Accurate Legal Reasoning at Scale: Neuro-Symbolic Offloading and Structural Auditability for Robust Legal Adjudication

Stanisław Sójka and Witold Kowalczyk
Delos AI Inc.
{stan, witold}@delosintelligence.com

Abstract

Legal texts often contain computational legal clauses—provisions whose understanding requires complex logic. While frontier Large Reasoning Models (LRMs) can describe such clauses, building production-ready systems is limited by reasoning errors and the high cost of inference. We propose Amortized Intelligence, a neuro-symbolic approach where we use an LLM once to translate a legal text into Deterministic Autonomous Contract Language (DACL): a typed graph intermediate representation. Adjudication then relies on deterministic graph executions with a visually auditable trace. In comparison against runtime LRM baselines (including GPT-5.2 and Gemini 3 Pro), our DACL-based Agent achieves near-perfect consistency and mitigates the "reasoning cliff" observed in probabilistic models. The system reduces compute costs by over 90% in high-volume workflows while satisfying the strict auditability requirements of legal adjudication.

1 Introduction

Legal texts—laws, regulations, and commercial contracts—are composed of complex clauses that define rights and obligations. Often, these provisions exceed the semantic complexity of the real-world activities they govern. Correctly interpreting them requires a synthesis of mathematical reasoning (e.g., tax formulas), logical condition evaluation (e.g., eligibility thresholds), and contextual language interpretation. We refer to such provisions as *Computational Legal Clauses*. They are widespread across high-stakes domains including civil procedure, insurance, taxation, and financial regulation.

The landscape of legal AI has expanded rapidly from discriminative tasks like judgment prediction (Aletras et al., 2016; Katz et al., 2017; Chalkidis et al., 2019, 2022; T.y.s.s et al., 2023) and automated contract review (Hendrycks et al., 2021; Leivaditi et al., 2020) to generative applications powered by Large Language Models (LLMs)(Hou et al., 2025).

However, while specialized models demonstrate impressive semantic fluency (Huang et al., 2023; Cui et al., 2023), they often lack the structural rigor required for statutory reasoning (Blair-Stanek et al., 2023). Recent benchmarking (Guha et al., 2023; Fan et al., 2026) reveals that even frontier models struggle with structural rule application, while specialized evaluations like LexNum (Zhang et al., 2025a) highlight critical failures in 'legal numeracy'—the precise intersection of procedural logic and arithmetic (Chen et al., 2022; Gao et al., 2023).

In production-ready adjudication, this probabilistic nature creates two prohibitive barriers: reliability and economics. First, LLMs can fail silently on arithmetic—hallucinating pricing coefficients or dropping nested conditions, a phenomenon often linked to the unfaithfulness of chain-of-thought reasoning (Turpin et al., 2023). In high-stakes contracts, a calculation cannot yield non-deterministic outputs across identical inputs. Second, relying on Chain-of-Thought (CoT) inference for every recurring transaction (e.g., processing thousands of monthly invoices) forces a linear cost scaling (Hsieh et al., 2023; Chen et al., 2023). This makes autonomous adjudication commercially unattractive for high-volume settings.

Foundational rigor in computational law has traditionally relied on logic programming, where Prolog (Körner et al., 2022) and its extensions—such as ProbLog (De Raedt et al., 2007), EPILOG (Porto, 1984), and Blawx (Morris, 2023)—enable defeasible reasoning (Antoniou and Bikakis, 2007) but suffer from a manual knowledge acquisition bottleneck. Recent neuro-symbolic research addresses this by employing LLMs to bridge unstructured text and structured logic. Systems like SOLAR (Sadowski and Chudziak, 2025b) and ProSLM (Vakharia et al., 2024) decompose reason-

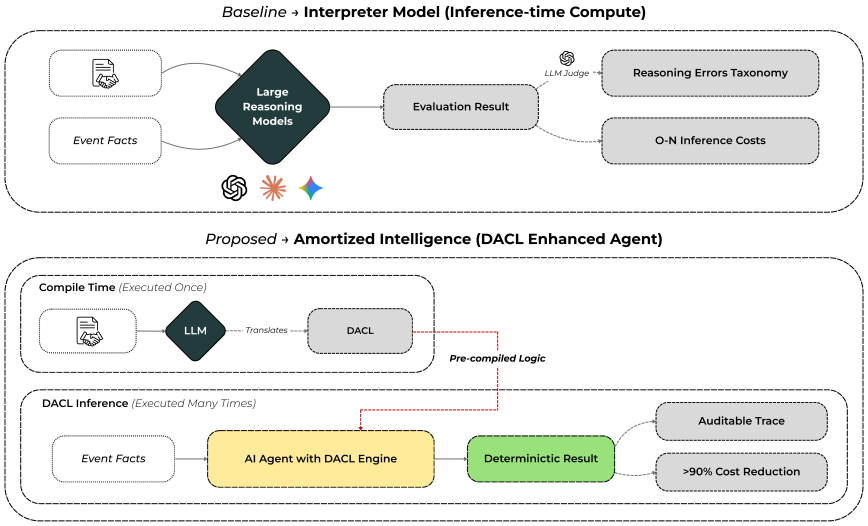

Figure 1: **Architectural Comparison: Baseline vs. Amortized Intelligence.** The baseline (top) relies on expensive, probabilistic *inference-time compute* for every transaction, leading to linear cost scaling and potential hallucinations. Our proposed approach (bottom) shifts reasoning to a one-time *compile-time* step, translating the contract into a DACL graph. This enables the runtime agent to execute deterministic logic with an auditable trace.

ing into extraction and application stages, while Kant et al. (2024, 2025) demonstrate effective policy-to-Prolog translation, with the latter introducing a guided encoding framework for improved adjudication accuracy. In commercial contexts, focus has shifted to automating semantic graph construction via reinforcement learning (Dechtiar et al., 2025b) and ensuring structural validity through uncertainty quantification (Dechtiar et al., 2025a) and SMT-verified logical consistency (Sadowski and Chudziak, 2025a).

While these systems advance the state of the art, they often remain fragmented—focusing on static graph extraction or theoretical simulations rather than the high-volume transaction processing required by industry. Many approaches simplify the legal problem space, omitting computationally complex components like real date arithmetic or full rule logic, and fail to address practical constraints on inference latency and cost.

To resolve these trade-offs, we introduce *Amortized Intelligence*: a neuro-symbolic architecture that shifts the paradigm from a *run-time interpreter* to a *logic compiler*. As illustrated in Figure 1, we use an LLM only once to translate a contract into DACL, a deterministic intermediate representation. Subsequent adjudications reduce to efficient graph execution. Unlike general-purpose logic programming (e.g., Prolog) optimized for theorem proving, DACL is engineered specifically for commercial logic, employing a typed abstract syntax tree with primitives for pricing formulas, dates, and procedural clauses. Compilation also makes interpretive choices on quantifiable provisions explicit and stable across transactions, so that identical inputs produce identical outputs by construction. We evaluate this design across four business domains, four clause primitives, and four LRM configurations.

The contributions of this paper are as follows:

- **DACL Architecture:** We introduce a graph-based neuro-symbolic intermediate representation optimized for commercial logic. DACL prioritizes legal primitives and explicit variable-dependency tracking over general theorem proving.
- **Amortized Intelligence:** We validate our approach on data sampled from industrial Enterprise Resource Planning/accounting systems and demonstrate the advantage of computational off-loading. By handling heavy legal interpretation once and subsequently running agentic inference on the resulting DACL graph, we achieve 99.5% accuracy on all tasks and $> 95\%$ cost reduction for high-volume tasks over the closest performing LRM setup.
- **Taxonomy of Failure & Structural Auditability:** We analyze the limitations of frontier Large Reasoning Models in legal arithmetic, isolating error modes. DACL provides a deterministic, traceable execution path for

every decision, satisfying explainability requirements essential for legal compliance.

- **Production Deployment:** The DACL Agent has been deployed in a live production environment for over 12 months, where it autonomously executes billing and accounts payable workflows across 150+ commercial agreements and approximately 1,000 monthly billing events. The system operates continuously under real-world constraints and enterprise compliance requirements, replacing manual billing processes and delivering sustained, measurable cost reductions.

2 Method

We evaluate two distinct architectures for automating industrial contract execution: a direct *Probabilistic Interpreter* (baseline) and a *Neuro-Symbolic Orchestrator* (DACL). The baseline represents the prevailing paradigm where a general-purpose frontier model reasons over the full contract text at runtime. In contrast, our proposed approach decouples reasoning from execution: the contract is compiled into a deterministic intermediate representation (DACL), which is then executed by a lightweight agent equipped with a symbolic engine.

2.1 Baseline: Probabilistic Interpretation

The baseline system operates as a stateless interpreter. For every input event e_i, the system constructs a prompt containing the full natural language text of the relevant contract C, the specific runtime facts F_i (e.g., shipment weight, transaction date), and the query Q. This approach incurs $O(N)$ inference costs, as the model must re-read and re-interpret the contract for every transaction.

To ensure a rigorous comparison against state-of-the-art capabilities, we benchmarked against three frontier model families (GPT-5.2, Claude 4.5 Sonnet, and Gemini 3 Pro), explicitly enabling their respective extended reasoning modes where available. Full configuration details are provided in Appendix A.

2.2 Symbolic Execution Engine (DACL)

The core of our proposed architecture is the DACL, a domain-specific intermediate representation that models contract logic as a directed acyclic graph (DAG). Unlike probabilistic LLM-based approaches that suffer from semantic drift across multi-step calculations, DACL guarantees *referential transparency*—the same input always produces the same output—enabling formal verification of contract computations. The translation of natural language contracts into this graph structure follows a rigorous five-stage pipeline involving agentic segmentation, classification, and verified generation (see Appendix D for full implementation details).

2.2.1 Clause Primitive Hierarchy

DACL utilizes a strongly-typed variable schema to ensure referential transparency. It supports four recursive clause primitives, each targeting a distinct pattern in contract logic. Detailed formal definitions and validation rules for these primitives are provided in Appendix E, together with the engine's specifications for temporal clause versioning, structural audit trail generation, and programmatic error recovery.

1. **Procedure**: Models sequential workflows where step t consumes outputs from $t-1$. Procedures support *hybrid clause patterns* with early termination: when a conditional step resolves to a terminal value, subsequent steps are skipped, optimizing execution time for complex fallback logic.

2. **Logical Clause**: Implements first-order propositional logic over contract variables. The engine evaluates conditions in declaration order, returning on the first match. This strictly enforces priority-based eligibility rules.

3. **Range Clause**: Maps continuous variables to discrete brackets with explicit boundary semantics. By enforcing strict, non-overlapping interval validation at load time, this primitive eliminates the "off-by-one" failures.

4. **Pricing Formula**: Executes arithmetic expressions with configurable decimal precision. Sub-formulas are evaluated sequentially in a sandboxed environment that blocks arbitrary code execution while supporting standard mathematical operations via variable interpolation.

Amendments and temporal versioning. DACL supports clause-level recompilation: when a contract is amended, only the affected clauses are re-translated. Each clause carries `validity_start_date` and `validity_end_date`

annotations (Appendix B.3.2), and the engine selects the version in force on the event date. This prevents retroactive rating errors without requiring full contract regeneration.

2.3 Neuro-Symbolic Agent

The runtime orchestrator is an autonomous agent built using the **OpenAI Agents SDK** (OpenAI, 2025c), following (Yao et al., 2022; Schick et al., 2023) to interleave reasoning traces with symbolic tool execution. Unlike the baseline, which relies on the LLM for arithmetic and logic, this agent uses the LLM solely for semantic routing and result synthesis.

To demonstrate the efficiency of this architecture, the agent utilizes `gpt-5-mini` (OpenAI, 2025a), a lightweight model optimized for low-latency tool use. The agent is provided with exactly one tool:

> `evaluate_clauses_tool(`K`,` F_i`)`: Invokes the symbolic engine to execute the logic blocks identified by K against the current event data F_i.

Formally, for a natural language query Q and runtime facts F_i, the agent approximates the adjudication function $\mathcal{A}(Q, F_i)$ via the following three-stage pipeline:

1. **Semantic Mapping**: Leveraging techniques from constrained semantic parsing (Shin et al., 2021; Geng et al., 2023), the agent first maps the unstructured query and context to a precise subset of contract clause identifiers $K \subset \mathcal{C}_{ids}$. This step isolates the relevant logic without executing it:

$$K = \mathcal{M}_{\theta_{small}}(Q, F_i) \tag{1}$$

2. **Symbolic Delegation**: The agent invokes the tool to trigger the deterministic engine Φ_{DACL}. This step executes the logic within the sandbox, returning both a computed value v and a structural audit trace τ:

$$(v, \tau) = \Phi_{DACL}(K, F_i) \tag{2}$$

   Critically, Φ_{DACL} is non-probabilistic ($P(v|K, F_i) = 1$), ensuring mathematical precision and strict adherence to contract boundaries.

3. **Synthesis**: Finally, the tool output is injected back into the context. The agent synthesizes the final natural language response y, grounding its explanation in the generated trace τ:

$$y = \mathcal{S}_{\theta_{small}}(Q, v, \tau) \tag{3}$$

This design ensures that all business-critical logic remains within the verified, deterministic boundary of the DACL engine (Φ_{DACL}), while the `gpt-5-mini` model handles the flexible user interaction at a fraction of the computational cost of the baseline models.

3 Results and Evaluation

We evaluate our neuro-symbolic architecture against state-of-the-art frontier models on four proprietary industrial agreements. Our evaluation focuses on three key dimensions: (1) accuracy and consistency across test samples, (2) error taxonomy analysis, and (3) computational efficiency.

3.1 Dataset and Experimental Setup

The dataset consists of four real-world commercial agreements spanning healthcare, energy, and transportation. To preserve commercial confidentiality, we refer to them by their domain codes: **Health-PPO**, **Energy-Sup**, **Logistics-MSA**, and **Muni-IFB**. Table 1 summarizes their domain, business criticality, and logical complexity.

Contract Representation. We process raw OCR text for all agreements. The DACL pipeline translates these into a compact taxonomy of clause types: *procedure* (sequential steps), *logical_clause* (branching if-then-else), and *range_clause* (continuous-to-discrete mapping). Appendix C details the structural complexity of each agreement. Notably, **Logistics-MSA** presents a "Very High" difficulty profile with 76 distinct decision states (logic branches and pricing brackets).

Evaluation Pipeline. Our evaluation simulates a production environment using a three-stage pipeline:

1. **Traffic Simulation**: We use a specialized Event Generator to create synthetic transaction data. The generator is seeded with scenarios drawn from live production traffic on the same four agreements, and parameters are varied within contract-valid ranges to produce 400 events (100 per agreement), covering every decision branch and pricing bracket.

ID	Domain	Business Criticality	Logic Complexity
Health-PPO	Healthcare	Compliance & liability; coverage determination and cost-sharing.	**High:** Nested boolean logic, multiple eligibility checks (age, relationship), and conditional copay waivers.
Energy-Sup	Energy	Procurement efficiency; calculation of price based on dynamic weight and market indices.	**Medium:** Arithmetic heavy formulas requiring precise unit conversions.
Logistics-MSA	Transport	Revenue assurance; calculation of complex transport rates and diesel surcharges.	**Very High:** Multi-branch decision trees (Day Rate vs. Mileage) with surcharge lookups.
Muni-IFB	Municipal	Contract management; automated invoice verification with time-varying fee schedules.	**Medium:** Date-based period lookups combined with index-based pricing formulas.

Table 1: Industrial agreements used in evaluation. Complexity ranges from linear arithmetic to massive combinatorial decision trees.

2. **Deterministic Ground Truth**: Ground truth is produced by a "Gold Standard" engine implementing the contract logic in deterministic code. The Gold Standard and the DACL graph were developed independently. Legal interpretation was provided by qualified lawyers and domain experts, and the resulting logic was then re-implemented independently of the DACL graph. This guarantees mathematical precision (e.g., strict rounding) and reproducibility.

3. **LLM-as-Judge:** Following recent protocols for scalable evaluation (Zheng et al., 2023), an independent model evaluates concordance between the system output and the Gold Standard.

3.2 Operational Accuracy and the Reasoning Cliff

We evaluated system performance on a dataset of 400 randomized real-world events, simulating standard industrial traffic distributions. As shown in Table 2, the **DACL Agent** achieved near-perfect reliability (99.5% overall accuracy), effectively mitigating the stochastic risks inherent in pure LLM deployments.

The Complexity Invariance of DACL. Crucially, the DACL Agent demonstrated *complexity invariance*: its performance remained stable (>98%) regardless of whether the domain required simple arithmetic (*Energy-Sup*, 1 state) or massive decision trees (*Logistics-MSA*, 76 states). In contrast, frontier LRMs exhibited a sharp "complexity cliff." While GPT-5.2 and Claude Sonnet 4.5 achieved parity with DACL on the arithmetic-focused *Energy-Sup* (>99%), they suffered catastrophic degradation on the *Logistics-MSA*.

The Limits of Probabilistic Reasoning. The *Logistics-MSA* domain serves as a stress test for the "reasoning as compute" hypothesis. Characterized by 76 distinct decision states and conditional surcharge tables, this domain overwhelmed the attention mechanisms of standard models. Notably, while the "Medium" reasoning effort improved GPT-5.2's performance on the temporal logic of *Muni-IFB* dramatically (raising accuracy from 36% to 95%), it failed to rescue the model in the high-branching Logistics domain. This indicates that while Chain-of-Thought can effectively solve *depth* (e.g., date lookups in Muni-IFB), it struggles with *breadth* (e.g., maintaining state across the 28-branch decision tree of Logistics-MSA).

Agreement *(Domain)*	**GPT-5.2** *(None)*	**GPT-5.2** *(Med)*	**Claude** *Sonnet*	**Gemini** *3 Pro*	**DACL Agent**
Health-PPO	74%	<u>91%</u>	73%	69%	**100%**
Energy-Sup	<u>100%</u>	99%	<u>100%</u>	91%	**100%**
Logistics-MSA	22%	<u>46%</u>	45%	30%	**98%**
Muni-IFB	36%	95%	93%	<u>96%</u>	**100%**
Overall	58.0%	<u>82.8%</u>	77.8%	71.5%	**99.5%**

Table 2: Accuracy on Test Events (n=400). **Bold** indicates best overall; <u>underlining</u> indicates best baseline.

3.3 Anatomy of Logic Failures

To determine the root cause of the baseline failures, we employed a "Judge LLM" (GPT-5.1) to classify errors into three distinct operational categories: *Variable Dependency* (VD), *Distraction Hallucination* (DH), and *Arithmetic Hallucination* (AH). Agreement-specific process guidance ensures the judge verifies not only outcome equivalence but adherence to the contractual calculation sequence.

Figure 2 reveals a critical architectural insight: the primary failure mode of frontier models is **Variable Dependency (VD)**, which accounted for 71% of all recorded errors across the four evaluated models. Conversely, **Arithmetic Hallucination (AH)** was statistically negligible (< 1 error per model). This presents a paradoxical finding: frontier mod-

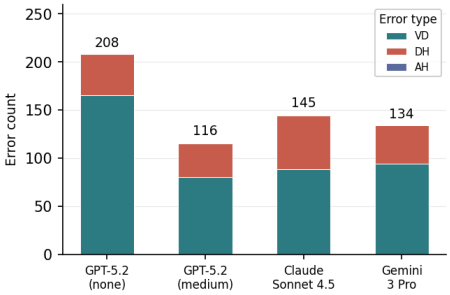

Figure 2: **Error Taxonomy by Model Configuration.** The stacked bars show the total failure count for each model ($N = 400$ events). **Arithmetic (AH)** errors remain negligible across all models, confirming that the difficulty lies in state tracking rather than computation.

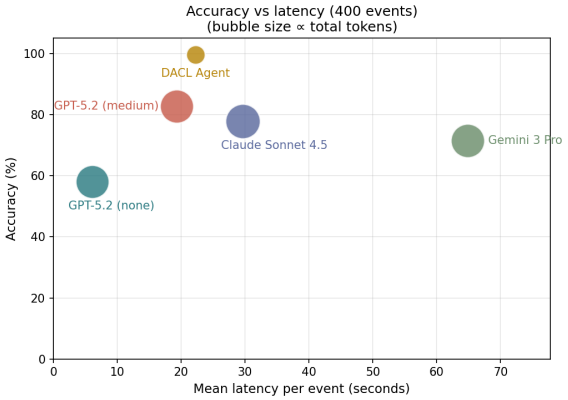

Figure 3: **Pareto Efficiency Analysis (Accuracy vs. Latency).** Bubble size is proportional to total token consumption. The DACL Agent (gold) occupies the optimal quadrant, achieving near-perfect accuracy (≈99.5%) with moderate latency.

els have mastered the *computational* primitives of law (arithmetic) but lack the *structural* fidelity to apply them to the correct variables.

In complex environments like the *Logistics-MSA* (characterized by 76 distinct decision states), models frequently calculated the math correctly but conditioned it on the wrong state - for instance, applying a "Tier A" rate to a "Tier B" shipment weight. This indicates that the difficulty of legal adjudication lies not in the complexity of the equations, but in the *state tracking* required to navigate nested boolean logic.

DACL errors. The two DACL errors on Logistics-MSA both originated in the orchestrator (`gpt-5-mini`), not in the symbolic engine, which is deterministic by construction. One was a semantic routing error where the agent selected the clause for a query that should have resolved against a different clause; the other was a tool-calling error where the agent invoked clause evaluation tool with an incomplete fact dictionary. Both classes are addressable with stricter tool-input schemas.

3.4 Computational Economics and Latency Profile

For industrial applications, the viability of an architecture is determined by its token economics and latency profile (Chen et al., 2023). We analyzed the resource consumption for processing the full 400-event evaluation set.

Token Economy. The DACL architecture demonstrates a structural efficiency advantage. By retrieving only the active logic clauses rather than the full contract context, DACL reduced total token consumption by a factor of **9.9x** compared to the GPT-5.2 Medium baseline (1.35M vs. 13.44M tokens).

Latency vs. Reliability. As illustrated in Figure 3, while the standard GPT-5.2 (No Reasoning) offered the lowest latency (∼5–8s), its poor accuracy (58%) renders it unsuitable for production. Crucially, the ability to offload complex reasoning to the deterministic DACL engine enabled the use of a lower-latency LLM (`gpt-5-mini`) for orchestration, avoiding the need for the heavy-duty reasoning models required by the baseline. Among the viable high-accuracy configurations, DACL provided superior throughput. On the complex *Logistics-MSA* task, the DACL Agent averaged **26.8 seconds** per event, significantly faster than GPT-5.2 Medium, Claude Sonnet 4.5 and Gemini Pro (∼164 s). This result suggests that shifting the "reasoning" burden from the LLM to a symbolic engine improves both speed and correctness simultaneously, avoiding the diminishing returns of extended inference observed in hierarchical legal reasoning (Zhang et al., 2025b).

4 Conclusion

Current legal AI systems predominantly function as assistants, requiring a human expert to review and validate every output. While this “human-in-the-loop” paradigm is useful for individual practitioners, it does not offer a credible path to scalability or for tackling the systemic issues surrounding access to justice. Legal aid systems, courts, and public agencies must process millions of cases annually—a volume that renders manual review mathemati-

cally impossible.

To meet this industrial challenge, we cannot rely on probabilistic models alone. High-stakes adjudication requires a system that is fully automated, operates without human supervision, and remains consistent, explainable, and affordable at scale. Our work on *Amortized Intelligence* demonstrates that achieving these requirements is feasible through a neuro-symbolic architecture. By treating the LLM as a one-time semantic compiler rather than a runtime interpreter, DACL eliminates the stochastic risks of "System 2" reasoning (OpenAI, 2025b) reducing the marginal cost of adjudication to levels that make mass automation viable.

The implications of this architecture extend well beyond commercial contracts. The same deterministic approach demonstrated for Logistics and Healthcare agreements could be applied to other complex regulatory domains, such as civil procedure rules, tax code interpretation, city council bylaws, and social insurance claims. Extending the compiler target to defeasible and probabilistic formalisms is a natural next step, for which DACL's structured audit trail provides the interpretability substrate.

Building such systems offers more than just efficiency; it offers stability, uniformity of rules, and speed in their application. Ultimately, by decoupling legal reasoning from the cost and unpredictability of generative inference, we pave the way for a legal infrastructure that is not only intelligent but structurally auditable and accessible to all.

Limitations

While our neuro-symbolic architecture offers significant improvements in reliability and cost for industrial contract execution, several limitations remain.

First, the **expressivity of the intermediate representation** is currently bounded by the predefined primitives in DACL. Our system currently supports arithmetic, first-order logic, and range-based lookups, which cover the vast majority of commercial agreements. However, it does not yet support defeasible reasoning, open-textured legal standards (e.g., "reasonable care"), or higher-order logic often found in statutory interpretation or case law. Expanding the primitive library to cover these broader legal domains remains future work.

Second, the **reliance on a "Gold Standard" for evaluation** restricts the scale of our testing. To rigorously measure correctness, we had to manually implement the ground truth logic for every agreement in code. This high annotation cost limits our ability to evaluate the system on thousands of different contract types simultaneously, a challenge common to neuro-symbolic research where automated ground truth is unavailable.

The traffic used in evaluation is synthetically generated rather than sampled directly from production logs. Seeding from real scenarios gives strong coverage of valid input variations and normal operations, but it under-represents tail phenomena such as malformed inputs or rare multi-amendment sequences that only a full production-distribution study would surface.

Third, our experiments focused on **English-language commercial contracts** within specific domains (Logistics, Energy, Healthcare). We have not yet evaluated the performance of the semantic compiler on non-English jurisdictions or on informal, unstructured agreements that lack clear clause demarcations.

Finally, while the compilation step is a one-time cost, **compiler errors** can still occur. If the LLM misinterprets a clause during the initial translation to DACL, that error becomes deterministic and repeats for every subsequent transaction until corrected. Although our strict type system catches many such errors at compile time, human review of the generated graph is still recommended before deploying high-value contracts into production. We describe the pipeline's qualitative error profile in Appendix F.

Our LRM baselines use the full contract as context. A retrieval-augmented variant that feeds the model only the clauses relevant to a given event would be a stronger baseline. We expect it to narrow the gap on depth-heavy clauses but not on breadth-heavy ones, since Variable-Dependency errors arise from state-tracking across dozens of branching nodes rather than from context length. Quantifying this is left to future work.

References

Nikolaos Aletras, Dimitrios Tsarapatsanis, Daniel Preotiuc-Pietro, and Vasileios Lampos. 2016. Predicting judicial decisions of the european court of human rights: a natural language processing perspective. *PeerJ Comput. Sci.*, 2:e93.

Anthropic. 2025. Claude sonnet 4.5 system card. Tech-

nical report, Anthropic. Introducing Extended Thinking capabilities.

Grigoris Antoniou and Antonis Bikakis. 2007. Dr-prolog: A system for defeasible reasoning with rules and ontologies on the semantic web. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):233–245.

Andrew Blair-Stanek, Nils Holzenberger, and Benjamin Van Durme. 2023. Can gpt-3 perform statutory reasoning? In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, ICAIL '23, page 22–31, New York, NY, USA. Association for Computing Machinery.

Ilias Chalkidis, Ion Androutsopoulos, and Nikolaos Aletras. 2019. Neural legal judgment prediction in English. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4317–4323, Florence, Italy. Association for Computational Linguistics.

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, Dublin, Ireland. Association for Computational Linguistics.

Lingjiao Chen, Matei A. Zaharia, and James Y. Zou. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance. *ArXiv*, abs/2305.05176.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Trans. Mach. Learn. Res.*, 2023.

Jiaxi Cui, Zongjia Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. Chatlaw: A multi-agent collaborative legal assistant with knowledge graph enhanced mixture-of-experts large language model.

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. 2007. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI'07: Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2462–2467.

Moriya Dechtiar, Daniel Martin Katz, Sylvain Jaume, and Hongming Wang. 2025a. Llm as a judge for evaluating contract graphs: Multi-judge benchmarking and agentic uncertainty-aware refinement. Also available via SSRN: https://ssrn.com/abstract=5937996.

Moriya Dechtiar, Daniel Martin Katz, Mari Sundaresan, Sylvain Jaume, and Hongming Wang. 2025b. Graph-grpo-lex: Contract graph modeling and reinforcement learning with group relative policy optimization. *arXiv preprint arXiv:2511.06618*.

Google DeepMind. 2025. Gemini 3 pro model card. Technical report, Google. Version: gemini-3-pro-preview.

Yu Fan, Jingwei Ni, Jakob Merane, Yang Tian, Yoan Hermstrüwer, Yinya Huang, Mubashara Akhtar, Etienne Salimbeni, Florian Geering, Oliver Dreyer, Daniel Brunner, Markus Leippold, Mrinmaya Sachan, Alexander Stremitzer, Christoph Engel, Elliott Ash, and Joel Niklaus. 2026. Lexam: Benchmarking legal reasoning on 340 law exams. *Preprint*, arXiv:2505.12864.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Saibo Geng, Martin Josifosky, Maxime Peyrard, and Robert West. 2023. Grammar-constrained decoding for structured nlp tasks without finetuning. In *Conference on Empirical Methods in Natural Language Processing*.

Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya Narayana, Alex Chohlas-Wood, Austin M. K. Peters, Brandon Waldon, Daniel N. Rockmore, Diego A. Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, and 21 others. 2023. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. *ArXiv*, abs/2308.11462.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. Cuad: An expert-annotated nlp dataset for legal contract review. *Preprint*, arXiv:2103.06268.

Abe Bohan Hou, Orion Weller, Guanghui Qin, Eugene Yang, Dawn Lawrie, Nils Holzenberger, Andrew Blair-Stanek, and Benjamin Van Durme. 2025. CLERC: A dataset for U. S. legal case retrieval and retrieval-augmented analysis generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7913–7928, Albuquerque, New Mexico. Association for Computational Linguistics.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, Toronto, Canada. Association for Computational Linguistics.

Quzhe Huang, Mingxu Tao, Zhenwei An, Chen Zhang, Cong Jiang, Zhibin Chen, Zirui Wu, and Yansong Feng. 2023. Lawyer llama technical report. *CoRR*, abs/2305.15062.

Manuj Kant, Manav Kant, Marzieh Nabi, Preston Carlson, and Megan Ma. 2024. Equitable access to justice: Logical llms show promise. *arXiv preprint arXiv:2410.09904*.

Manuj Kant, Sareh Nabi, Manav Kant, Roland Scharrer, Megan Ma, and Marzieh Nabi. 2025. Towards robust legal reasoning: Harnessing logical llms in law. *Preprint*, arXiv:2502.17638.

Daniel Martin Katz, Michael J Bommarito, and Josh Blackman. 2017. A general approach for predicting the behavior of the supreme court of the united states. *PloS one*, 12(4):e0174698.

Philipp Körner, Michael Leuschel, João Barbosa, Vítor Santos Costa, Verónica Dahl, Manuel V. Hermenegildo, Jose F. Morales, Jan Wielemaker, Daniel Diaz, Salvador Abreu, and Giovanni Ciatto. 2022. Fifty years of prolog and beyond. *Theory and Practice of Logic Programming*, 22(6):776–858.

Spyretta Leivaditi, Julien Rossi, and E. Kanoulas. 2020. A benchmark for lease contract review. *ArXiv*, abs/2010.10386.

Jason Morris. 2023. Building blawx. In *3rd International Workshop on Goal-Directed Execution of Answer Set Programs (GDE 2023)*.

OpenAI. 2025a. Gpt-5 mini: Lightweight reasoning for agentic workflows. Model version: gpt-5-mini-2025-12-11.

OpenAI. 2025b. Gpt-5.2 system card. Technical report, OpenAI. Model version: gpt-5.2-2025-12-11.

OpenAI. 2025c. Openai agents sdk: A framework for orchestrating intelligent workflows. `https://github.com/openai/openai-agents-python`. Software library.

António Porto. 1984. Epilog: A language for extended programming in logic. In *Implementations of Prolog*, pages 268–278.

Albert Sadowski and Jarosław A. Chudziak. 2025a. Explainable rule application via structured prompting: A neural-symbolic approach. *arXiv preprint arXiv:2506.16335*.

Albert Sadowski and Jarosław A. Chudziak. 2025b. On verifiable legal reasoning: A multi-agent framework with formalized knowledge representations. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM)*, pages —. Also available on arXiv:2509.00710.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessí, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: language models can teach themselves to use tools. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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 *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Santosh T.y.s.s, Oana Ichim, and Matthias Grabmair. 2023. Zero-shot transfer of article-aware legal outcome classification for European court of human rights cases. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 605–617, Dubrovnik, Croatia. Association for Computational Linguistics.

Priyesh Vakharia, Abigail Kufeldt, Max Meyers, Ian Lane, and Leilani Gilpin. 2024. Proslm: A prolog synergized language model for explainable domain specific knowledge based question answering. *arXiv preprint arXiv:2409.11589*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *ArXiv*, abs/2210.03629.

Kepu Zhang, Guofu Xie, Weijie Yu, Mingyue Xu, Xu Tang, Yaxin Li, and Jun Xu. 2025a. Legal Mathematical Reasoning with LLMs: Procedural Alignment through Two-Stage Reinforcement Learning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 1586–1602, Singapore. Association for Computational Linguistics.

Li Zhang, Matthias Grabmair, Morgan A. Gray, and Kevin Ashley. 2025b. Thinking longer, not always smarter: Evaluating llm capabilities in hierarchical legal reasoning. *ArXiv*, abs/2510.08710.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

A LRM Experimental Settings

- OpenAI GPT-5.2 (`gpt-5.2-2025-12-11`) (OpenAI, 2025b): Configured in two variants with default reasoning_effort="none" and

extended `reasoning_effort="medium"` via the Responses API. This enables a balanced chain-of-thought depth suitable for industrial logic without excessive latency.

- Anthropic Claude 4.5 Sonnet (Anthropic, 2025): Configured with "Extended Thinking" enabled and a `max_tokens=8192`. This allows the model to generate internal reasoning traces before emitting the final structured output.
- Google Gemini 3 Pro (`gemini-3-pro-preview`) (DeepMind, 2025): Configured with the default `thinking_level="high"`, leveraging dynamic computation to traverse complex logical dependencies.

All baselines are constrained to output a strict schema containing a `reasoning` field (for auditability) and a `result` field (for automated evaluation).

B DACL Formal Specification

B.1 Logical Clause Semantics

Logical clauses represent branching conditions. Conditions are expressed in conjunctive normal form (CNF) with support for nested `AND`/`OR` blocks. The formal grammar for a condition ϕ is:

$$\phi ::= \texttt{lhs} \bowtie \texttt{rhs} \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \quad (4)$$

where $\bowtie \in \{=, \neq, <, \leq, >, \geq\}$.

B.2 Range Clause Validation

Range clauses map continuous inputs to discrete outputs. To ensure determinism, the DACL engine enforces the following constraints at load time:

- **Non-Overlap**: No two brackets may cover the same numerical range.
- **Strict Inclusive Boundaries**: Intervals are evaluated as $\texttt{min} \leq x \leq \texttt{max}$.
- **Exhaustive Coverage**: Warnings are issued if gaps exist between brackets without a defined default fall-back value.

B.3 Pricing Formula Sandbox

Pricing formulas allow for arithmetic operations. To maintain security and determinism, the evaluation context is sandboxed. Only the following mathematical functions are whitelisted:

- `ceil`, `floor`, `round`
- `sqrt`, `exp`, `log`

Arbitrary code execution (e.g., Python `eval`) is strictly blocked.

B.3.1 Type System and Variable Model

A DACL contract definition begins with a strongly-typed variable schema. Variables are categorized by *source*:

- **External**: Runtime inputs provided at evaluation time (e.g., shipment weight, service date)
- **Const**: Contract-specific constants with optional temporal validity windows
- **Derived**: Intermediate values computed during execution

The engine performs runtime type coercion with strict validation. The validation layer intercepts malformed inputs in production deployments before they propagate to calculation steps.

B.3.2 Temporal Validity and Amendment Support

DACL natively supports clause versioning through `validity_start_date` and `validity_end_date` annotations. When multiple clauses share an identifier, the engine selects the appropriate version based on an `evaluation_date` context parameter. The system enforces:

- Temporal adjacency: No gaps between validity periods for clauses with the same name
- Non-overlap: At most one clause may be active for any given date
- Open-ended restriction: Only the chronologically latest clause may omit an end date

This enables mid-contract rate amendments (common in logistics and energy agreements) without requiring full contract regeneration.

B.3.3 Execution Model and Audit Trail

Each evaluation produces a structured audit trail containing:

- **Variable states**: All input values with type validation results
- **Decision points**: For range/logical clauses, the specific condition or bracket that matched

ID	Primary Type	Clauses	States	Output
Energy-Sup	procedure	1	1	numeric
Logistics-MSA	logical + range	2	76	numeric
Muni-IFB	procedure	1	3	numeric
Health-PPO	logical	1	5	string

Table 3: Decision state complexity. *States* refers to the number of distinct logic branches (e.g., if/else paths) or pricing brackets

- **Formula breakdown**: For pricing formulas, each sub-calculation with intermediate values
- **Execution path**: The sequence of clause identifiers traversed

B.3.4 Error Handling and Recovery

Rather than failing silently, DACL generates actionable error messages that guide correction:

- Missing variables: "Variable `X` required for clause `Y`. Expected: `type - description`."
- Type mismatches: "Variable `X` must be a number, got string `'abc'`."
- Enum violations: "Variable `X` has invalid value `'foo'`. Valid values: `A`, `B`, `C`."

These messages are designed for programmatic consumption, enabling upstream systems (including LLM agents) to self-correct without human intervention.

C Decision state complexity

Table 3 specifies the difficulty level for each benchmark scenario based on the underlying logic required to resolve the clause. We quantify complexity primarily through the number of "States," which represents the count of distinct decision paths, condition combinations, or pricing tiers inherent in the source text. As the data indicates, the selected cases cover a broad spectrum of structural complexity, ranging from simple linear procedures (e.g., Energy-Sup) to high-dimensional logical ranges (e.g., Logistics-MSA with 76 distinct states).

D DACL Compilation Pipeline

The transformation of unstructured contract text into the deterministic DACL representation is achieved through a five-stage pipeline. This process ensures high fidelity between the source legal text and the resulting execution graph:

1. **Agentic Contract Segmentation:** The raw document is first processed by a segmentation agent that decomposes the text into discrete, semantically coherent blocks, preserving hierarchical context (e.g., section headers and definitions).
2. **Computability Classification:** Each segment is analyzed to determine if it contains computational logic (e.g., pricing formulas, boolean eligibility rules) or purely declarative prose. Only segments classified as *Computational Legal Clauses* proceed to the translation phase.
3. **Dynamic Few-Shot Generation:** We employ Anthropic Claude 4.5 Sonnet for the translation step. The model is prompted with domain-specific, dynamic few-shot examples—retrieved based on semantic similarity to the current clause—to generate the corresponding DACL primitive.
4. **Automated Scenario Testing:** The generated DACL graph undergoes immediate verification. An auxiliary agent generates synthetic test vectors (edge-case inputs) to execute the graph, flagging runtime errors or type inconsistencies.
5. **Expert Supervision:** Finally, the compiled logic is subject to human-in-the-loop review. Legal engineers validate the structural audit trail against the source text to certify production readiness.

E Data: Sample Legal Clauses and Sample Events

E.1 Sample Clauses Used

Below we present an overview of the legal clauses found in contracts that we use for testing the DACL system. All clauses are taken from existing, real-world contracts that the DACL is processing in production deployment.

E.1.1 Example Clause 1: Transportation Rate Clause in Logistics MSA

This example illustrates a representative pricing clause commonly found in transportation and logistics master service agreements.

> ***3.8 Pricing.*** *All pricing – including rates, charges (e.g., fuel surcharges), fees, or*

tariffs for the Vendor's services – shall be as set forth in the Pricing Schedule and shall remain firm for the term of this Agreement, unless the Parties mutually agree in writing to amend it. If amended in writing, the rates and fuel surcharges in effect on the date the Bill of Lading is issued shall govern. Under no circumstances may the Vendor "gross up" the agreed fee for any taxes, fees, licenses, or other charges; the Client need not pay any such amounts and may apply any payments made toward future invoices as a credit. No additional rates, charges, fees, or tariffs shall be imposed by the Vendor without the Client's prior written authorization. Any rate or charge for oversized shipments or services not covered by the Pricing Schedule must be documented in writing (e.g., email) between the Parties, and a copy of that special rate agreement shall accompany the Vendor's invoice as supporting documentation.

Pricing is structured across multiple service categories:

Base Linehaul Rates

- Tier A: Distance-based lift service for shipments up to 0.5 ton, charged at $X.XX per mile plus fuel surcharge (Exhibit A).
- Tier B: Distance-based lift service for shipments between 0.5 and 0.75 ton, charged at $X.XX per mile plus fuel surcharge (Exhibit A).

Local Tariffs

- Zone 1 (within 45 miles): Flat pickup and drop-off fees by equipment type (Exhibit B).
- Zone 2 (45–140 miles): Flat local service fees by equipment type (Exhibit C).

Ancillary Fees

- Backhaul discount: 50% reduction of the applicable linehaul rate (Exhibit D).
- Additional stops: $100 per extra pickup or delivery location.

Fuel Adjustment

- A fuel surcharge applies to all mileage-based charges.
- The surcharge calculation method is defined in Exhibit E and locks at the Bill of Lading issuance date.

E.1.2 Example Clause 2: Diesel Surcharge Estimation in Logistics MSA

A diesel surcharge (or fuel surcharge) is an additional fee applied to transportation costs to offset fluctuations in diesel prices. The surcharge is calculated using published diesel price indexes and adjusted regularly, ensuring equitable compensation for fuel-related cost changes during transit.

> *The applicable percentage fuel surcharge ("FSC") is tied to the U.S. DOE Gulf-Coast diesel index. Percentages are applied to all billable charges that consume fuel. Index values are reviewed weekly; the FSC in effect on the date of service applies.*

Fuel surcharge percentages are determined by price brackets. For example:

- Diesel prices between $2.00 and $2.099 correspond to a 1% surcharge.
- Prices between $4.10 and $4.199 correspond to a 22% surcharge.
- Prices between $4.50 and $4.599 correspond to a 27% surcharge.
- Prices between $6.60 and $6.699 correspond to a 48% surcharge.

Intermediate price ranges increase incrementally, typically by 1% for each $0.10 rise in fuel price.

> *Over $6.70 per gallon, an additional 1% surcharge applies for every $0.10 increment.*

E.1.3 Example Clause 3: Natural Gas (LNG) Price Calculation in Energy-Sup

Natural gas pricing relies on regional index values, conversion factors, and contract-specific adjustments derived from external data sources.

> ***Clause 3.3.1: Fuel Cost Determination***
>
> *All prices for fuel per unit shall be based solely on each month's respective regional gas index, with adjustments reflecting each month's index price.*

Clause 5.11: Pricing Methodology

The price per unit payable by the purchaser is calculated as:

$$PriceperUnit = \frac{RegionalGasIndex}{ConversionFactor} + Y + R + F$$

where:

- *Regional Gas Index: average natural gas price for the delivery month.*
- *Conversion Factor: units per MMBtu of natural gas.*
- *Y: supplier costs and profit per delivered unit.*
- *R: fixed rebate per unit representing purchaser credit participation.*
- *F: freight cost per unit based on standard delivery volumes.*

Attachment A: Price Sheet

Provides detailed values for variables Y, R, and F across the contract term.

E.2 Event Sample Data

The events evaluated in this study are grouped into domains corresponding to the various contracts we process, e.g. : Logistics Master Service Agreements (Logistics MSA), Energy Supply Agreements (Energy-Sup), and Healthcare Preferred Provider Organization plans (Health PPO). Each event represents a concise factual scenario requiring adjudication against a corresponding legal clause. Conceptually, each evaluation asks:

> *Based on the provided event information and governing agreement, determine the applicable outcome for the specified clause.*

For each contractual category, structured natural-language queries are generated to compute transportation rates, fuel surcharges, commodity pricing, or healthcare coverage determinations.

Logistics MSA Events

Logistics events describe shipment-specific operational facts used to evaluate transportation rate clauses and diesel fuel surcharge provisions.

> *A 0.6-ton shipment travels 348 miles from Hawkins to a regional terminal and is delivered on January 17, triggering Tier B linehaul pricing and a diesel surcharge based on the weekly Gulf-Coast index.*

> *A local pickup occurs within 40 miles of the origin facility with two additional delivery stops, requiring application of Zone 1 tariffs and ancillary stop fees.*

> *A return shipment qualifies for backhaul discount following completion of a long-haul delivery under the base rate schedule.*

Energy-Sup Events

Energy supply events capture commodity delivery scenarios requiring application of regional gas index pricing formulas.

> *An LNG shipment is delivered to an industrial customer, with fuel pricing calculated using the monthly Southern California gas index and contractual conversion factors.*

> *A bulk natural gas delivery incurs additional freight charges and supplier margin adjustments under the contract's pricing methodology.*

> *A scheduled energy shipment applies a fixed rebate per unit based on Attachment A pricing tables for the current contract term.*

Health PPO Events

Healthcare events represent emergency medical scenarios requiring coverage determination under PPO plan provisions.

> *A 15-year-old child on a family plan is transported by ground ambulance for an emergency condition, receives medically necessary care in the emergency department, and is admitted to the hospital.*

> *A 17-year-old dependent child on a family plan is airlifted to a hospital for emergency treatment and receives medically necessary services, despite the deductible not being met.*

A 15-year-old child on a family plan is transported by ambulance but later determined to have received routine follow-up care rather than active emergency treatment.

Together, these scenarios provide the factual foundation required to evaluate contractual outcomes across logistics pricing, energy commodity calculation, and healthcare coverage. Distance values inform linehaul rates, fuel indexes determine surcharge applicability, regional gas benchmarks drive energy pricing, and patient and transport attributes govern healthcare eligibility. This structured representation enables consistent evaluation across domains, allowing models to apply contractual rules directly to operational and clinical facts.

F Compilation Pipeline: Qualitative Observations

We report qualitative observations from operating the compilation pipeline across production agreements. We frame these as a limitation rather than a quantitative benchmark, since per-clause compilation outcomes are not instrumented for clean error-rate reporting.

Human review is mandatory. Production readiness in this setting does not require zero-error automation; it requires an auditable artefact that a legal engineer can sign off on. Step 5 of the pipeline enforces this review.

The type checker catches most structural errors. Range non-overlap violations, type mismatches in pricing formulas, and missing `validity_end_date` on non-terminal clause versions are caught at load time and surfaced to the compiler agent as actionable messages, enabling self-correction without human intervention.

Scenario testing catches most remaining semantic errors. Automated test-vector generation surfaces clauses whose runtime behaviour contradicts the synthetic cases derived from the clause text.