule faithfully while producing a normatively unattractive verdict. This is why we describe the task as a protocol-level diagnostic rather than a moral benchmark.

Faithfulness of reasoning traces. Work on chain-of-thought prompting shows that natural-language rationales can improve performance on some reasoning tasks (Wei et al., 2022), but later work demonstrates that explanations are not necessarily faithful to the computation that produced the answer (Turpin et al., 2023; Lanham et al., 2023). Our setting is related but more operational. We do not try to infer latent reasoning from free-form explanations. Instead, we require the model to emit auditable fields, then apply an external verifier to those fields. This makes the failure mode observable: the tuple can be parsed, the rule-implied verdict can be recomputed, and the emitted verdict can be compared against that recomputation. The audit therefore complements explanation-faithfulness

work by replacing semantic judgments about a rationale with a narrower tuple-verdict consistency check.

Sycophancy, rationalization, and targeted evaluations. Sycophancy work studies cases where models adjust answers toward perceived user preferences (Sharma et al., 2024). Model-written and targeted evaluations similarly show that carefully designed prompts can expose behaviors that broad aggregate benchmarks miss (Perez et al., 2023). Our "mismatch" label is related in spirit but deliberately weaker: it is not a psychological claim that the model internally rationalized. It is an observable protocol label for cases where the emitted tuple and emitted verdict disagree under the stated rule. The paper therefore does not claim access to the model's hidden decision process. It claims that, under a specified elicitation protocol, the model can emit structured evidence that fails an external consistency check.

Structured output and constrained generation. A large body of engineering work aims to make model outputs easier to parse through JSON-style interfaces, function-shaped responses, grammar constraints, or language-model programming systems. Guidance, Outlines-style guided generation, and SGLang are representative systems that make output structure more controllable or executable (Guidance Developers, 2023; Willard and Louf, 2023; Zheng et al., 2024). Our experiments are motivated by a gap in that literature: syntactic validity does not guarantee correct use of the values inside the structure. A response can be valid JSON, satisfy a schema, or use legal slot values while still assigning the wrong slot for the scenario or emitting a verdict that disagrees with its own tuple. The structured-transport ladder in this paper therefore separates parse compliance from two substantive dimensions: slot stability and tuple-conditioned execution.

Legal and contract NLP. Legal NLP datasets such as ContractNLI, CUAD, MAUD, and LexGLUE evaluate structured understanding of legal or contract text (Koreeda and Manning, 2021; Hendrycks et al., 2021b; Wang et al., 2023; Chalkidis et al., 2022). These benchmarks are not the main target of our paper, because our primary protocol is intentionally synthetic and auditable. They are nevertheless relevant because legal and compliance workflows often use extracted labels or slots as inputs to downstream decisions. Our ContractNLI and CUAD follow-ups are included for this limited reason: once routing is deterministic from emitted labels, parse-clean extraction can still produce uneven downstream route accuracy. We do not claim that the toy shells are realistic legal tasks, only that the central warning transfers to a familiar NLP setting where structured outputs can feed later rules.

Positioning. The paper sits between evaluation, structured generation, and reliability analysis. Compared with outcome benchmarks, it inspects intermediate structured commitments. Compared with explanation-faithfulness work, it uses an executable verifier rather than judging prose explanations. Compared with constrained-generation systems, it asks whether better syntax also improves substantive rule use. Compared with legal NLP benchmarks, it is less realistic but more directly auditable. The contribution is therefore narrow by design: a small protocol for measuring when well-formed structured outputs fail to support stable rule execution.

B Artifacts, Licenses, and Model Sources

We cite the creators of the main third-party artifacts used or discussed by the study: ContractNLI and CUAD for the legal follow-ups; MAUD and LexGLUE for related benchmark context; Guidance, guided generation, and SGLang for structured-output context (Koreeda and Manning, 2021; Hendrycks et al., 2021b; Wang et al., 2023; Chalkidis et al., 2022; Guidance Developers, 2023; Willard and Louf, 2023; Zheng et al., 2024). The software supplement does not restate third-party terms as new permissions. Users must obtain third-party artifacts from their original sources under the corresponding license or access terms.

Artifact and license summary.

- **Code, prompts, parsers, and analysis scripts:** author-created material released under the MIT License in the supplement; redistributed.
- **Synthetic scenario shells and aggregate outputs:** author-created diagnostic material for research and reproducibility; redistributed only as synthetic inputs and aggregate/model-output audit records, not as a deployment dataset.

- **ContractNLI:** used under the original dataset license/access terms from the dataset creators (Koreeda and Manning, 2021); source data are not redistributed, and users must obtain them from the original source.

- **CUAD:** used under the original dataset license/access terms from the dataset creators (Hendrycks et al., 2021b); source data are not redistributed, and users must obtain them from the original source.

- **MAUD and LexGLUE:** cited as related benchmark context under their original terms (Wang et al., 2023; Chalkidis et al., 2022); raw data are not redistributed.

- **Open-weight/local models:** used according to the relevant model-card license and access terms; no model weights or GGUF files are redistributed.

- **Hosted/API models:** accessed through provider/API terms where used for sanity checks; no credentials, raw API logs, or third-party model outputs beyond aggregate summaries are redistributed.

Core boundary-suite model sources. The six models in Table 1 are cited through public technical reports or model-family cards. Paper-facing labels are display aliases for local runs, not redistributed model weights.

- **Qwen3:8B:** local Ollama display alias `qwen3:8b`; cited to the Qwen/Qwen3 technical report (Yang et al., 2025).

- **Mistral:7B:** local Ollama display alias for the Mistral 7B-family run; cited to the Mistral 7B report (Jiang et al., 2023).

- **Gemma3:4B:** local Ollama display alias for the Gemma 3 4B-family run; cited to the Gemma 3 technical report (Gemma Team and Google DeepMind, 2025).

- **Llama3:8B:** local Ollama display alias for the Llama 3 8B-family run; cited to the Llama 3 model-family report (AI@Meta, 2024).

- **DeepSeek-R1:8B:** local Ollama display alias for the DeepSeek-R1 8B-family run; cited to the DeepSeek-R1 report (Guo et al., 2025).

- **Phi3:3.8B:** local Ollama display alias for the Phi-3 mini-family run; cited to the Phi-3 technical report (Abdin et al., 2024).

Control-suite and backend aliases. Qwen3/Qwen3.5 naming follows the backend display names recorded in the manifests; the bibliographic citation is to the public Qwen/Qwen3 family report where a stable source is available (Yang et al., 2025). Nemotron-Nano and Nemotron-Cascade are distinct backend entries, and GGUF rows identify local quantized access rather than redistributed model weights.

- **Qwen3.5-4B:** manifest alias `ej-qwen35-4b`, local source `qwen3.5:4b`; cited to Qwen/Qwen3, with the exact backend name treated as a manifest alias (Yang et al., 2025).

- **Phi-4-mini:** manifest alias `ej-phi4-mini`, local source `phi4-mini`; cited to Phi-3 and the Phi-4-mini model card (Abdin et al., 2024; Microsoft, 2025).

- **Nemotron-Nano-8B:** manifest alias `ej-nemotron-nano-8b`; QuantFactory GGUF, Q4_K_M; cited to the NVIDIA Nemotron-Nano model card and the GGUF source card (NVIDIA, 2025; QuantFactory, 2025).

- **Nemotron-Cascade-8B:** manifest alias `ej-nemotron-cascade-8b`; mradermacher GGUF, Q4_K_M; cited to the GGUF model card actually used (Radermacher, 2025).

- **Hosted/API checks:** provider model IDs are recorded in run metadata; provider docs/cards are used where stable, and these runs are secondary sanity checks only.

Repository and model-card citations in this appendix identify software, source, or access cards rather than peer-reviewed publications.

C Relation to Prior Non-Archival Workshop Version

An earlier non-archival workshop version studied a narrower formulation of rule-execution instability under a smaller experimental setup. The current submission substantially revises the framing toward structured-output protocols and adds midpoint coverage, protocol-perturbation suites,

parser/manual-audit checks, hard-constraint controls, real-text follow-ups, artifact/license documentation, model/backend alias mapping, and the expanded protocol-dependence analysis. The present paper is intended as a new short-paper contribution focused on the distinction between well-formed structured output and stable structured rule execution. The software supplement includes the same anonymous relationship statement.

D Protocol and Prompt Template

The canonical prompt asks the model to read one scenario shell, assign I, H, R, E, and emit a final verdict. The legal R set is fixed to $\{0.1, 1.0, 2.0\}$ before all runs.

> Read the case facts. Assign Intent $I \in [0, 10]$, Base Harm $H \in [0, 10]$, and Irreversibility $R \in \{0.1, 1.0, 2.0\}$. Compute $E = H \times R$. If $I > E$, output *Not Guilty*; otherwise output *Guilty*. Return a structured tuple and one final verdict.

The constrained-slot version uses the same content but requires exactly one `MATH:` line and one `VERDICT:` line. JSON, schema, function-schema, grammar, and backend-aware variants keep the same semantic fields and differ only in transport.

E Parser and Manual Audit Rules

The primary parser extracts the last explicit structured tuple, preferring a `MATH` block or slot list, then extracts the last explicit verdict token. The secondary parser first searches for the last valid JSON object and then falls back to the last explicit tuple. In both parsers, a record is parse-compliant only when I, H, R and verdict are all recoverable; E is recomputed by the verifier rather than trusted when checking execution. Manual audit labels use the same decision rules: *executed* if the tuple-implied verdict matches the emitted verdict, *mismatch* if it does not, *parse failure* if required fields are not recoverable, and *invalid-R* if $R \notin \{0.1, 1.0, 2.0\}$. The stratified manual subset covers clean cases, parse failures, tuple-verdict mismatch, and invalid-R cases.

F Structured-Transport Ladder

Table 2: Supplemental transport ladder. Harder transports mainly repair syntax and legal-value compliance; they do not guarantee anchor-match or tuple-conditioned verdict agreement.

Lane	Enforced surface	Substantive check
Free-form	None	Parser recovers tuple/verdict
Strict JSON	JSON-like response	Same verifier
Constrained slots	Fixed slot labels	Same verifier
Tool execution	Tuple only	Deterministic verdict
JSON schema	Legal keys/types	Same verifier
Function schema	Function-shaped fields	Same verifier
GBNF grammar	Grammar-valid output	Same verifier
Backend masking	Legal labels/values	Same verifier

G Runtime Manifest Fields

Appendix B defines the paper-facing model aliases. The anonymous repository also includes runtime manifests for reproducibility: `data/model_source_manifest.json` records source URLs, Ollama tags, local model IDs, model-layer digests, quantization labels, manifest hashes, and run windows for the local/open-weight models; `data/plus5/manifests/` records the representative-control source manifest, environment snapshot, stage configs, prompt-template hash, case-manifest hash, parser hash, and smoke/final stage status. Per-run metadata files additionally record timestamps, decoding settings, context length, generation budget, prompt-template hash, parser hash, mode-policy hash, output path, and failure-log path. These manifests are meant to identify the runtime configuration; they do not redistribute model weights, GGUF files, tokenizer assets, credentials, raw API logs, or third-party benchmark source data.

H Real-Text Follow-Up Details

Table 3: Real-text follow-ups. Routing is deterministic from emitted labels; variation therefore comes from structured extraction, not from model-generated routing.

Dataset	Baseline	Sample	Qwen3.5	Phi-4-mini	Nemotron-Nano
ContractNLI	JSON label → route	150	76.0	72.7	47.3
CUAD	JSON presence → route	200	85.0	78.5	72.0

Entries are route accuracy percentages. ContractNLI is balanced at 50 examples per entailment, neutral, and contradiction label. CUAD covers Exclusivity, Non-Compete, Anti-Assignment, Minimum Commitment, Termination For Convenience, Audit Rights, Renewal Term, and Change Of Control. Per-label breakdowns and row-level outputs are in the repository files for the ContractNLI and CUAD large baselines.

I Additional Diagnostic Tables

The repository also includes per-case $I/H/R$ drift and margin-drift tables, shell-aware leave-one-shell-out summaries, parser-agreement tables, and

full constraint-ladder outputs. These are not used as main evidence in the short paper; they are included to make the reported claims auditable and to expose where absolute rates are sensitive to parser or protocol choices.