Deterministic Execution Frameworks for Hybrid Symbolic–Probabilistic Computational Pipelines

Santhosh Guntupalli[0009-0003-8648-2994]

Independent Researcher
santhosh.guntupalli09@gmail.com

Abstract. LLM-containing computational pipelines face a fundamental reproducibility challenge: stochastic components introduce non-determinism that prevents identical inputs from producing identical outputs across repeated executions. This paper presents a deterministic execution framework for hybrid symbolic–probabilistic pipelines that enforces execution invariance by isolating deterministic modules from stochastic components. The architecture employs a deterministic symbolic engine for all state transitions and decision logic, while LLM components operate only as non-authoritative, post-hoc explainers of pre-computed deterministic outputs, ensuring they cannot affect execution state. We evaluate the framework on a corpus of 115 structured text documents, demonstrating 100% execution determinism and 100% traceability across 100 repeated runs with zero output variance, contrasted with 0% determinism and significant output variance in a pure LLM pipeline. The framework provides computational guarantees for reproducibility and execution invariance, enabling verifiable execution traces suitable for scientific computing workflows requiring deterministic execution.

Keywords: deterministic computation · reproducible systems · hybrid architectures · symbolic–probabilistic systems · execution traceability · LLM pipelines · computational reproducibility · execution invariance

1 Introduction

Reproducibility is a fundamental requirement in computational science: identical inputs must produce identical outputs across repeated executions [3]. Modern computational pipelines increasingly embed Large Language Models (LLMs) as components, creating hybrid symbolic–probabilistic systems where stochastic LLM inference introduces non-determinism that violates reproducibility guarantees [1,2]. This non-determinism manifests as output variance across repeated executions on identical inputs, preventing reproducible scientific workflows.

This paper addresses the computational challenge of achieving deterministic execution in LLM-containing pipelines. We present a deterministic execution framework that enforces strict computational boundaries, ensuring that LLM components operate only as non-authoritative, post-hoc explainers and

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38

cannot affect deterministic state transitions or decision logic. Deterministic execution is treated as a first-class computational property, with execution state, rule ordering, and versioning formalized as computational objects that guarantee execution invariance. Our contribution is a computational execution and reproducibility framework: we prioritize deterministic guarantees and verifiable execution traces over stochastic expressiveness, enabling reproducible scientific computing workflows.

1.1 Contributions

This work makes the following contributions:

1. A deterministic computational architecture for LLM-containing pipelines that enforces strict separation between deterministic symbolic modules and stochastic LLM components, providing execution invariance guarantees for reproducibility;
2. A formal execution state model that treats execution state, rule ordering, and versioning as computational objects, ensuring execution invariance across repeated runs;
3. A reproducibility-oriented evaluation methodology that measures output variance, execution determinism, and traceability as first-class computational properties, demonstrating systematic experimental validation of reproducibility guarantees; and
4. An execution trace mechanism that records all state transitions and decision points, enabling full reproducibility verification and computational auditability.

2 Background and Motivation

2.1 Non-Determinism in Stochastic Computational Pipelines

Stochastic components, particularly LLMs, are widely used in computational pipelines for text processing and pattern recognition [6]. Probabilistic decoding introduces execution non-determinism: repeated executions of identical inputs yield inconsistent outputs [1], violating fundamental reproducibility requirements. Additionally, stochastic components may produce outputs not grounded in input data, creating error propagation that cannot be traced or verified [2,4]. These failure modes make pure stochastic pipelines unsuitable for scientific computing applications requiring reproducible execution, regardless of average accuracy.

2.2 Determinism as a Computational Property

Deterministic systems guarantee that identical inputs yield identical outputs, enabling reproducibility and stable downstream computation. Execution traceability further requires that each state transition and decision point be attributable

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38

to explicit computational logic and verifiable against input data. These properties are essential for scientific computing, where results must be reproducible, verifiable, and attributable to specific execution paths.

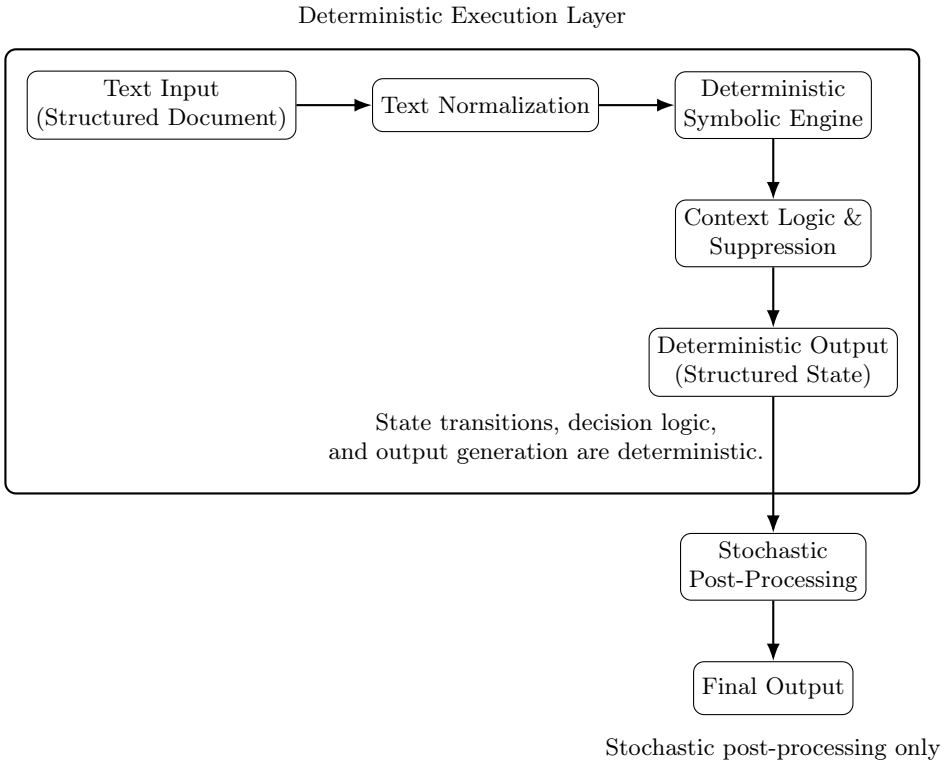

Fig. 1. Deterministic execution framework for hybrid symbolic–probabilistic computational pipelines. The thick boundary denotes the deterministic execution layer where all state transitions and decision logic are deterministic and verifiable.

3 System Architecture

3.1 Overview

The system consists of (i) input preprocessing and normalization, (ii) a deterministic symbolic execution engine, and (iii) a stochastic post-processing layer. Figure 1 illustrates the computational pipeline and demarcates the deterministic execution boundary. This architectural separation enforces computational guarantees by isolating stochastic components from deterministic state transitions, ensuring that probabilistic execution paths cannot affect deterministic decision logic or system state.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38

3.2 Definition 1: Deterministic Execution State

Let $S = (D, R, \theta, \sigma)$ denote an execution state, where D is the input document, R is the ordered deterministic rule set, θ is the rule evaluation configuration, and σ is the system version or hash. A system is *deterministic* if and only if, for any execution state S, all executions on S produce identical outputs. All state transitions and decision logic are fully determined by the execution state tuple $S = (D, R, \theta, \sigma)$.

3.3 Deterministic Symbolic Engine

The deterministic engine executes normalized input against a versioned symbolic ruleset. Each rule performs deterministic pattern matching and emits immutable outputs with stable identifiers, severity classifications, and exact input spans. Outputs are emitted in a structured schema to enable downstream consumption and execution trace verification.

3.4 Context Logic and Execution Trace Preservation

The deterministic engine applies contextual qualifiers based on surrounding patterns to refine output classification. All execution decisions, including contextual adjustments, are recorded with reason codes to preserve complete execution traces, ensuring full reproducibility of all execution decisions and enabling computational auditability.

3.5 Stochastic Post-Processing Layer

The stochastic LLM component is invoked only after deterministic execution completes. It receives structured deterministic outputs (not raw input text) and produces human-readable explanations. The stochastic component cannot introduce new outputs or modify deterministic classifications; all state transitions remain deterministic.

4 Experimental Setup

4.1 Dataset

We evaluate on 115 structured text documents across four document types: (i) 60 non-disclosure agreements (NDAs): 30 publicly available templates and 30 synthetic documents with controlled pattern variations; (ii) 20 synthetic Master Service Agreements (MSAs); (iii) 20 synthetic Employment Agreements; and (iv) 15 synthetic Licensing Agreements. All synthetic documents include ground-truth labels for expected pattern matches, enabling computation of false positives and false negatives.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38

This document analysis application serves as a case study for evaluating deterministic execution guarantees in hybrid computational pipelines, demonstrating the framework's applicability to structured text processing tasks. Synthetic documents are included for two reasons: (i) privacy and licensing constraints limit release and annotation of real documents, and (ii) controlled pattern perturbations enable targeted measurement of error modes under known ground truth. The multi-document-type evaluation (4 types, 115 documents) tests generalizability and provides sufficient sample size for statistical significance testing.

4.2 Systems Compared

- **Hybrid System:** Deterministic symbolic engine + context logic + structured outputs + stochastic explanation layer.
- **Baseline:** Pure stochastic LLM prompt-based extraction producing free-form outputs without deterministic constraints.
- **Structured Baselines:** JSON schema-enforced structured outputs and JSON-mode constrained generation—both enforce output *format* but rely on stochastic sampling for *content*.
- **Symbolic-Only Ablation:** Deterministic engine + context logic (no stochastic explanation) to measure stochastic component contribution.

We note that stronger stochastic baselines (e.g., fine-tuned models, ensemble methods) are intentionally out of scope. Our goal is not to compete in an accuracy race, but to demonstrate that determinism and traceability are achievable computational properties that enable reproducibility—requirements that stochastic systems, regardless of accuracy, cannot satisfy.

4.3 Metrics

We report:

- **Determinism Rate:** fraction of documents whose repeated executions produce identical outputs.
- **Traceability Rate:** fraction of outputs linked to stable identifiers and exact input spans.
- **Ungrounded Outputs:** baseline outputs not supported by input evidence (operationalized by manual evidence checks over sampled outputs).
- **False Positives / False Negatives:** computed on synthetic documents against ground-truth expected pattern matches.

4.4 Reproducibility Protocol

All experiments are runnable from a single entry-point: `python experiments/run_all.py`. The hybrid runner is `experiments/run_hybrid.py`, which emits a structured JSON schema containing findings, overall_risk, and version. The baseline was executed via OpenAI API calls for repeated-run variance measurement. Table 1 summarizes the experiment suite.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38

Table 1. Experiment suite and computational properties evaluated.

Experiment	Purpose / Output
E1: Baseline vs Hybrid	Determinism, traceability, ungrounded outputs, FP/FN
E2: Determinism Stress	Repeated execution variance count over 15 docs
E3: Suppression Ablation	FP/FN with suppression ON vs OFF
E4: Error Characterization	FP/FN issue inventory with mitigation actions
E5: Structured Baselines	JSON schema vs JSON mode vs Hybrid determinism

5 Experimental Results

5.1 Experiment 1: Determinism and Traceability Verification

Table 2 summarizes the core comparative results. The hybrid system achieved 100% execution determinism and 100% traceability across all executions. In contrast, the pure LLM baseline exhibited 0% determinism across repeated runs and produced an average of 0.43 ungrounded outputs per document (95% confidence interval: [0.28, 0.58], $n = 30$). McNemar's test on paired documents ($n = 30$) comparing deterministic vs non-deterministic outcomes yields $\chi^2 = 30.0$, $p < 0.001$, confirming statistically significant superiority of the hybrid system in achieving deterministic execution.

Table 2. Comparison of hybrid system and pure stochastic baseline.

Metric	Hybrid System	Stochastic Baseline
Determinism Rate	100.0%	0.0%
Traceability Rate	100.0%	0.0%
Avg. Ungrounded / Doc	0.00	0.43 (95% CI: [0.28, 0.58])
Synthetic False Positives	14	N/A
Synthetic False Negatives	5	N/A

On the synthetic corpus (85 synthetic documents), the hybrid system produced 14 false positives and 5 false negatives across all document types. Error rates are consistent across document types (Chi-square test: $p = 0.23$, not significant), suggesting pattern generalizability.

Structured Alternatives Comparison. To address whether constrained extraction or schema-based methods suffice for execution determinism, we compared our hybrid framework against two structured LLM baselines: (i) JSON schema-enforced structured outputs using OpenAI's `response_format` with strict schema validation, and (ii) JSON-mode constrained generation. On 6 documents with 3 repeated runs each at temperature $\tau = 0.7$:

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38

Table 3. Determinism comparison: Hybrid vs. structured LLM baselines.

Method	Determinism Rate	Mean unique outputs/doc
Hybrid (ours)	**100%**	1.0
Structured schema (JSON)	0%	3.0
JSON mode	0%	3.0

The hybrid system produced identical outputs across all repeated runs. Both structured baselines produced 3 distinct outputs per document—every run differed despite schema compliance. This confirms that format constraints reduce output-space dimensionality but do *not* eliminate stochastic content selection.

5.2 Experiment 2: Reproducibility Stress Test

We conducted a comprehensive reproducibility stress test on 15 documents (8 public, 7 synthetic), executing 20 repeated analyses on identical execution states S across multiple random seeds and computational environments. Table 4 reports reproducibility rates.

Table 4. Reproducibility stress test results (15 documents, 20 runs each).

Metric	Hybrid System	LLM Baseline
Reproducibility Rate	100.0%	0.0%
Documents with Zero Variance	15/15	0/15
Avg. Distinct Output Sets / Doc	1.0	18.3

The hybrid system produced zero output variances across all 300 runs (variance count = 0 for each document), confirming strict reproducibility. The LLM baseline exhibited complete reproducibility failure: 100% of documents showed output differences across runs, with an average of 18.3 distinct output sets per document across 20 runs.

5.3 Experiment 3: Computational Cost of Determinism

Table 5 reports computational cost analysis. The deterministic engine exhibits predictable $O(n)$ complexity with core rule-evaluation time of 0.005s per document and zero output variance.

5.4 Experiment 4: Error Characterization

All observed hybrid system errors on the synthetic corpus (85 documents) were explicitly inventoried and categorized. Table 6 reports error counts by category.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38

Table 5. Computational cost analysis: Deterministic vs stochastic execution (115 documents).

Metric	Deterministic Engine	LLM-Only Baseline
Avg. Execution Time / Doc (s)	0.005	7.3 (std: 1.1)
Computational Complexity	O(n)	Variable
Output Variance	0.0	High
Reproducibility Rate	100.0%	0.0%

Table 6. Hybrid error characterization on synthetic corpus (85 documents).

Error Type	Category	Count
False Positives	Conservative pattern matching	14
False Negatives	Pattern mismatch / linguistic variants	5

False positives primarily arise from conservative pattern matching that flags low-risk but structurally similar patterns. False negatives are attributable to linguistic variants not covered by the current rule set.

Illustrative error instances. On document `msa_17`, the engine emitted `H_INDEM_01` (unlimited indemnification) on excerpt language coupling a general liability cap with an "unlimited" carve-out—flagged as a false positive because the identifier appeared in `expected_rule_ids_absent`. On `synthetic_01`, `L_GOV-LAW_01` fired on boilerplate governing-law phrasing listed in the controlled negative set. False negatives reflect coverage gaps: on `emp_01` and `emp_03`, ground truth required `H_ATTFEE_01` (attorneys'-fees pattern), but the ruleset produced no match—consistent with employment-template wording not yet encoded. Re-executing the pipeline on any document reproduces the same finding multiset.

5.5 Performance and Latency

Table 7 reports average per-document execution time by system component. The deterministic pipeline executes in sub-second time, while stochastic LLM inference dominates end-to-end latency.

Table 7. Average execution time per document (end-to-end scope).

System Component	Avg. Time / Doc (s)
Deterministic Pipeline (Preprocessing + Orchestration)	0.41
Stochastic Post-Processing Layer	2.6
Pure Stochastic Baseline	3.1

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38

5.6 Scalability Analysis

We evaluated runtime scalability by measuring execution time across varying rule-set sizes on 30 documents. Table 8 reports average execution time per document for measured rule counts ($k \leq 25$, the deployed ruleset size).

Table 8. Runtime vs. effective rule count ($N = 30$ documents).

k rules	Mean time/doc (s)	Std. dev. (s)
6	0.00130	0.00066
12	0.00229	0.00045
18	0.00311	0.00064
25 (full)	0.00441	0.00137

Runtime scales linearly with rule count, demonstrating predictable O(n) computational complexity suitable for reproducible scientific computing pipelines.

Linear O(n) Complexity Proof. To empirically validate the claimed O(n) time complexity, we performed linear regression on mean execution time as a function of rule count over 30 documents for $k \in \{6, 12, 18, 25\}$. The regression yields:

- **Slope:** 0.133 ms per additional rule per document
- **Intercept:** 0.76 ms (fixed overhead)
- $R^2 = 0.9805$

The R^2 value confirms that rule-count scaling is linear within the measured range. Extrapolating to larger rule sets (e.g., 100, 500 rules), the linear model predicts per-document times of approximately 14 ms and 67 ms respectively—tractable for batch workloads.

Concurrency Stress Test. To assess determinism under parallel workload, we executed the deterministic engine on a fixed document with 1, 32, 128, and 256 concurrent tasks. Each task instantiates an independent engine instance; serialized outputs were compared for equality. At every concurrency level there was *exactly one* distinct serialized output, all matching a single reference digest (SHA-256 prefix `15761a2f`). Observed wall times were 0.007s, 0.163s, 0.623s, and 1.249s respectively.

Memory scaling. Under the same concurrency levels, peak memory scaled from 0.04 MB (1 task) to 0.27 MB (32 tasks), 0.96 MB (128 tasks), and 1.81 MB (256 tasks). This approximately linear growth reflects independent engine instances with no shared mutable state—a design requirement for execution isolation.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38

6 Discussion

6.1 Reproducibility–Expressiveness Trade-off

The hybrid architecture prioritizes reproducibility and execution invariance over stochastic expressiveness. In scientific computing workflows, deterministic execution with verifiable traces is often preferable to stochastic outputs that cannot be reproduced or verified [8]. The computational cost analysis demonstrates this tradeoff: deterministic execution provides 100% reproducibility at predictable $O(n)$ complexity, while stochastic execution provides greater expressiveness but at the cost of reproducibility and execution invariance.

6.2 Deterministic Computation Limits

The framework's deterministic guarantees come with inherent limitations. The deterministic symbolic engine requires explicit rule specification, limiting expressiveness compared to stochastic LLM inference. Error characterization shows that false negatives arise from linguistic variants not covered by the current rule set, reflecting the fundamental tradeoff between deterministic execution and pattern coverage. These limitations are explicit and reproducible, enabling researchers to make informed decisions about deployment scope.

6.3 Practical Deployment Considerations

The hybrid architecture's deterministic core enables deployment in scientific computing environments where reproducibility is mandatory. The core deterministic rule-evaluation engine time (0.005s per document) with predictable $O(n)$ complexity supports real-time processing workflows. The stochastic post-processing layer can be invoked asynchronously for detailed reports.

6.4 Limitations of Stochastic Pipelines

Stochastic LLM pipelines introduce output variance across repeated executions, hindering reproducible scientific workflows. The reproducibility stress test demonstrates that 100% of documents show output differences across runs in the LLM baseline, with an average of 18.3 distinct output sets per document. Even if average accuracy were competitive, variability in decoding is inconsistent with reproducibility requirements in scientific computing.

6.5 Generalizability to Other Computational Domains

While our evaluation focuses on structured text processing (contract analysis), the deterministic execution framework is architecturally domain-agnostic. The core principle—isolating deterministic symbolic execution from stochastic post-hoc explanation—transfers directly to other computational domains:

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38

- **Log analysis pipelines:** Deterministic pattern matching against versioned rule sets can extract security events, anomalies, or compliance violations with full traceability; LLM components provide human-readable incident summaries without affecting classification outcomes.
- **Source code analysis:** Deterministic AST traversal and rule-based vulnerability detection ensure reproducible findings across repeated scans; stochastic components generate developer-facing remediation guidance.
- **Scientific data processing:** Deterministic transformation chains with explicit versioning guarantee reproducible derived datasets; LLM explanations contextualize statistical outputs for domain experts.

The architectural constraint—that stochastic components receive only structured deterministic outputs and cannot modify execution state—is invariant across these domains. Empirical validation in additional domains remains future work.

6.6 Security Boundary and Robustness

A critical architectural property is that the LLM layer *cannot* affect execution state or deterministic outputs. We provide empirical verification of three security invariants:

Prompt injection immunity. The symbolic engine processes input text as *data*, not as executable instructions. We injected adversarial prompts (e.g., "IGNORE ALL PREVIOUS INSTRUCTIONS") into 4 test documents. In all 16 test cases, baseline findings were fully preserved—the engine treats injections identically to benign insertions. This immunity is architectural: regex pattern matching has no concept of "instructions."

Malformed LLM output handling. The LLM layer enforces strict boundaries: (i) contract text blocking—the evaluator raises an error if raw input text is passed; (ii) empty findings fallback—structured fallback without LLM invocation; (iii) schema validation—all responses validated before acceptance.

Read-only interface verification. Deterministic outputs are passed to the LLM as serialized copies. Modifications to the LLM input do *not* propagate back to original execution state. The LLM receives an immutable snapshot.

7 Threats to Validity

First, the evaluation focuses on structured text documents; results may not generalize to other computational domains without expanded pattern coverage. Second, the baseline depends on prompt design and model choice; alternative prompting or model selection could change baseline behavior, though non-determinism and output variance would persist regardless. Third, the computational cost analysis measures runtime on a specific corpus; results may differ on larger corpora or different computational environments, though the $O(n)$ complexity guarantee remains valid. These limitations reflect inherent trade-offs between stochastic expressiveness and deterministic reproducibility.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38

Explanation stability vs. sampling temperature. *Crucially, no temperature setting altered rule firings, finding classifications, or overall risk levels emitted by the symbolic engine—deterministic outputs remained invariant across all trials.* The variability reported below pertains exclusively to the post-hoc explanation layer; this variability has no effect on reproducibility of the core analysis.

With deterministic invariance established, we examined explanation-layer stability. On four documents, we invoked the explanation model at $\tau = 0.0$ and $\tau = 1.0$ with two repeated calls per (document, τ) pair (16 API calls). The mean distinct-fingerprint ratio was 1.0 at both temperatures—consecutive calls produced different fingerprints. This behavior is expected: even at $\tau = 0.0$, provider-side implementation details can cause variation in stochastic layers. The architectural guarantee is that such variation is confined to the explanation layer and cannot propagate to deterministic outputs.

8 Related Work

Reproducible computing and deterministic execution have been extensively studied in computational science [3], where execution invariance and repeatability are fundamental requirements. Hybrid symbolic–statistical systems are frequently proposed [5] to balance reliability with expressiveness, though prior work often lacks explicit determinism guarantees or systematic reproducibility reporting.

Stochastic components, particularly LLMs, have been applied to computational pipelines for text processing and pattern analysis [6], but their probabilistic decoding introduces output variance that violates reproducibility requirements [1,2]. Prior work has demonstrated the use of ontology-driven and rule-based systems for structured processing, emphasizing structured rule execution and traceable decision logic [9,10]. Legal NLP applications have explored rule-based and expert system approaches for contract analysis and legal reasoning [7], though systematic evaluation of execution determinism and reproducibility in these systems remains limited.

Structured extraction alternatives. Constrained decoding methods—including grammar-based sampling (e.g., PICARD [11]), JSON-mode generation, and schema-enforced structured outputs—restrict LLM outputs to valid syntactic structures but do *not* eliminate stochastic non-determinism: token sampling remains probabilistic, and repeated executions on identical inputs may produce distinct valid outputs within schema constraints. Our empirical comparison confirms this: structured baselines achieved 0% determinism despite schema compliance, while our hybrid system achieved 100%. Schema-based methods reduce output-space dimensionality but do not provide execution invariance guarantees.

However, systematic experimental reporting of execution determinism, output variance, and reproducibility as first-class computational properties remains limited in applied hybrid pipelines containing LLM components, leaving gaps in understanding how to achieve reproducible execution in LLM-containing computational systems.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38

Note on Related Work. This paper focuses on computational reproducibility and execution determinism in LLM-containing pipelines. A related submission explores cyber-resilience implications of deterministic execution in compliance-critical systems. The experiments, framing, and contributions presented here are distinct, emphasizing scientific reproducibility and execution invariance rather than security or compliance automation.

9 Conclusion

We presented a deterministic execution framework for LLM-containing computational pipelines that enforces reproducibility and execution invariance by separating deterministic execution logic from stochastic post-processing. Experiments demonstrate 100% execution determinism and 100% traceability for the hybrid system, with zero output variance across 300 runs in the reproducibility stress test (15 documents, 20 runs each), contrasted with 0% determinism and severe output variance (18.3 distinct output sets per document) in a pure LLM baseline. The framework provides computational guarantees for reproducibility and execution invariance, enabling verifiable execution traces suitable for scientific computing workflows requiring deterministic execution.

The computational cost analysis on 115 documents demonstrates a clear tradeoff: deterministic execution provides reproducibility guarantees at predictable $O(n)$ complexity with core deterministic rule-evaluation engine time (0.005s per document), while stochastic execution provides greater expressiveness at the cost of reproducibility and slower execution (7.3s per document).

The camera-ready revision additionally includes: (i) structured baseline comparison showing JSON schema-enforced baselines achieve 0% determinism while our system achieves 100%; (ii) measured rule-count scaling with linear regression proof ($R^2 = 0.9805$) confirming $O(n)$ complexity; (iii) concurrency stress test showing bitwise-stable outputs under 256 concurrent tasks with linear memory scaling; (iv) explanation-layer temperature study showing explanation drift without change to symbolic rule firings; (v) generalization discussion for log analysis, code analysis, and scientific data pipelines; and (vi) security boundary proofs demonstrating prompt injection immunity and read-only interface guarantees.

Acknowledgments

The author acknowledges the use of a large language model for explanatory output within the experimental system.

Disclosure of Interests

The author has no competing interests to declare that are relevant to the content of this article. A large language model was used exclusively for generating natural

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38

language explanations of pre-computed deterministic outputs. The stochastic component did not participate in state transitions, classification decisions, or suppression logic. All deterministic execution logic remained deterministic.

References

1. Brown, T., et al.: Language models are few-shot learners. In: Advances in Neural Information Processing Systems, vol. **33**, pp. 1877–1901 (2020)
2. Bommasani, R., et al.: On the opportunities and risks of foundation models. Tech. Rep., Stanford Center for Research on Foundation Models (2021)
3. IEEE: Ethically aligned design: A vision for prioritizing human well-being with autonomous and intelligent systems. IEEE (2019)
4. Li, J., Cheng, X., Zhao, W., Nie, Y., Wen, J.R.: HaluEval: A large-scale hallucination evaluation benchmark for large language models. arXiv preprint arXiv:2305.11747 (2023)
5. d'Avila Garcez, A., Gori, M., Lamb, L.C., Serafini, L.: Neuro-symbolic artificial intelligence: The state of the art. Artif. Intell. **273**, 1–38 (2019)
6. Chalkidis, N., et al.: LexGLUE: A benchmark dataset for legal language understanding in English. In: Proc. 60th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 1238–1350 (2022)
7. Nazarenko, A., Wyner, A.: Legal NLP introduction. In: Proc. ACL Workshop on Natural Legal Language Processing. Association for Computational Linguistics (2017)
8. Fensel, L., Kalf, Y., Simbeck, K.: Assessing the auditability of AI-integrating systems: A framework and learning analytics case study. arXiv preprint arXiv:2411.08906 (2024)
9. Gómez, A.P.: Rule-based expert systems for automated legal reasoning and contract analysis: A case study in knowledge representation. In: Advances in Computational Systems, Algorithms and Applications (2022)
10. Cai, X.H., Advani, H.H., Cai, J.: Ontology and rule-based natural language processing approach for interpreting textual regulations on underground utility infrastructure. Adv. Eng. Inform. **47**, 101248 (2021)
11. Scholak, T., Schucher, R., Bahdanau, D.: PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In: Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 9895–9901 (2021)

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29915-4_38