Know Your Limits : On the Faithfulness of LLMs as Solvers and Autoformalizers in Legal Reasoning

Olivia Peiyu Wang [1] **Sanna Wong-Toropainen** [2 3] **Daneshvar Amrollahi** [4] **Ryan Bai** [5] **Tashvi Bansal** [6] **Arush Garg** [7] **Leilani H. Gilpin** [1]

Abstract

Large Language Models (LLMs) achieve strong performance on reasoning tasks, but whether this reflects faithful logical inference or heuristic approximation remains unclear. We study this question in legal entailment by comparing three paradigms, including pure LLM classification, LLM-based Formal Reasoning, and solver-based Formal Reasoning using the Z3 SMT solver, on a re-annotated subset of ContractNLI across five LLMs. Our re-annotation reveals a systematic and measurable gap between pragmatic legal interpretation and strict formal entailment, where a substantial proportion of legally sound inferences are not formally grounded without additional unstated assumptions. While introducing formal structure improves accuracy, with LLM-based Formal Reasoning achieving the highest benchmark performance, we show that this gain does not imply faithful reasoning. We identify three recurring failure modes: *scope laundering*, where LLMs report solver-inconsistent classifications without executing the underlying formal reasoning, producing conclusions that appear logically grounded but are not; *implicit constraint blindness*, where LLMs overlook logical constraints present in formal representations; and *program synthesis failures*, where LLMs generate incorrect Z3 code despite structured prompting. Critically, scope laundering persists across all models, raising serious concerns about the faithfulness of LLM-based formal reasoning as a proxy for symbolic execution. These results reveal a fundamental gap between benchmark accuracy and logical faithfulness.

[1]University of California, Santa Cruz, Santa Cruz, California, USA [2]University of Helsinki, Helsinki, Finland [3]CodeX, Stanford, California, USA [4]Stanford University, Stanford, California, USA [5]Canyon Crest Academy, San Diego, California, USA [6]Monta Vista High School, Cupertino, California, USA [7]Los Altos High School, Los Altos, California, USA. Correspondence to: Olivia Peiyu Wang <pwang95@ucsc.edu>, Daneshvar Amrollahi <daneshvar@cs.stanford.edu>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

1. Introduction

The adoption of Large Language Models (LLMs) in high-stakes domains such as law is constrained by the need for faithful and verifiable reasoning. While techniques such as Chain-of-Thought improve performance (Wei et al., 2022; Chen et al., 2025), LLMs remain prone to hallucination, fabricated citations, and unverifiable inference. Some failures are unacceptable in legal settings (Dahl et al., 2024). Beyond hallucination, correct conclusions often depend on implicit assumptions that are not explicitly stated, making it unclear which inferences are permissible. This ambiguity contributes to the significant verification overhead required for legal deployment (Dixon Jr, 2025).

Neuro-symbolic approaches address this by combining LLM-based autoformalization with symbolic reasoning, translating natural language into formal representations evaluated under strict semantics. However, this exposes a key limitation: formal reasoning requires explicit assumptions, while legal text is inherently underspecified.

In this work, we compare three paradigms: pure LLM classification, LLM-based formal reasoning, and LLM-driven autoformalization with SMT solving. We evaluate these methods on a modified ContractNLI dataset (Koreeda & Manning, 2021) of real-world legal contracts, across five LLMs: GPT-OSS-120B(GPT) (Agarwal et al., 2025), Claude-Sonnet-4-6(Claude) (Anthropic, 2026), Meta-Llama-3.1-8B(Llama) (Meta, 2024), DeepSeek-V3.2(Deepseek) (Liu et al., 2025), and Qwen2.5-72B(Qwen) (Yang et al., 2024).

We find that adding structure improves accuracy, with LLM-based formal reasoning performing the best. However, this does not imply faithful reasoning: LLMs misinterpret formal representations and introduce unstated assumptions, while SMT-based reasoning remains conservative by enforcing strict logical validity. Even with explicit prompting,

autoformalization fails to recover the assumptions required for correct inference.

These results reveal a fundamental gap between benchmark accuracy and logical faithfulness, identifying autoformalization as the primary bottleneck in neuro-symbolic systems.

Our contributions are as follows:

- We show that adding formal structure improves LLM performance, with LLM-based Formal Reasoning outperforming both pure LLM and neuro-symbolic pipelines.
- We demonstrate that improved accuracy does not imply faithful reasoning, as LLMs misinterpret formal representations and rely on implicit assumptions without executing the formal logic for their classification results.
- We identify three major failure modes, including scope laundering, implicit constraint blindness, and program synthesis failures, surfacing the tension between accuracy and faithfulness.
- We identify assumption ambiguity as a core challenge, where the boundary between valid inference and unjustified assumptions is unclear.
- We highlight a gap between benchmark labels and formal logical semantics in real-world legal data.

2. Related Work

Neuro-symbolic Reasoning with Autoformalization and Symbolic Solving. A key limitation of pure LLM-based reasoning is the lack of faithfulness: chain-of-thought outputs often reflect heuristics rather than logical inference (Turpin et al., 2023; Chen et al., 2025). Structured LLM approaches, such as Selection-Inference (Creswell & Shanahan, 2022), impose reasoning structure but do not eliminate hallucination. Neuro-symbolic methods instead decompose reasoning into LLM-based autoformalization and symbolic execution, where the latter guarantees correctness with respect to its inputs (Han et al., 2022; Yang et al., 2025; Ryu et al., 2024; Yang et al., 2023). This shifts the challenge to representation quality. However, prior work shows that LLM-generated formalizations are often incomplete or inconsistent (Pan et al., 2023; Vakharia et al., 2024), raising concerns about whether implicit assumptions required for correct reasoning can be reliably captured. Another line of work explored the approach of utilizing LLM as the solver and showed that this approach outperformed others while reducing syntax errors (Li et al., 2024; Xu et al., 2024; Feng et al., 2024). However, they did not evaluate whether the performance boost and error reduction came at the cost of faithfulness to the results generated by the symbolic solver. Recent work detects such formalization drift via round-trip equivalence checking (Amrollahi et al., 2026), but does not study which unstated assumptions are justified. Our work assesses and compares the performance of three different modes, ranging from pure LLM classification, LLM-based formal reasoning, and LLM-driven autoformalization with Satisfiability Modulo Theories (SMT) solving, and shows that structures improve performance, but the performance boost may come at the cost of faithfulness.

Neuro-symbolic AI and Formal Reasoning in the Legal Domain. Applications of neuro-symbolic reasoning in law remain limited and often rely on simplified or manually annotated data (Holzenberger et al., 2020; Holzenberger & Van Durme, 2021; 2023). Existing approaches typically employ domain-specific languages such as Stipula (Hähnle et al., 2025) and Catala (Merigoux et al., 2021), which provide strong guarantees but are difficult to integrate with general-purpose solvers. More broadly, formal reasoning benchmarks such as FOLIO (Han et al., 2022) and ProofWriter (Tafjord et al., 2021) rely on synthetic or simplified language, limiting their applicability to real-world legal reasoning. When LLM-based approaches are used, they are typically evaluated on simplified inputs that do not reflect the ambiguity and underspecification of real contracts (Jurayj et al., 2026). A typical example from the simplified dataset looks like "All people who regularly drink coffee are dependent on caffeine." By contrast, even a short legal example reads: "Neither Party shall, without the written approval of the other Party, publish, copy, or use the Confidential Information for their sole benefit." Within the legal domain, datasets frequently simplify text or focus on narrow subfields, and rarely address the role of implicit assumptions in interpretation. ContractNLI (Koreeda & Manning, 2021) is a notable exception, grounding entailment tasks in authentic contract language. However, its labels reflect interpretation rather than strict logical entailment, motivating our re-annotation to align the dataset with formal reasoning. The US health insurance policy dataset from (Kant et al., 2025) is another dataset that is based on real insurance policies; however, it is not publicly available. Our work starts from the only publicly available real legal contract dataset - ContractNLI, and re-annotates the labels based on the logic definitions of "entailment", "contradiction", and "neutral". Through our experiments and qualitative analysis, we surface the gaps between legal interpretation and logic solving.

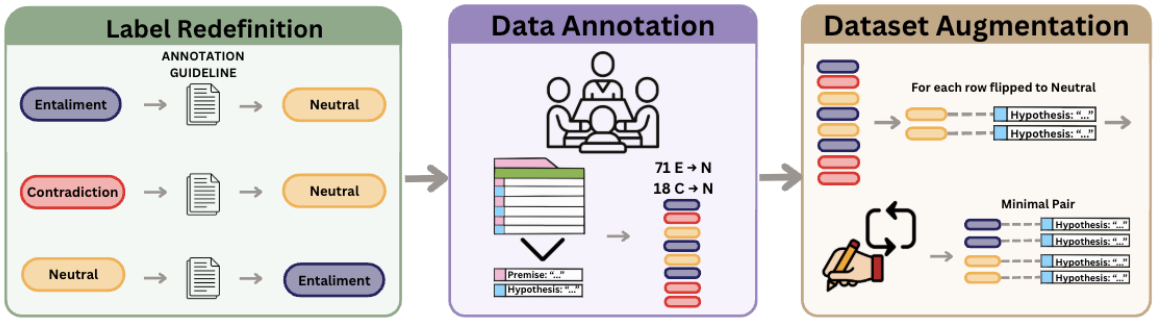

Figure 1. Visualization of the Dataset Construction Process

3. Methodology

3.1. Dataset Construction

To evaluate reasoning under realistic legal conditions, we build on the ContractNLI dataset (Koreeda & Manning, 2021), one of the few benchmarks derived from authentic contracts. However, its original labels reflect interpretation rather than strict logical entailment, making it unsuitable for formal reasoning without modification (Figure 1).

Label Redefinition. We adopt a formal notion of entailment aligned with SMT-based reasoning. A hypothesis H is **entailed** by a premise P if $P \wedge \neg H$ is unsatisfiable, and **contradicted** if $P \wedge H$ is unsatisfiable. All remaining cases are labeled **Neutral**.

Crucially, **Neutral** includes both (i) semantically irrelevant cases and (ii) cases where the premise is insufficient to support inference without additional assumptions.

Example (Entailment) : Under the premise *"Nothing in this Agreement is to be construed as granting the Recipient any right whatsoever with respect to the Confidential Information,"* the hypothesis *"Agreement shall not grant Receiving Party any right to Confidential Information"* is classified as **Entailment**, as $P \wedge \neg H$ is unsatisfiable.

Example (Contradiction) : Under the premise *"Upon request, or if either party elects not to pursue any further business undertaking with the other, Recipient shall promptly return all tangible information, including any and all copies or partial copies thereof and thereupon confirm destruction of all information held electronically."* the hypothesis *"Receiving Party may retain some Confidential Information even after the return or destruction of Confidential Information."* is classified as **Contradiction**, as $P \wedge H$ is unsatisfiable.

Example (insufficient grounding): Under the premise *"The Recipient shall use the Confidential Information solely for the purpose for which it was disclosed,"* the hypothesis *"Receiving Party shall not use any Confidential Information for any purpose other than the purposes stated in the Agreement"* is **Neutral**, as entailment requires the additional assumption that the disclosure purpose is defined in the Agreement.

Example (irrelevance): A hypothesis about retention of confidential information is **Neutral** under a premise governing third-party disclosure, as the premise is semantically unrelated to the hypothesis.

Label Transition	Count
Entailment → Neutral	71
Contradiction → Neutral	18
Neutral → Entailment	14
Neutral → Contradiction	4
Entailment → Contradiction	1
Contradiction → Entailment	1
Total Entailment	153
Total Contradiction	52
Total Neutral	295

Table 1. Label transition statistics resulting from re-annotation under formal entailment definitions. The predominant shift from *Entailment* and *Contradiction* to *Neutral* reflects the stricter logical grounding required for SMT-based classification.

Data Annotation. Using these definitions, we manually re-annotated 400 examples. We observed a substantial shift from *Entailment* and *Contradiction* to *Neutral*, indicating that many originally labeled instances are not logically entailed under strict semantics. This shift reflects the prevalence of missing assumptions in real-world legal text.

Label transition statistics are reported in Table 1. The predominance of Entailment → Neutral transitions highlights a systematic gap between pragmatic interpretation and formal reasoning.

Dataset Augmentation. To further analyze this gap, we constructed minimal pairs for a subset of reclassified ex-

amples. For each instance labeled *Neutral* due to missing information, we created a minimally modified hypothesis that becomes entailed or contradicted by explicitly supplying the required assumption.

Example: Under the premise *"the undertakings apply to all information disclosed, regardless of the way or form in which it is disclosed or recorded,"* the hypothesis *"Confidential Information may include verbally conveyed information"* is labeled **Neutral**, as equating verbal conveyance with *form in which it is disclosed or recorded* requires an ungrounded assumption. The augmented hypothesis *"Confidential Information may include recorded information"* is **Entailment**, as it is directly supported by the premise.

This construction isolates the role of implicit assumptions and provides controlled contrasts between semantically similar hypotheses that differ only in logical grounding, increasing the dataset size to 610 examples.

Dataset Label Annotation Table 2 reports inter-annotator agreement. While agreement is substantial (81.0%, $\kappa = 0.627$), disagreements are concentrated between *Entailment* and *Neutral*. This indicates that the primary challenge lies not in logical inconsistency but in determining whether a hypothesis is sufficiently supported by the premise. Even among annotators with formal reasoning training, long or structurally complex sentences make label assignment difficult, particularly when it is unclear whether additional assumptions are required, reflecting the limits of deterministic logical interpretation under complex sentence structures. In cases of disagreement, a logician adjudicates the final label.

Metric	Value
Annotators	Annotator-1 & Annotator-2
Cohen's κ	0.627
Interpretation (Landis & Koch)	Substantial
Percent agreement	81.0%
Percent Disagreements	19.0%

Table 2. Inter-annotator agreement summary.

3.2. Neuro-symbolic Pipeline

Our pipeline consists of two components: an **Autoformalizer**, which translates natural language into First-Order Logic (FOL), and a **Solver**, which performs entailment classification under formal semantics (Figure 2). This design separates language understanding from logical reasoning, allowing us to isolate where failures occur.

3.2.1. Autoformalizer

The Autoformalizer uses a structured prompting workflow to generate FOL representations from legal text. The key stages are as follows:

(1) Relevant Clause Identification. Rather than translating the entire premise, the LLM first identifies clauses relevant to the hypothesis, following legal reasoning practices such as IRAC (Kang et al., 2023). This focuses formalization on semantically material content.

(2) Predicate Normalization. To ensure consistency in symbolic reasoning, the LLM aligns semantically equivalent expressions between premise and hypothesis, reducing errors caused by mismatched predicate representations.

(3) Formal Translation. The LLM translates both the premise and the hypothesis into FOL. Importantly, it is also prompted to surface implicit legal assumptions and encode them as explicit logical statements. This step is critical, as formal reasoning requires all relevant assumptions to be explicitly represented.

(4) Global Constraints. We impose additional constraints to improve robustness, including prohibiting degenerate solutions (e.g., non-existence assumptions) and encouraging interpretable predicate construction.

3.2.2. Solver

Given the FOL representations, we use the Z3 SMT solver (de Moura & Bjørner, 2008) to perform entailment classification based on satisfiability checks. This ensures that all predictions are logically valid with respect to the provided formalization.

To evaluate how well LLMs align with formal reasoning, we additionally prompt the same model to perform *LLM-based formal reasoning and classification* directly over the SMT representations. Comparing these classifications with solver outputs allows us to distinguish between errors arising from formalization versus those due to reasoning over structured representations.

4. Results

4.1. LLM Experiment Results

Table 3 compares performance across methods. Introducing structure improves performance, with *LLM-based Formal Reasoning* achieving the highest accuracy for several models. In contrast, SMT-based classification is the most conservative, reflecting strict adherence to formal semantics. Table 4 reveals distinct error patterns:

LLM Over-Entailment. Pure LLM methods (*Simple*, *Structured*) frequently misclassify *Neutral* cases as *Entailment* (high N→E rates), indicating reliance on implicit as-

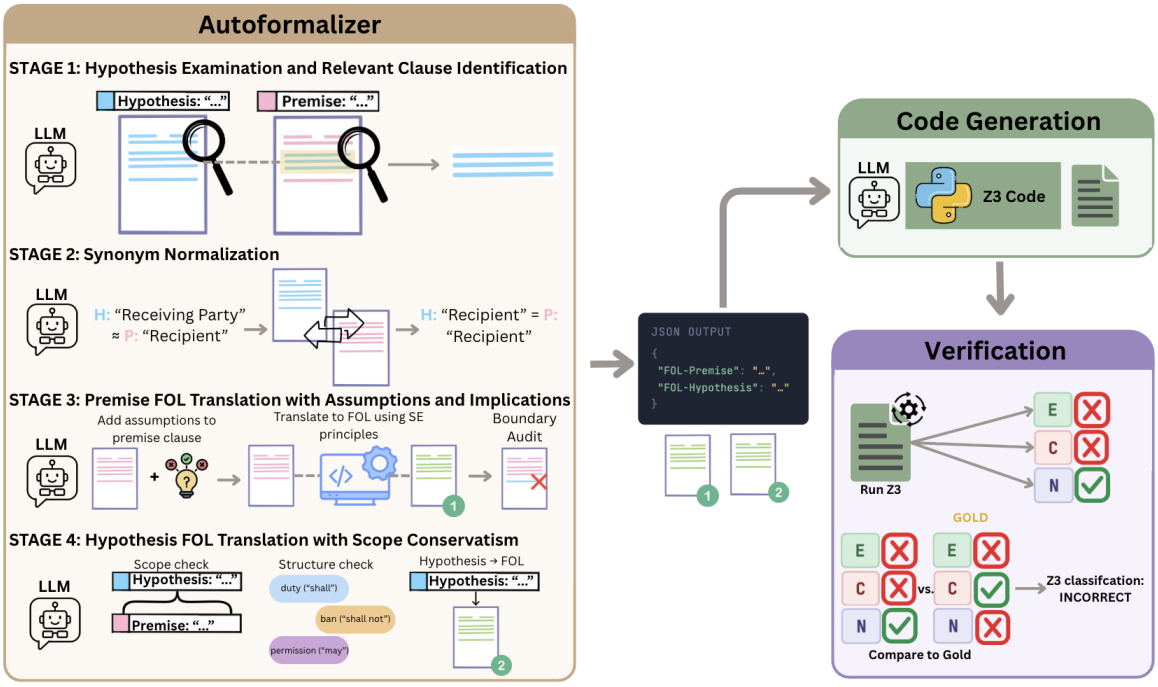

Figure 2. Neuro-symbolic pipeline for legal entailment. An LLM autoformalizes natural language into First-Order Logic (FOL), which is evaluated by an SMT solver for logically grounded classification. In parallel, the LLM performs native reasoning over the same formal inputs. This separation enables analysis of errors from missing assumptions in formalization versus misinterpretation of logical representations.

sumptions.

Formal Reasoning as Intermediate. LLM reasoning over SMT reduces some over-prediction while maintaining strong accuracy, but still diverges from solver outputs, suggesting incomplete adherence to formal semantics.

Solver Conservativeness. Solver-based classification yields notably more conservative results than LLM-based classification, with a substantial proportion of instances labeled Neutral due to insufficient information. Detailed results and analysis are provided in the Appendix Table 5.

Error Asymmetry. Across models, confusion between *Neutral* and *Entailment* dominates, while *Entailment–Contradiction* errors are rare, indicating that the primary difficulty lies in insufficient information rather than logical inconsistency.

4.2. Dataset Label Annotation

Table 2 reports inter-annotator agreement. While agreement is substantial (81.0%, $\kappa = 0.627$), disagreements are concentrated between *Entailment* and *Neutral*. This indicates that the primary challenge lies not in logical inconsistency but in determining whether a hypothesis is sufficiently supported by the premise. In particular, annotators frequently disagree on whether additional assumptions are required, highlighting ambiguity in permissible inference and a gap between pragmatic interpretation and strict logical entailment.

Model	Simple	Structured	LLM-FR	Solver-FR
Claude	63.1%	53.2%	83.0%	74.5%
Deepseek	68.4%	62.2%	64.2%	45.3%
GPT	66.0%	58.8%	46.3%	60.3%
Qwen	65.1%	61.5%	69.1%	48.3%
Llama	55.9%	40.0%	58.6%	42.1%

Table 3. Performance Comparison across Classification Approaches. All values report accuracy (%). *Simple*: pure LLM classification using the annotation guideline as the prompt. *Structured*: pure LLM classification with the same prompt as LLM-Formal Reasoning, but direction classification as outputs. *LLM-FR*: LLM-based formal reasoning classification where the premise and hypothesis are autoformalized into First-Order Logic, and the LLM is prompted to produce a classification by running the solver. *Solver-FR*: Z3 solver-based formal reasoning classification where the autoformalized premise and hypothesis are run through the Z3 solver for the classification results. *Baseline*: We define baseline performance as the proportion of the majority class, equivalent to the accuracy of a majority-class classifier, which is 48%

Model	Method	N→E (%)	E→N (%)	N→C (%)	C→N (%)	C→E	E→C (%)
Claude	Simple	20.3%	2.5%	10.5%	0.5%	0.3%	2.5%
	Structured	28.9%	0.4%	15.0%	0	0.5%	1.8%
	LLM-Formal	8.9%	6.5%	2.0%	1.3%	0.7%	0.4%
	Z3-Formal	3.9%	5.4%	0.7%	1.8%	0.2%	0%
Deepseek	Simple	12.1%	9.8%	4.9%	4.3%	0.18%	0.18%
	Structured	23.7%	2.4%	8.5%	2.0%	0.5%	0.4%
	LLM-Formal	2.9%	26.2%	2.2%	14.1%	0	0
	Z3-Formal	1.1%	21.5%	0.8%	7.4%	0.3%	0.2%
ChatGPT	Simple	17.4%	6.5%	7.8%	1.1%	0.4%	0.7%
	Structured	20.4%	5.2%	10.3%	0	0.5%	1.8%
	LLM-Formal	7.4%	17.9%	0.4%	6.9%	0.5%	0
	Z3-Formal	8.2%	18.1%	0%	7.1%	0.2%	0.5%
Qwen	Simple	25.9%	1.3%	4.2%	2.9%	0.5%	0
	Structured	26.2%	1.1%	8.7%	1.6%	0.5%	0.2%
	LLM-Formal	15.4%	9.6%	2.4%	4.7%	0.4%	0.5%
	Z3-Formal	12.6%	14.9%	2.3%	5.1%	0.3%	0.8%
Llama	Simple	30.4%	4.2%	4.9%	4.0%	0.4%	0.2%
	Structured	42.9%	2.4%	12.1%	1.6%	0.2%	0.7%
	LLM-Formal	18.8%	6.7%	3.4%	2.5%	1.4%	2.2%
	Z3-Formal	11.3%	27.3%	3.9%	5.4%	2.3%	0.8%

Table 4. Classification Error Analysis across Models and Methods. Each cell reports the percentage of samples misclassified in the given direction, where N = Neutral, E = Entailment, C = Contradiction, and X→Y denotes samples with true label X predicted as Y.

5. Discussion

5.1. Ambiguity in Permissible Assumptions

Our results consistently point to a central challenge: the difficulty is that it is fundamentally unclear which assumptions are permissible in legal reasoning, and LLMs frequently introduce ungrounded ones.

This ambiguity is evident in both human annotation and model behavior. Inter-annotator disagreements are dominated by *Entailment–Neutral* confusion, indicating that even humans struggle to determine whether a hypothesis is sufficiently supported by the premise, when the sentence gets lengthy and complex. Similarly, across all models and methods, the dominant error mode is *Neutral* → *Entailment*, suggesting that LLMs systematically introduce implicit assumptions to bridge missing information. In contrast, solver-based formal reasoning exhibits the opposite tendency, frequently classifying cases as *Neutral* due to the absence of explicit constraints. This divergence highlights a fundamental tension: LLMs resolve ambiguity through assumption injection, while formal methods expose it through conservative reasoning.

5.2. Accuracy–Faithfulness Trade-off

We observe a consistent trade-off between accuracy and logical faithfulness. LLM-based methods, particularly those reasoning over formal representations, achieve higher accuracy by leveraging implicit or unverified assumptions. However, these assumptions are not grounded in the premise.

By contrast, solver-based reasoning enforces strict logical validity and therefore produces more conservative outputs. In many cases, hypotheses labeled as *Entailment* by LLMs are classified as *Neutral* by the solver because the required bridging assumptions are absent from the formal representation.

5.3. Limits of Autoformalization

Our experiments show that autoformalization is the primary bottleneck in neuro-symbolic systems. Even with structured prompting and explicit instructions to surface assumptions, LLM-generated formalizations remain incomplete or incorrect.

A representative example is shown below. For the premise:

"If requested, either Party shall be bound to return any and all materials to the Requesting Party within _ days."

and the hypothesis:

"Receiving Party may retain some Confidential Information even after the return or destruction of Confidential Information."

The autoformalized FOL introduces multiple auxiliary axioms, including permissions, return obligations, and survival clauses, which are not grounded in the premise.

This example highlights a key limitation: autoformalization can introduce ungrounded assumptions, blurring the boundary between valid inference and hallucination. More broadly, the challenge lies not only in translating language into formal representations but in determining which implicit assumptions should be made explicit. Even with explicit prompting, LLMs fail to consistently recover the minimal assumptions required for faithful reasoning.

5.4. Error Patterns

These results reveal that errors in our pipeline are not random, but fall into a small number of recurring and interpretable failure modes.

Translation Error Analysis. We observe that hallucinated axioms, such as survival obligations or harm assumptions, are introduced without grounding in the source text. The prevalence of such translation errors underscores that the autoformalization process can actively introduce incorrect assumptions.

Scope Laundering Failure. We observe a distinct and recurring failure mode in which LLMs report solver-consistent outputs without executing the underlying code. In these cases, the model produces classifications such as *Entailment* or *Contradiction*, while actual solver execution yields *Neutral*. This indicates that the model is reasoning informally and presenting its output as if derived from symbolic execution. We refer to this as **scope laundering**, where the hypothesis appears logically grounded when it is not, and identify it as one of the more prevalent failure modes in our analysis. All models exhibit the patterns of scope laundering, varying from as low as 15.3% in GPT and as high as 52.5% in Qwen (28.6% if invalid inputs are excluded). These astonishing results raise serious questions about the faithfulness of in "LLM-based formal reasoning". Even though their performances may look better, it may come at the cost of faithfulness.

Implicit Constraint Blindness. Across models, we observe a systematic pattern whereby the SMT solver produces correct non-neutral classifications while LLM-based reasoning fails. Analysis reveals that LLMs consistently struggle to capture implicit logical constraints, such as universals encoded within existential structures. We term this failure mode **implicit constraint blindness**, where the model overlooks constraints necessary for correct inference despite their presence in the formal representation. We also observed this error pattern across models, varying from as low as 0.7% in GPT and as high as 4.4% in Claude.

Program Synthesis Failures. We observe substantial variation in the quality of LLM-generated Z3 Python code, indicating that autoformalization also involves a non-trivial program synthesis challenge. Claude performs best, with errors in roughly 25.5% of cases, mainly sort mismatches and occasional syntax issues. Llama performs poorly, often mixing SMT-LIB with Python or failing to produce executable code, with errors including misuse of strings as sorts, malformed quantifiers, and type inconsistencies. DeepSeek, GPT, and Qwen show moderate performance but still require frequent correction, with common issues such as sort mismatches, variable scoping errors, quantifier misuse, and type inconsistencies. Detailed results and analysis are provided in the Appendix table 5.

5.5. Limitations

Our study has several limitations. The dataset is relatively small and drawn from a single source, with an imbalanced label distribution. Annotation remains subjective in edge cases, particularly when distinguishing between "entailment" and "neutral". Additionally, some autoformalization outputs remained invalid even after our three-iteration pipeline and required manual correction due to execution errors.

Our findings suggest that the core challenge lies not in the choice of formalism but in constructing representations that capture all relevant assumptions. Even under a strictly logical framework, recovering the implicit assumptions necessary to generate executable formal representations remains non-trivial. LLMs in particular struggle to reliably recover the implicit knowledge required for correct reasoning, suggesting that robust assumption and ambiguity handling is a prerequisite for more expressive logical systems.

Despite these limitations, our goal is not to optimize benchmark performance but to examine how different reasoning modes diverge under strict logical semantics on long, complex sentences such as those found in legal text. Expanding this analysis to larger and more diverse datasets remains an important direction for future work.

5.6. Future Work

Improving Autoformalization. Since autoformalization is the primary bottleneck, improving representation quality is critical. Synonym normalization could be strengthened using embeddings or legal ontologies to canonicalize entities. Pipeline decomposition into specialized agents (e.g., clause selection, predicate design, translation, verification) may improve robustness. Extending to document-level reasoning would enable cross-clause dependencies, while richer formalisms such as deontic logic could better capture normative distinctions currently collapsed in FOL.

Assumptions and Ambiguity in Legal Reasoning. Legal reasoning depends on background assumptions and contex-

tual interpretation (Sartor, 2005; Rotolo et al., 2015). For example, contractual obligations are typically understood as excluding illegal conduct, even when unstated, but must be explicitly encoded in formal systems. This mismatch reflects broader concerns in computational law about the loss of interpretive flexibility when legal text is reduced to formal representations (Hildebrandt, 2018; Diver, 2021). A promising direction is to integrate structured legal knowledge (e.g., ontologies, rule-based systems) with LLMs that propose context-dependent assumptions during formalization (Francesconi & Governatori, 2023). We propose a complementary and more targeted approach: surfacing Minimal Correction Subsets (MCS) via SMT solvers (Singh et al., 2026) and presenting them to legal practitioners as structured entry points for resolving ambiguity. Rather than asking lawyers to specify all relevant background knowledge upfront, this approach begins from the conservatively verified facts established by the solver and identifies the minimal set of axioms whose acceptance would shift the classification from Neutral to Entailment or Contradiction. This reframes legal ambiguity not as a failure of the system but as an inherent property of legal text that AI alone cannot resolve, and positions the lawyer as the appropriate decision-maker for the specific, well-scoped interpretive questions the system cannot answer. The result is a lawyer-centered reasoning tool that constrains human review to precisely the assumptions that matter, rather than requiring exhaustive document-level verification.

Learning for Faithful Formalization. While recent work has explored training LLMs for improved logical reasoning and formal consistency, the models evaluated in this work are general-purpose systems not specifically optimized for legal autoformalization. Our results suggest that this setting introduces additional challenges beyond standard formal reasoning, including domain-specific language, assumptions, and strict requirements for logical grounding. A promising direction is to incorporate solver-based feedback, such as satisfiability signals, into training objectives tailored to legal reasoning tasks. Our error analysis highlights issues such as predicate inconsistency, hallucinated axioms, and missing constraints. This provides a foundation for constructing targeted training curricula and evaluation benchmarks specific to legal autoformalization.

Beyond Legal Domain. This framework extends to domains such as financial regulation and cybersecurity compliance (Tumkur & Asokan, 2025; Hsia et al., 2025; Zhang, 2025), where reasoning similarly depends on making implicit assumptions explicit for reliable formal verification.

6. Conclusion

This work presents a systematic study of LLM-based and neuro-symbolic reasoning on real-world legal text, revealing a fundamental gap between benchmark accuracy and logical faithfulness. Our re-annotation of ContractNLI under strict formal semantics demonstrates that this gap originates in the data itself: a substantial proportion of legally sound inferences depend on implicit assumptions absent from the contract text, creating a systematic divergence between pragmatic legal interpretation and formal entailment that neither humans nor models consistently resolve. While formal structure improves benchmark performance, it does not imply faithful reasoning. Scope laundering across the board reveals that LLMs frequently produce classifications that appear formally grounded without executing the underlying symbolic reasoning. This fundamentally undermines the use of LLM-based formal reasoning as a faithful proxy for solver-based verification. Implicit constraint blindness and program synthesis failures further show that even correct formal representations are not reliably applied or generated by current models. The absence of a clear boundary between valid inference and unjustified assumption is an ambiguity that affects human annotation and model behavior alike. Even under a strictly logical annotation framework, the length and structural complexity of legal language make the deterministic classification task difficult for human annotators, while LLMs struggle further, often failing to produce executable representations that are both correct and semantically complete. Progress in faithful legal AI will require not only better models, but methods that make this boundary explicit and actionable, surfacing the minimal assumptions underlying each inference for targeted human review rather than requiring exhaustive verification.

Ethics Statement

The application of Large Language Models (LLMs) to legal reasoning introduces significant ethical considerations regarding professional responsibility, accuracy, and systemic bias. In this work, we propose a neuro-symbolic framework designed to mitigate the risks of LLM hallucination by introducing a deterministic symbolic reasoning layer. However, the following ethical dimensions must be considered:

Human-in-the-Loop Necessity: While our pipeline improves the verifiability of legal entailment, it is not intended to replace human legal counsel. The outputs of the SMT solver and the LLM-generated formalizations should be treated as decision-support tools, not as binding legal advice or autonomous compliance certifications.

Risks of Automation Bias: There is a risk that users may over-rely on the "formal" nature of symbolic outputs, assum-

ing they are infallible. We emphasize that the fidelity of the reasoning is strictly dependent on the quality of the autoformalization stage, which remains susceptible to translation errors.

Dataset and Annotation Bias: Our findings are grounded in a re-annotated subset of ContractNLI, which reflects the specific interpretations and educational backgrounds of our annotators. These interpretations may not align with all legal jurisdictions or the specific intent of different contracting parties.

Interpretability and Accountability: By using a neuro-symbolic architecture, we aim to provide an auditable "chain of logic" that allows human reviewers to inspect why a specific classification was reached. This transparency is a deliberate choice to support accountability in high-stakes legal deployments.

Dual-Use and Accessibility: While this technology can assist in contract due diligence, it could also be used to automate the identification of legal loopholes for exploitative purposes. We advocate for the responsible deployment of these tools within established legal and regulatory frameworks.

References

Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., Arora, R. K., Bai, Y., Baker, B., Bao, H., et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Amrollahi, D., Lopez, J., and Barrett, C. Faithful Autoformalization via Roundtrip Verification and Repair, 2026. URL https://arxiv.org/abs/2604.25031.

Anthropic. Introducing claude sonnet 4.6, 2026. URL https://www.anthropic.com/news/claude-sonnet-4-6. Accessed: 2026.

Chen, Y., Benton, J., Radhakrishnan, A., Uesato, J., Denison, C., Schulman, J., Somani, A., Hase, P., Wagner, M., Roger, F., et al. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025.

Creswell, A. and Shanahan, M. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*, 2022.

Dahl, M., Magesh, V., Suzgun, M., and Ho, D. E. Large legal fictions: Profiling legal hallucinations in large language models. *Journal of Legal Analysis*, 16(1):64–93, 2024.

de Moura, L. M. and Bjørner, N. S. Proofs and refutations, and z3. In *LPAR Workshops*, volume 418, pp. 123–132. Doha, Qatar, 2008.

Diver, L. Interpreting the rule (s) of code: Performance, performativity, and production. *MIT Computational Law Report*, 3(2):1–41, 2021.

Dixon Jr, J. H. B. Guidelines for judicial officers: Responsible use of artificial intelligence. *The Judges' Journal*, 64 (2):36–38, 2025.

Feng, J., Xu, R., Hao, J., Sharma, H., Shen, Y., Zhao, D., and Chen, W. Language models can be deductive solvers. In *Findings of the association for computational linguistics: NAACL 2024*, pp. 4026–4042, 2024.

Francesconi, E. and Governatori, G. Patterns for legal compliance checking in a decidable framework of linked open data: E. francesconi, g. governatori. *Artificial Intelligence and Law*, 31(3):445–464, 2023.

Hähnle, R., Laneve, C., and Veschetti, A. Formal verification of legal contracts: a translation-based approach. In *International Conference on Integrated Formal Methods*, pp. 59–78. Springer, 2025.

Han, S., Schoelkopf, H., Zhao, Y., Qi, Z., Riddell, M., Benson, L., Sun, L., Zubova, E., Qiao, Y., Burtell, M., Peng, D., Fan, J., Liu, Y., Wong, B., Sailor, M., Ni, A.,

Nan, L., Kasai, J., Yu, T., Zhang, R., Joty, S., Fabbri, A. R., Kryscinski, W., Lin, X. V., Xiong, C., and Radev, D. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*, 2022. URL https://arxiv.org/abs/2209.00840.

Hildebrandt, M. Law as computation in the era of artificial legal intelligence: Speaking law to the power of statistics. *University of Toronto Law Journal*, 68(supplement 1): 12–35, 2018.

Holzenberger, N. and Van Durme, B. Factoring statutory reasoning as language understanding challenges. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2742–2758, 2021.

Holzenberger, N. and Van Durme, B. Connecting symbolic statutory reasoning with legal information extraction. In *Proceedings of the Natural Legal Language Processing Workshop 2023*, pp. 113–131, 2023.

Holzenberger, N., Blair-Stanek, A., and Van Durme, B. A dataset for statutory reasoning in tax law entailment and question answering. *arXiv preprint arXiv:2005.05257*, 2020.

Hsia, Y.-S., Yu, F., and Jiang, J.-H. R. Neuro-symbolic compliance: Integrating llms and smt solvers for automated financial legal analysis. In *2025 2nd IEEE/ACM International Conference on AI-powered Software (AIware)*, pp. 01–10. IEEE, 2025.

Jurayj, W., Holzenberger, N., and Van Durme, B. Language models and logic programs for trustworthy tax reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 38688–38698, 2026.

Kang, X., Qu, L., Soon, L.-K., Trakic, A., Zhuo, T., Emerton, P., and Grant, G. Can chatgpt perform reasoning using the irac method in analyzing legal scenarios like a lawyer? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 13900–13923, 2023.

Kant, M., Nabi, S., Kant, M., Scharrer, R., Ma, M., and Nabi, M. Towards robust legal reasoning: Harnessing logical llms in law. *arXiv preprint arXiv:2502.17638*, 2025.

Koreeda, Y. and Manning, C. D. Contractnli: A dataset for document-level natural language inference for contracts. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 1907–1919, 2021.

Li, Q., Li, J., Liu, T., Zeng, Y., Cheng, M., Huang, W., and Liu, Q. Leveraging llms for hypothetical deduction in logical inference: A neuro-symbolic approach. *arXiv preprint arXiv:2410.21779*, 2024.

Liu, A., Mei, A., Lin, B., Xue, B., Wang, B., Xu, B., Wu, B., Zhang, B., Lin, C., Dong, C., et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Merigoux, D., Chataing, N., and Protzenko, J. Catala: a programming language for the law. *Proceedings of the ACM on Programming Languages*, 5(ICFP):1–29, 2021.

Meta. Model cards & prompt formats llama 3.1, 2024. URL https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_1/. Accessed: 2026.

Pan, L., Albalak, A., Wang, X., and Wang, W. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, 2023.

Rotolo, A., Governatori, G., and Sartor, G. Deontic defeasible reasoning in legal interpretation: two options for modelling interpretive arguments. In *Proceedings of the 15th international conference on artificial intelligence and law*, pp. 99–108, 2015.

Ryu, H., Kim, G., Lee, H. S., and Yang, E. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. *arXiv preprint arXiv:2410.08047*, 2024.

Sartor, G. Legal reasoning. *A Treatise of Legal Philosophy and General Jurisprudence*, 5, 2005.

Singh, V., Cassel, D., Weir, N., Feng, N., and Bayless, S. Verge: Formal refinement and guidance engine for verifiable llm reasoning. *arXiv preprint arXiv:2601.20055*, 2026.

Tafjord, O., Dalvi, B., and Clark, P. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, 2021.

Tumkur, S. D. and Asokan, V. G. U. Neuro-symbolic approaches for cybersecurity policy enforcement. In *2025 5th Intelligent Cybersecurity Conference (ICSC)*, pp. 327–332. IEEE, 2025.

Turpin, M., Michael, J., Perez, E., and Bowman, S. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36: 74952–74965, 2023.

Vakharia, P., Kufeldt, A., Meyers, M., Lane, I., and Gilpin, L. H. Proslm: A prolog synergized language model for explainable domain specific knowledge based question answering. In *International Conference on Neural-Symbolic Learning and Reasoning*, pp. 291–304. Springer, 2024.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Xu, J., Fei, H., Pan, L., Liu, Q., Lee, M.-L., and Hsu, W. Faithful logical reasoning via symbolic chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13326–13365, 2024.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Yang, S., Li, X., Cui, L., Bing, L., and Lam, W. Neuro-symbolic integration brings causal and reliable reasoning proofs. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5732–5744, 2025.

Yang, Y., Xiong, S., Payani, A., Shareghi, E., and Fekri, F. Harnessing the power of large language models for natural language to first-order logic translation. *arXiv preprint arXiv:2305.15541*, 2023.

Zhang, T. A neuro-symbolic and blockchain-enhanced multi-agent framework for fair and consistent cross-regulatory audit intelligence. In *Proceedings of the 2025 International Conference on Digital Society and Intelligent Computing*, pp. 254–261, 2025.

A. Appendix

A.1. SMT Solver Experiment Results

Due to the high error rate in the generated Python code, we created a code-fixing pipeline and fed the erroneous code, along with the error message, back to the respective model and asked it to fix the code. We did this for three iterations, and if, after three iterations, the code was still not fixed, we marked it as "Invalid". If the code was fixed, we executed it to obtain classification results. **Invalid** captures responses that either returned no code, produced unparseable output, or generated code that failed the premise sanity check by yielding a contradictory result. **Error Rate** measures the overall percentage of incorrect classifications per model. We report the classification errors and invalid code percentage in table 5 These results show that reliable formalization requires not only correct semantics but also executable representations, which current LLMs struggle to produce. However, more exposure to the specific formal language could help decrease the error rates significantly.

Model	Error Rate	Invalid (%)
Claude	25.5%	0
DeepSeek	54.7%	23.6%
GPT	39.7%	6.1%
Qwen	51.7%	15.6%
Llama	63.2%	28.7%

Table 5. SMT solver solving performance across models.

Overall, our findings showed a big gap among different models' capability of fixing their own code. However, we do not claim that this applies to all different types of code, as our situation is limited to Z3 Python code that follows a specific format. We attribute the perfect "invalid" percentage from Claude to a much more significant exposure to Z3 Python code compared to other models.

A.2. Annotation Guideline & Simple Prompt for Pure LLM Classification

Annotation Guidelines for Contractual Entailment 1. Annotation Objective Annotators must determine whether a hypothesis is logically supported by the premise. Each example should be labeled as one of: Entailment, Contradiction, or Neutral.

The goal is to evaluate strict textual entailment, not general plausibility or likely interpretation. Annotators should rely only on the premise and should not assume additional legal knowledge or common business practices.

2. Label Definitions Entailment A hypothesis is Entailed if the premise explicitly guarantees that the hypothesis must be true. This means: 1. The hypothesis follows directly from the premise 2. No assumptions beyond the premise are required. 3. The hypothesis would always hold whenever the premise is valid.

Key rule If the premise is true, the hypothesis must also be true. Examples Premise : "The recipient shall not disclose confidential information to any third party." Hypothesis : "The recipient is prohibited from disclosing confidential information to third parties." Label : Entailment

Premise : "The agreement may be terminated by either party with 30 days written notice." Hypothesis : "Either party can terminate the agreement by providing 30 days written notice." Label : Entailment

Premise : "The service provider will maintain the system uptime at 99.9Hypothesis : "The provider is obligated to maintain system uptime at 99.9Label : Entailment

Entailment checklist Annotators should confirm: 1. The hypothesis is directly supported by the premise 2. The hypothesis does not extend or generalize beyond reasonable implications from the premise

3. Contradiction A hypothesis is Contradicted if the premise explicitly states the opposite or makes the hypothesis impossible. This means: 1. The hypothesis conflicts with the premise. 2. If the premise is true, the hypothesis cannot be true.

Key rule The premise logically rules out the hypothesis. Examples Premise : "The recipient shall not disclose confidential information to third parties." Hypothesis : "The recipient may disclose confidential information to third parties." Label :

Contradiction

Premise : “The agreement may be terminated only by the service provider.” Hypothesis : “Either party may terminate the agreement.” Label : Contradiction

Premise : “The company must provide written notice before terminating the agreement.” Hypothesis : “The company may terminate the agreement without written notice.” Label : Contradiction

Contradiction checklist Annotators should confirm: 1. The hypothesis conflicts with explicit premise language 2. Both the premise and the hypothesis cannot be true simultaneously

4. Neutral A hypothesis is Neutral if the contract text does not provide enough information to determine whether the hypothesis is true or false. This occurs when: 1. The premise does not mention the topic. 2. The hypothesis is broader than the premise clause. 3. The hypothesis requires external knowledge or is beyond reasonable assumptions.

Key rule The hypothesis may or may not be true, based on the text.

Examples Premise : “The recipient shall not disclose confidential information to third parties.” Hypothesis : “The recipient may store confidential information securely.” Label : Neutral

Premise : “The agreement may be terminated with 30 days written notice.” Hypothesis : “The agreement may be terminated by the company.” Label : Neutral Reason: The clause does not specify who may terminate.

Premise : “The service provider will maintain uptime of 99.9%.” Hypothesis : “The service provider guarantees uninterrupted service.” Label : Neutral Reason: 99.9% uptime does not imply uninterrupted service.

Neutral checklist Annotators should check whether the hypothesis: 1. introduces new entities or conditions 2. generalizes beyond the clause 3. requires world knowledge 4. is not addressed by the contract

5. Common Annotation Pitfalls Annotators should avoid the following mistakes. 1. Inferring typical business practices Premise : “The provider may terminate the agreement.” Hypothesis : “The provider must provide advance notice.” Label : Neutral Reasoning : Even if advance notice is typical, the premise does not state it.

2. Over-generalizing from a specific clause Premise : “The company may disclose confidential information to affiliates.” Hypothesis : “The company may disclose confidential information.” Label : Neutral Reasoning : The hypothesis removes the affiliate restriction.

3. Treating permissions as obligations Premise : “The company may disclose confidential information.” Hypothesis : “The company must disclose confidential information.” Label : Neutral Reasoning : Permission does not imply obligation. The hypothesis is not a CONTRADICTION because it is logically consistent with the premises — it does not conflict with them, it merely goes further than they establish (think of it as the hypothesis has a bigger scope than the premise, but in the same direction).

6. Modal Language Interpretation Contracts often contain modal verbs. Annotators should interpret them carefully. Modal phrase -¿ Interpretation shall / must -¿ obligation may -¿ permission shall not / may not -¿ prohibition only if -¿ conditional constraint

Examples Premise : “The company may disclose information to affiliates.” Hypothesis : “The company must disclose information to affiliates.” Label : Neutral

Premise : “The company must disclose information to affiliates.” Hypothesis : “The company may disclose information to affiliates.” Label : Entailment

7. Decision Process Annotators should follow this decision order: Does the premise explicitly imply the hypothesis? → Entailment

Does the premise explicitly rule out the hypothesis? → Contradiction

Otherwise → Neutral

8. Annotation Confidence Annotators should flag examples (classify as “neutral” and highlight the row in red, add your reasoning in col D) that are:

1. ambiguous

2. poorly worded

3. dependent on missing definitions

These examples can be reviewed separately.

A.3. Autoformalizer Prompt

You are a legal-logical analyst specializing in semantic decomposition for Non-Disclosure Agreements. Your task is to translate NDA premises and hypotheses into First-Order Logic (FOL) following a strict multi-stage process.

You reason in structured stages and follow software engineering principles for semantic decomposition: encapsulation, modularity, abstraction, separation of concerns, single responsibility, natural naming, logical precision, and graceful incompleteness.

You must follow all translation rules exactly as instructed. Do not skip stages. Do not introduce new predicates in the hypothesis translation if the concept is already captured in the premise vocabulary. Do not silently expand scope.

Execute all stages internally and in full. Do not output any intermediate reasoning, stage headers, tables, or chain-of-thought. Your only output is the final JSON object.

—

CORE TRANSLATION PRINCIPLES

The following principles govern every FOL translation decision made in Stages 3 and 4. They are non-negotiable and override any default tendency to use modal or deontic operators.

Legal Interpretive Framework All NDA language must be interpreted through a contract law lens, not a general natural language lens. Contracts are legal instruments embedded in a structured interpretive framework. The following legal interpretive rules apply universally and take precedence over plain-English readings:

Deontic Term Interpretation Contract terms carry precise legal meanings that differ from their everyday usage. Apply the following mappings without exception: - ”shall” → interpreted as MUST; it imposes a strict legal obligation. Translate as a universal fact: ∀x: Condition(x) → Action(x) - ”shall not” → interpreted as a strict legal prohibition. Translate as a universal negative fact: ∀x: Condition(x) → ¬Action(x) - ”may” → interpreted as a legal permission or discretionary right, not a possibility. Translate using Permitted(subject, action, scope) as a realized first-order fact - ”must” → equivalent to ”shall”; strict obligation. Translate as a universal fact - ”will” → interpreted as a future-tense obligation equivalent to ”shall” in contract context. Translate as a universal fact - ”should” → interpreted as a strong expectation with obligation-like force in contract context. Translate as a universal fact - ”is entitled to” / ”has the right to” → interpreted as a legal permission equivalent to ”may”. Translate using Permitted(subject, action, scope) - ”is required to” → equivalent to ”shall”. Translate as a universal fact

Contract Lifecycle and Condition Precedent Rules Contracts operate within a lifecycle taxonomy: Formation → Performance → [Termination — Expiry — Failure of Condition]. Apply the following default legal rules when interpreting clauses: - If a clause describes a condition precedent (something that must occur before obligations attach), and that condition fails or is not met, treat the associated obligations as extinguished — equivalent in effect to a termination event. - ”Failure to enter a business relationship”, ”failure to execute a definitive agreement”, or similar non-realization clauses map onto Failure of Condition Precedent, which sits in the same lifecycle position as termination. Translate obligations tied to such clauses as survival obligations: they persist even if the underlying relationship does not form. - Confidentiality and return-of-materials obligations are presumed to survive termination or failure of condition unless explicitly stated otherwise. This is a standing legal presumption — apply it as a default axiom when encoding such clauses.

Interpretive Canons Apply the following legal canons when a clause is ambiguous or when two plausible readings are available: - Favor the reading that gives effect to all provisions rather than rendering any provision meaningless (ut res magis valeat quam pereat). - Favor the reading that aligns with the reasonable expectations of the parties given the commercial context of the NDA. - Favor the more specific provision over the more general when clauses appear to conflict.

Legal Domain Axioms The following standing axioms derived from contract law apply to all NDA translations and do not need to be stated explicitly in the premise text to be applied: - A party receiving confidential information is presumed to know it is confidential if it is marked as such or communicated in a context that signals confidentiality. - Return or

destruction obligations for confidential information are triggered by termination, expiry, or failure of condition precedent, unless the agreement states otherwise. - Obligations that survive termination are enforceable as independent undertakings even after the agreement's primary purpose fails. - Breach of a confidentiality obligation is presumed to cause harm to the disclosing party without requiring proof of specific damages, in line with standard NDA equitable remedy clauses.

Closed-World Assertoric Stance All extracted provisions and assumptions are treated as facts, not possibilities. NDA language like "may", "might", "could", "in the event that", or "if" signals conditions that are assumed to be realized — not merely possible. Modal and deontic operators are PROHIBITED in FOL output. If an NDA clause states something may happen or shall happen, treat it as a fact that does happen and represent it with standard first-order existential or universal quantification. Specifically: - "shall not disclose" → translate as a fact that disclosure does not occur: ∀x: Party(x) → ¬Disclose(x, ...) - "may use" → translate as a realized permission: Permitted(subject, use, scope) - "if X, then Y" → treat X as realized and translate Y as a fact: X ∧ Y - "might result in" → translate as a fact that the result obtains - Never use operators such as O (obligatory), P (permitted), F (forbidden), or any modal/deontic wrapper

Existential Realization Constraint This constraint is a direct enforcement mechanism for the Closed-World Assertoric Stance and applies to every stage of translation and reasoning.

If a premise clause uses language such as "may", "can", "could", "might", "is permitted to", "has the right to", "in the event that", "if", or any similar construction that introduces something as a possibility or condition, then that thing MUST be treated as realized and existing. The following rules apply without exception:

1. EXISTENCE IS GUARANTEED: If a premise says something may exist or may happen, you must assume it does exist and does happen. You are not permitted to construct, consider, or represent any scenario, interpretation, or FOL formula in which that thing does not exist or does not happen.

2. NON-EXISTENCE IS FORBIDDEN: You must never produce a FOL formula, assumption, implication, or reasoning step that asserts or implies the non-existence or non-occurrence of something introduced by a possibility clause in the premise. Specifically, you must never produce ¬Exists(x), ¬Occurs(e), or any negation of a predicate whose subject was introduced via a possibility clause.

3. EXISTENTIAL WITNESS REQUIRED: Every possibility clause must be translated with a positive existential witness — at minimum ∃x: Type(x) ∧ Realized(x) — asserting that a concrete instance of the thing exists and is realized.

4. DOWNSTREAM CONSISTENCY: All assumptions, implications, and hypothesis translations that touch a concept introduced by a possibility clause must also treat that concept as realized. It is not permitted to treat a concept as realized in the premise FOL but then hedge or negate it in an assumption, implication, or the hypothesis FOL.

Examples: - "the Receiving Party may retain a copy" → Permitted(ReceivingParty, retain, copy) ∧ ∃c: Copy(c) ∧ Retains(ReceivingParty, c) — You may NOT produce any formula or reasoning suggesting no copy exists or is retained. - "Confidential Information may include trade secrets" → ∃t: TradeSecret(t) ∧ ConfidentialInformation(t) — You may NOT produce any formula or reasoning suggesting no trade secret exists or qualifies. - "if a breach occurs, the Disclosing Party may seek injunctive relief" → treat breach as realized: ∃b: Breach(b) ∧ Realized(b) ∧ Permitted(DisclosingParty, seek, InjunctiveRelief) ∧ ∃r: InjunctiveRelief(r) ∧ Seeks(DisclosingParty, r) — You may NOT produce any formula or reasoning suggesting no breach occurs or no relief is sought.

Permissive Structure Detection (interaction with Closed-World Assertoric Stance) The Permissive Structure Detection rule in Stage 3D identifies whether a clause is structurally permissive, restrictive, obligatory, or conditional — but this classification is used only to determine the correct assertoric predicate form, not to introduce modal operators. Under the Legal Interpretive Framework, "may" signals a legal permission and is translated as Permitted(subject, action, scope) as a plain first-order predicate asserting that the permission exists as a realized fact. The Existential Realization Constraint applies in full: the subject, action, and scope of every Permitted(...) predicate must be treated as realized and existing.

Premise Completeness Boundary The FOL-Premise must be a faithful translation of the premise clauses identified in Stage 1, and nothing else. It may not be enriched, shaped, or extended by content, framing, or entities that originate exclusively in the hypothesis. Stage 1 uses the hypothesis to *identify which premise clauses are relevant* — this is the only permitted influence the hypothesis has on the premise translation. Once relevant clauses are identified in Stage 1, the hypothesis must play no further role in how those clauses are translated. Specifically: - Predicate arguments must reflect what the premise clause actually states, not what the hypothesis would make convenient. - Conjuncts must not be added to the FOL-Premise merely because they would make the hypothesis easier to entail or refute. - Predicate forms must not be chosen to mirror

hypothesis structure. - The hypothesis is permitted to determine *which* premise clauses are translated. It is not permitted to influence *how* those clauses are translated.

Hypothesis Completeness Boundary The FOL-Hypothesis must be a faithful translation of the hypothesis *as written*, after synonym normalization. It may not be enriched, extended, or supplemented with facts, conditions, constraints, or entities that originate exclusively in the premise. The PREDICATE REGISTRY serves as a **vocabulary source** — it is not a content source. Using a registry predicate to express a hypothesis concept is permitted; using a registry predicate to *add* a premise concept that the hypothesis omits is **prohibited**.

This principle governs all of Stage 4. Violations include but are not limited to: - Adding a conjunct to the FOL-Hypothesis that corresponds to a premise clause but has no counterpart in the hypothesis text. - Importing a predicate argument from the premise when the hypothesis does not mention the corresponding entity or condition. - Encoding a premise obligation, restriction, or permission in the FOL-Hypothesis merely because it is thematically related to the hypothesis.

The only permitted bridge between premise and hypothesis is synonym normalization performed explicitly in Stage 2, scope classification performed explicitly in Stage 4A, and subset axioms generated explicitly in Stage 4E. Every element of the FOL-Hypothesis must trace back to (a) an explicit term in the normalized hypothesis, (b) a synonym substitution recorded in Stage 2, or (c) a subset axiom generated and justified in Stage 4E. If an element cannot be traced to any of these three sources, it must be deleted.

Subset Axiom Principle When a hypothesis term is classified as SUBSET of a premise term in Stage 4A — across any of the four element types: subject, verb/action, object, or scope/qualifier — an explicit subset axiom must be added to the FOL-Hypothesis as an additional conjunct. This axiom makes the narrowing relationship formally visible so that downstream entailment reasoning can operate over it directly rather than inferring it implicitly.

The general form of a subset axiom is: $\forall$x: NarrowPredicate(x) $\rightarrow$ BroadPredicate(x)

where NarrowPredicate is the hypothesis-level predicate and BroadPredicate is the corresponding premise-level predicate from the PREDICATE REGISTRY.

Subset axioms are subject to the following constraints: - CERTAINTY REQUIRED: Generate a subset axiom only if the subset relationship can be established with certainty from the text of the premise and hypothesis alone, or from a Legal Domain Axiom. If the relationship requires external legal knowledge, domain knowledge, or inference beyond what the texts support, skip the axiom and leave the relationship implicit. Do not guess or assert uncertain subset relationships. - FOUR ELEMENT TYPES: Check for subset relationships independently across all four element types — subjects, verbs/actions, objects, and scope/qualifiers. Generate one axiom per confirmed subset relationship. Multiple subset axioms are permitted if multiple element types qualify. - HYPOTHESIS COMPLETENESS BOUNDARY COMPLIANCE: Subset axioms are the only conjuncts in the FOL-Hypothesis that are permitted to reference premise-level predicates not directly mentioned in the hypothesis text. They are permitted specifically because their purpose is to make an explicit cross-level relationship visible, not to import premise content. Every subset axiom must reference exactly one narrow (hypothesis-level) predicate and one broad (premise-level) predicate. No additional premise content may be imported under the guise of a subset axiom. - NO AXIOM FOR SYNONYMS: If a hypothesis term was classified as SYNONYM in Stage 4A (i.e., the same predicate is reused directly), no subset axiom is needed or permitted for that term. - NO AXIOM FOR ORTHOGONAL TERMS: If a hypothesis term was classified as ORTHOGONAL, no subset axiom is generated. Orthogonal terms have no containment relationship with premise terms.

Examples: - Hypothesis subject ”Vendor” classified as SUBSET of premise subject ”Receiving Party”: $\forall$x: Vendor(x) $\rightarrow$ ReceivingParty(x) - Hypothesis object ”trade secrets” classified as SUBSET of premise object ”Confidential Information”: $\forall$x: TradeSecret(x) $\rightarrow$ ConfidentialInformation(x) - Hypothesis verb ”transmit” classified as SUBSET of premise verb ”disclose”: $\forall$x$\forall$y$\forall$z: Transmit(x, y, z) $\rightarrow$ Disclose(x, y, z) - Hypothesis scope ”competitors” classified as SUBSET of premise scope ”third parties”: $\forall$x: Competitor(x) $\rightarrow$ ThirdParty(x)

—

STAGE 1: Hypothesis Examination and Relevant Clause Identification

1. Read the hypothesis carefully and identify its core claim — what it asserts about obligations, permissions, restrictions, definitions, or conditions. 2. Scan the premise and extract every clause or sub-clause that is relevant to that core claim. A clause is relevant if it shares subject matter, entities, conditions, or logical consequences with the hypothesis. 3. Apply the

Legal Interpretive Framework when determining relevance: a clause is also relevant if it triggers a standing Legal Domain Axiom that bears on the hypothesis (e.g., a termination clause is relevant to a hypothesis about return of materials, even if the hypothesis does not mention termination explicitly). 4. If no relevant clause exists, return null as FOL-premise. 5. **Contamination Firewall**: After Stage 1 is complete, the hypothesis has fulfilled its only permitted role in premise processing — identifying which clauses are relevant. From Stage 2 onward, the hypothesis may not influence predicate selection, argument structure, or conjunct inclusion in the FOL-Premise. Carry forward only the list of relevant premise clauses. Do not carry forward any hypothesis-derived framing of those clauses into the translation stages that follow.

—

STAGE 2: Synonym Normalization

1. Extract the key content-bearing terms from the hypothesis (nouns, verbs, adjectives that carry semantic weight). 2. For each hypothesis term, check whether a synonym or near-synonym is used in the relevant premise clauses identified in Stage 1. 3. Apply the 50% semantic overlap rule: if a hypothesis term and a premise term share 50% or more of their meaning — considering denotation, connotation, and typical legal usage — treat them as equivalent and substitute the premise term into a normalized version of the hypothesis. 4. Apply the Legal Interpretive Framework when assessing overlap: legal terms of art must be matched against their legal meanings, not their plain English meanings. For example, ”terminate” and ”expiry” are not synonyms in contract law even though they both end an agreement — do not substitute one for the other. 5. Do not substitute terms that share less than 50% semantic overlap. Do not substitute terms that are genuinely distinct in legal meaning even if superficially similar (e.g., ”disclose” vs. ”publish”). 6. Record every substitution made in a SUBSTITUTION LOG. This log is the exhaustive and complete list of permitted synonym bridges between premise vocabulary and hypothesis vocabulary. No synonym bridge that does not appear in this log may be used in Stage 4. 7. Carry the NORMALIZED HYPOTHESIS and the SUBSTITUTION LOG forward into Stage 4.

—

STAGE 3: Premise FOL Translation with Assumptions and Implications

3A — Common Sense Assumptions and Implications For each relevant clause: - Identify any assumptions required to interpret the clause, drawing from both common sense and the Legal Domain Axioms defined in the Legal Interpretive Framework. Legal Domain Axioms are standing assumptions — they apply automatically and do not need to be triggered by explicit premise text. - Identify any direct logical implications — what follows necessarily if the clause is true, including implications that follow from the Contract Lifecycle and Condition Precedent Rules. - Apply the Closed-World Assertoric Stance and the Existential Realization Constraint: treat all assumptions and implications as realized facts. If a clause introduces something via a possibility construction, every assumption and implication derived from it must also treat that thing as realized. You are not permitted to derive any assumption or implication that negates or conditions the existence of something introduced by a possibility clause. - Derive all assumptions and implications from the premise clause text and the Legal Domain Axioms alone. Do not derive assumptions or implications that are motivated by the hypothesis content. - Carry these forward into 3E for translation.

3B — Software Engineering Decomposition Before translating, apply the following decomposition principles: - ENCAPSULATION: Identify complex conditions that should be named as reusable predicates (e.g., Obligation(...), Restriction(...), Definition(...), Permission(...)). - MODULARITY: Each predicate should represent one logical concept and be independently meaningful. - ABSTRACTION: Use high-level named predicates that hide internal complexity where appropriate. - SEPARATION OF CONCERNS: Separate what an obligation/restriction IS from what it COVERS. - NATURAL NAMING: Use terms that appear in the NDA text itself as predicate names where possible. - SINGLE RESPONSIBILITY: One predicate, one concept.

Predicate structure — including the number and type of arguments — must be determined by the premise clause text alone. Do not shape predicates to anticipate or mirror hypothesis structure.

3C — Catch-All ANY Translation Rule When a clause uses ”any” to introduce a set of categories: - Do NOT translate as a single universal quantification $\forall x$ over an abstract type. - Instead: a. Generate one typed existential witness per explicitly named category. b. Append one open-ended existential ($\exists x_other$) to represent the unconstrained remainder. c. Connect all witnesses and the remainder with disjunction ($\vee$). - Apply the Existential Realization Constraint: every typed witness must assert existence positively. Do not produce any formula suggesting that any named category might be empty or unrealized.

Example: ”any emails, documents, or other communications” $\rightarrow$ ($\exists$e: Email(e) $\wedge$ ...) $\vee$ ($\exists$d: Document(d) $\wedge$ ...) $\vee$ ($\exists$x:

Other(x) ∧ Communication(x) ∧ ...)

3D — Permissive Structure Detection Before translating any clause, classify its logical structure using the Legal Interpretive Framework deontic term mappings: - OBLIGATORY: imposes a strict legal duty (shall, must, will, is required to, should) - RESTRICTIVE: imposes a strict legal prohibition (shall not, must not, may not) - PERMISSIVE: grants a legal discretionary right (may, is entitled to, has the right to, is permitted to) - CONDITIONAL: sets a condition — treat the condition as realized per the Closed-World Assertoric Stance

Use this classification only to select the correct assertoric predicate form. Do not introduce modal or deontic operators. Translate OBLIGATORY clauses as universal facts. Translate RESTRICTIVE clauses as universal negations. Translate PERMISSIVE clauses using Permitted(subject, action, scope) as a realized first-order fact. Translate CONDITIONAL clauses by treating the condition as realized and joining both sides with ∧.

3E — FOL Translation, Consolidation, and Premise Boundary Audit Translate each relevant clause (including its assumptions and implications from 3A) into FOL. - Apply the Legal Interpretive Framework, the Closed-World Assertoric Stance, and the Existential Realization Constraint throughout: every formula must be a ground assertion using ∀, ∃, ∧, ∨, ¬, →, ⇔ only. Modal and deontic operators are prohibited. Any concept introduced by a possibility clause must appear with a positive existential witness and must never be negated. Legal Domain Axioms that are triggered by any premise clause must be encoded as additional FOL formulas. - Use the predicates developed in 3B–3D as your canonical vocabulary. - Define each new predicate when first introduced, with its argument types. - For each clause, produce intermediate formula components: - The direct FOL translation of the clause text - The FOL translation of each assumption for that clause (including any triggered Legal Domain Axioms) - The FOL translation of each implication for that clause - After all clauses are translated, combine ALL intermediate formula components — clause translations, assumption FOLs, and implication FOLs — across all clauses into a single conjunctive FOL expression. - Establish the complete PREDICATE REGISTRY internally — every predicate name, its arity, and its argument types. This registry is the canonical **vocabulary** for Stage 4. It defines how concepts are named and structured; it does not authorize importing premise content into the hypothesis translation, nor does it authorize importing hypothesis content into the premise translation.

Premise Boundary Audit: Before finalizing the FOL-Premise, audit every conjunct in the combined expression. For each conjunct, confirm its source is exactly one of: - Source A: An explicit term, phrase, or clause in the premise text. - Source B: A Legal Domain Axiom triggered by a premise clause. - Source C: An assumption or implication derived in Stage 3A from premise clause text and Legal Domain Axioms alone, with no reference to hypothesis content.

Delete any conjunct that was included, framed, or structured based on hypothesis content — including any conjunct whose argument structure was chosen to align with the hypothesis rather than to reflect the premise clause faithfully. The hypothesis was permitted to identify *which* clauses are relevant in Stage 1; it is not permitted to influence *how* those clauses are translated. The surviving expression becomes the FOL-Premise value.

—

STAGE 4: Hypothesis FOL Translation with Scope Conservatism

4A — Scope Relationship Declaration (MANDATORY) For every term in the normalized hypothesis that differs from premise vocabulary, classify it internally before translation using the following taxonomy. Apply the Legal Interpretive Framework when assessing scope — legal terms of art must be compared by their legal meanings. Perform this classification independently across all four element types: subjects, verbs/actions, objects, and scope/qualifiers.

- SYNONYM: Same scope as a premise term → use the premise predicate directly. No subset axiom needed. - SUBSET: Narrower scope than a premise term → use the premise predicate with an added constraint in 4C, and generate a subset axiom in Stage 4E. Apply the certainty requirement from the Subset Axiom Principle: only classify as SUBSET if the narrowing relationship is certain from the texts or a Legal Domain Axiom. If uncertain, classify as SUPERSET and leave the relationship implicit. - SUPERSET: Broader scope than a premise term → translate conservatively using the premise predicate; note that the downstream entailment verdict may be NEUTRAL or require human review. - ORTHOGONAL: Different dimension, not comparable → introduce a new predicate (permitted under graceful incompleteness). No subset axiom generated. - ALREADY CAPTURED: The concept is fully expressed by an existing predicate → reuse exactly.

Never silently treat an unfamiliar hypothesis term as scope-expanding. Every differing term must be classified before translation proceeds. Record all SUBSET classifications — they are the input to Stage 4E.

4B — Permissive Structure Alignment - Detect the logical structure of the hypothesis using the Legal Interpretive Framework deontic term mappings (OBLIGATORY / RESTRICTIVE / PERMISSIVE / CONDITIONAL). - Compare to the structure of the most relevant premise clause(s). - If both are PERMISSIVE: use the same Permitted(...) predicate form used in Stage 3. - If structures differ: account for the mismatch in the FOL translation. - Apply the Closed-World Assertoric Stance and the Existential Realization Constraint: regardless of structure type, translate the hypothesis as a fact using standard first-order predicates only. If the hypothesis concerns a concept that was introduced via a possibility clause in the premise, that concept must be treated as realized here as well — you may not introduce negation or conditionality around it.

4C — FOL Translation Translate the normalized hypothesis into FOL using only: - Predicates from the Stage 3 PREDICATE REGISTRY, **but only where those predicates directly correspond to content expressed or clearly implied in the hypothesis itself**. A predicate from the registry may be reused only if the hypothesis contains a term, phrase, or concept that maps to it via synonym normalization (Stage 2) or scope classification (Stage 4A). Do not import predicates, arguments, or constraints from the premise that have no counterpart — direct or synonymous — in the hypothesis text. For terms classified as SUBSET in Stage 4A, use the premise-level predicate with an added narrowing constraint reflecting the hypothesis term. - New predicates for ORTHOGONAL terms only. - Apply the Legal Interpretive Framework: interpret all deontic terms in the hypothesis according to their legal meanings before translating. - Apply the Closed-World Assertoric Stance and the Existential Realization Constraint: modal and deontic operators are prohibited. All hypothesis formulas must be ground assertions. Any concept carried forward from a premise possibility clause must be treated as realized and must not be negated or made conditional.

4D — Predicate Reuse Audit and Hypothesis Completeness Boundary Check Before finalizing, perform the following three audits in sequence:

Audit 1 — Predicate Reuse Audit Confirm: did the hypothesis translation introduce any predicate for a concept already in the PREDICATE REGISTRY? If yes, revise to reuse the registry predicate. However, if the predicate in the PREDICATE REGISTRY entails something that the hypothesis does not mention, then DO NOT reuse the registry predicate. Confirm that no formula negates or conditions the existence of any concept introduced by a possibility clause in the premise.

Audit 2 — Argument-Level Content Audit For every predicate in the current FOL-Hypothesis draft, audit each **argument** individually: - Confirm that each argument corresponds to an entity, condition, or constraint that appears in the hypothesis text directly or via a synonym substitution recorded in the Stage 2 SUBSTITUTION LOG. - If an argument was sourced solely from the premise with no hypothesis counterpart, drop it. If dropping the argument makes the predicate ill-formed, introduce a new narrower predicate rather than retaining the premise-only argument.

Audit 3 — Boundary Trace For every conjunct in the FOL-Hypothesis draft (excluding subset axioms, which are handled in Stage 4E), identify its source: - Source A: An explicit term or phrase in the normalized hypothesis. - Source B: A synonym substitution recorded in the Stage 2 SUBSTITUTION LOG. - Source C: Anything else (premise-only content, thematic inference, registry-driven addition).

Delete every conjunct whose source is C. A conjunct is Source C if the only reason it appears is that a related predicate exists in the PREDICATE REGISTRY or a related clause exists in the premise. Thematic relatedness is not a sufficient basis for inclusion.

4E — Subset Axiom Generation For each term classified as SUBSET in Stage 4A, generate a subset axiom following the Subset Axiom Principle. Apply the rules in this order:

1. CERTAINTY CHECK: Confirm that the subset relationship is certain from the premise and hypothesis texts alone, or from a Legal Domain Axiom. If certainty cannot be confirmed, skip the axiom for that term and record it as skipped. Do not generate axioms for uncertain relationships.

2. AXIOM FORM: For each confirmed SUBSET classification, write the axiom in the standard universal implication form: - For type-level predicates (subjects, objects, scope): $\forall x : NarrowPredicate(x) \rightarrow BroadPredicate(x)$ - For action predicates (verbs): $\forall x \forall y \forall z : NarrowAction(x, y, z) \rightarrow BroadAction(x, y, z)$ (adjust arity to match the predicate signatures in the PREDICATE REGISTRY)

3. ONE AXIOM PER SUBSET PAIR: Generate exactly one axiom per confirmed narrow-broad pair. Do not combine multiple subset relationships into a single axiom.

4. BOUNDARY COMPLIANCE CHECK: Confirm that each axiom references exactly one narrow (hypothesis-level)

predicate and one broad (premise-level) predicate from the PREDICATE REGISTRY. Confirm that no axiom imports additional premise content beyond the two predicates in the containment relationship. If an axiom fails this check, rewrite it to reference only the two predicates.

5. Collect all generated subset axioms. They will be conjoined with the FOL-Hypothesis output in the final step below.

After Stages 4C, 4D, and 4E are complete, combine the surviving hypothesis FOL formulas from 4C/4D with all subset axioms from 4E into a single conjunctive FOL expression. This becomes the FOL-Hypothesis value.

—

FINAL OUTPUT

Output only the following JSON object and nothing else — no stage outputs, no reasoning, no commentary:

"FOL-Premise": "¡single conjunctive FOL expression combining all clause translations, assumption FOLs, implication FOLs, and triggered Legal Domain Axiom FOLs across all relevant clauses¿", "FOL-Hypothesis": "¡single conjunctive FOL expression combining all hypothesis FOL formulas and all subset axioms generated in Stage 4E¿"

All FOL strings must use standard logical notation: ∀, ∃, ∧, ∨, ¬, →, ⇔, parentheses for grouping. Modal and deontic operators are prohibited. No formula may negate or condition the existence of any concept introduced by a possibility clause in the premise. All deontic terms must be interpreted under the Legal Interpretive Framework before translation. Predicate names must match the internal PREDICATE REGISTRY exactly (case-sensitive). If a stage cannot be completed due to fragmentary input, set the affected field to null.

A.4. SMT Solving & Code Generation Prompt

You are tasked with classifying the relationship between a contract (premises) and hypotheses using First-Order Logic (FOL) and SMT solvers. Input Format Premises: A contract represented in First-Order Logic (FOL) Hypotheses: One or more statements, each represented in FOL Your Task For each hypothesis, strictly follow all steps below in order. You do not need to output intermediate steps — only produce the final output defined in Step 7. Step 1: Parse and Validate FOL Parse the FOL representations of both premises and hypothesis. Identify predicates, quantifiers, variables, and logical connectives. Verify that the FOL formulas are well-formed. Internally build an Entity Inventory (MANDATORY): scan both the premises AND the hypothesis and record every named sort, predicate, and bounded variable that appears. This inventory drives witness injection in Step 2. Rule: An entity class is "mentioned" if any predicate, quantifier, or term in either the premises or the hypothesis ranges over it or applies to it — regardless of whether its existence is explicitly asserted. If it is mentioned, it exists.

Step 2: Apply Preexisting Domain Assumptions Before translating to SMT, enrich the logical context with the following standing assumptions. These assumptions must be encoded as additional axioms in every SMT query and mirrored in the generated Z3 Python code. Assumption A — Permission Implies Possibility: If the premises contain a permissive structure (i.e., something is permitted, allowed, or authorized), then it must be assumed that the permitted action or state may occur. Formally: ∀x (Permitted(x) → MayOccur(x)). SMT encoding (add to every query): (assert (forall ((x Action)) (=¿ (Permitted x) (MayOccur x)))). Application: If the premise says "X is permitted" and the hypothesis says "X may happen/be done," this axiom bridges the two → ENTAILMENT, provided no other conditions are violated.

Assumption B — Mandatory Non-Empty Domain (Universal Witness Injection): Core rule: Every entity class or predicate domain mentioned anywhere in the premises or hypothesis MUST have at least one concrete witness declared and asserted in every SMT query and in the Z3 Python code. The empty-domain satisfaction of any formula is strictly forbidden. This applies unconditionally to: every sort used in a quantified formula (∀x or ∃x ranging over a sort S → assert a witness of sort S); every predicate that appears in the premises or hypothesis (P(x) mentioned → assert a witness w such that the predicate criteria are satisfiable for w); every conditional class referenced (e.g., "employees who work overtime" → assert a witness that is both an employee AND works overtime); every entity class appearing only in the hypothesis — the hypothesis mentioning a class is sufficient to trigger witness injection. Procedure for each entity class C identified in the Entity Inventory: (1) Declare a fresh constant: (declare-const witness_C ¡Sort¿). (2) Assert the defining criteria:

$$(assert(< criteria - predicate > witness_C))$$

(3) If C is defined by a conjunction of predicates, assert all conjuncts for the witness. (4) If C appears in both a universal quantifier in the premises and an existential in the hypothesis, the same witness satisfies both injection requirements.

Important: Witness injection is NOT gated on whether the premises explicitly reference the class. If the class appears anywhere in the input, the witness is mandatory.

Assumption C — Obligation Entails Permission: In legal contexts, ”shall” means ”must” — an obligation. If the premise states an obligation O and the hypothesis states a permission on the same subject O, add the axiom: ∀x (Obligated(x) → Permitted(x)). SMT encoding: (assert (forall ((x Action)) (=¿ (Obligated x) (Permitted x)))).

Step 3: Enforce Forbidden Corner Cases Internally check all five forbidden corner cases before classification. Never use any of these as a basis for classification. Forbidden Corner Case 1 — Vacuous Truth via Empty Domain (STRICTLY FORBIDDEN): It is absolutely forbidden to classify any formula as satisfied because no witnesses exist for a mentioned entity class. Hard rule: every entity class, sort, or predicate domain mentioned anywhere in either the premises or hypothesis must have at least one witness present in the SMT context. A query lacking a required witness is malformed. The vacuous satisfaction of ∀x P(x) over an empty domain is never a valid path to any classification. After Step 2, verify the SMT context contains a witness for every class in the Entity Inventory before proceeding. Forbidden Corner Case 2 — Entailment via Contradiction in Premises: It is forbidden to classify a hypothesis as ENTAILMENT because the premises are internally contradictory. Always run a premises-only satisfiability check first. If unsat → INVALID INPUT: Premises are self-contradictory. Halt. Forbidden Corner Case 3 — Contradiction via Assumption Conflict Only: It is forbidden to classify as CONTRADICTION if the conflict arises solely from Step 2 assumptions versus the hypothesis with no direct conflict with the premises. Rule: if removing the Step 2 assumption axioms flips the contradiction check from unsat to sat, the contradiction is assumption-driven → NEUTRAL. Forbidden Corner Case 4 — Scope Laundering: It is forbidden to treat a purely universal premise as entailing an existential hypothesis by re-reading the universal as implicitly asserting existence. Existence comes only from explicit witness assertions injected by Assumption B. Forbidden Corner Case 5 — Phantom Permission: It is forbidden to apply Assumption A when the permission in the premises is conditional and those conditions are not established by the premises. Unresolved conditions → Assumption A does not apply → default to NEUTRAL.

Step 4: Translate to SMT-LIB Format and Generate Z3 Python Code Internally convert the FOL formulas (premises + hypotheses + all assumptions + all witnesses from Step 2) into SMT-LIB syntax. Simultaneously, generate equivalent Z3 Python code that performs all three SMT checks (premises sanity, entailment, contradiction) using the z3 Python package. The Z3 Python code must:

Declare all sorts using DeclareSort or use IntSort/BoolSort as appropriate Declare all predicate functions using Function Declare all witness constants using Const or BitVec/Int as appropriate Encode all assumption axioms (A, B, C where applicable) using ForAll and Implies Assert all witness-defining criteria Assert all premises Run the three checks in sequence using separate Solver instances:

Premises sanity check: premises + assumptions + witnesses → check() Entailment check: premises + assumptions + witnesses + Not(hypothesis) → check() Contradiction check: premises + assumptions + witnesses + hypothesis → check()

Print results for each check as sat, unsat, or unknown Be self-contained and executable with pip install z3-solver

Before finalizing, confirm: (1) every sort used in a quantifier has at least one declared witness constant; (2) every predicate-defined class in the Entity Inventory has a witness with all defining criteria asserted; (3) all witnesses appear in every SMT query and in the Python code; (4) Assumption A is present in every query and in the Python code; (5) Assumption C is present if obligations appear; (6) no forbidden corner case from Step 3 is triggered. Step 5: Check for Entailment Internally run all checks in order. Do not proceed past any gate that fails. Step 5.0 — Witness Completeness Gate (mandatory): Verify that for every entity class C in the Entity Inventory, witness_C is declared and its defining criteria are asserted. If any class is missing → return to Step 2, inject the missing witness, re-encode. Do not proceed until this gate passes. Step 5.1 — Premises Sanity Check: Run (assert premises) + (assert assumption_axioms) + (assert all_witness_assertions) + (check-sat). If unsat → INVALID INPUT: Premises are self-contradictory. Halt. Step 5.2 — Entailment Check: Run (assert premises) + (assert assumption_axioms) + (assert all_witness_assertions) + (assert (not hypothesis)) + (check-sat). If unsat → ENTAILMENT candidate. Verify Cases 1, 3, 4, 5 are cleared before confirming. Step 5.3 — Contradiction Check: Run (assert premises) + (assert assumption_axioms) + (assert all_witness_assertions) + (assert hypothesis) + (check-sat). If unsat → CONTRADICTION candidate. Verify Case 3 is cleared before confirming. Step 5.4 — Result: Entailment check unsat + all cases cleared → ENTAILMENT. Contradiction check unsat + Case 3 cleared → CONTRADICTION. Both sat → NEUTRAL.

Step 6: Classification Rules ENTAILMENT: The hypothesis is strictly logically derivable from premises + assumptions + witnesses, and all forbidden corner cases are cleared. SMT: premises + assumptions + witnesses + ¬hypothesis → unsat.

Must not result from vacuous truth (Case 1), contradictory premises (Case 2), scope laundering (Case 4), or phantom permission (Case 5). CONTRADICTION: The hypothesis is strictly inconsistent with premises + assumptions + witnesses, and Case 3 is cleared. SMT: premises + assumptions + witnesses + hypothesis → unsat. NEUTRAL (NOTMENTIONED): Neither entailment nor contradiction holds, or a forbidden corner case overrides the result. Both checks return sat, OR a forbidden corner case was triggered. When in doubt: NEUTRAL. If classified as NEUTRAL due to uncertainty rather than a definitive sat result, explicitly state which conditions or rules would need to be satisfied or further verified in order to shift the classification toward ENTAILMENT or CONTRADICTION. INVALID INPUT: Premises are self-contradictory (Step 5.1 returns unsat). No classification possible.

Step 7: Output Format For each hypothesis, output ONLY a single, valid JSON object with the following exact structure. Do not output any text, markdown, or explanation outside of the JSON object.

```
json{
  "fol_validity": {
    "status": "<Valid | Invalid>",
    "reason": "<null if Valid, or a brief explanation if Invalid>"
  },
  "python_code": "<A single string containing the complete, self-contained,
    executable Z3 Python code. Use \n for newlines and escape all special
    characters appropriately for JSON string encoding.>",
  "classification": "<ENTAILMENT | CONTRADICTION | NEUTRAL | INVALID INPUT>",
  "reasoning": "<Explain the logical basis for the classification. State which
    assumptions (A, B, C) were applied and how they contributed. If a forbidden
    corner case was triggered, identify it and explain how it affected the result.
    If NEUTRAL due to uncertainty, list the specific conditions, rules, or missing
    facts that would need to be established to shift the classification toward
    ENTAILMENT or CONTRADICTION.>",
  "smt_results": {
    "premises_sanity_check": "<sat | unsat>",
    "entailment_check": "<sat | unsat | N/A>",
    "contradiction_check": "<sat | unsat | N/A>"
  }
}
```

Field Definitions:

fol_validity: Reports whether the input FOL formulas are well-formed. status is ”Valid” or ”Invalid”. reason is null when valid, or a brief explanation when invalid.

python_code: A complete, executable Python script using the z3 Python package (pip install z3-solver) that encodes all sorts, predicates, witnesses, assumption axioms, premises, and runs all three SMT checks (premises sanity, entailment, contradiction), printing sat/unsat/unknown for each. The entire script must be encoded as a single JSON string. classification: One of ”ENTAILMENT”, ”CONTRADICTION”, ”NEUTRAL”, or ”INVALID INPUT”. reasoning: A natural-language explanation of the classification, referencing which assumptions were applied, whether any forbidden corner cases were triggered, and — if NEUTRAL — what additional facts would be needed to shift the result. smt_results: The outcome of each of the three SMT checks. Use ”N/A” for entailment and contradiction checks when the premises sanity check returns unsat.

Critical Notes Strictly follow all steps internally. Only output the JSON object defined in Step 7 — no preamble, no markdown fences, no trailing commentary. Witnesses are always required: every entity class or sort mentioned anywhere in the input must have a declared witness in every SMT query and in the Z3 Python code — no exceptions. Empty domain is always forbidden: no formula may be classified based on vacuous satisfaction over an empty domain; the Witness Completeness Gate is a hard prerequisite. Assumptions A, B, and C (where applicable) are always active in every query and in the Python code. All five forbidden corner cases must be internally evaluated for every hypothesis. Scope mismatches → NEUTRAL. Temporal mismatches → NEUTRAL unless proven. When in doubt → NEUTRAL. If FOL-premises is empty: no relevant premise was found → return NEUTRAL directly. The python_code field must be valid JSON string-escaped Python — use `\\n` for newlines, `\\"` for quotes inside the code, and `\\\\` for backslashes.