

PercepLog-0: LLM-Augmented Backward Chaining for Legal Reasoning

Parisa Kordjamshidi^{1,2*}, Sai Amrit Bulusu¹, Samer Aslan¹, Madhavan Seshadri¹,
Leslie Barrett¹, Enrico Santus¹,

¹Bloomberg, ²Michigan State University

{pkordjamshid, sbulusu6, saslan10, mseshadri, lbarrett4, esantus}@bloomberg.net

Abstract

In-depth logical reasoning in domains such as legal analysis remains challenging for large language models (LLMs), exhibiting brittle and inconsistent performance on tasks requiring multi-step deductive inference. This limitation is expected, as LLMs primarily rely on learned distributional associations rather than performing explicit step-by-step logical inference through formal rule application. To address this gap, we propose a novel framework, called PercepLog, that incorporates a classical deductive reasoning algorithm, backward chaining, within an outer control loop, while utilizing LLMs for primitive operations such as substitution and unification. By following the backward chaining algorithm, the framework benefits from the interpretability and explainability of symbolic reasoning procedures while retaining the flexibility of LLM perception for processing natural-language legal text. We evaluate the proposed method on statutory reasoning tasks across a range of legal domains as well general logical reasoning. The results demonstrate that our approach enables more robust, explainable, and principled reasoning over unstructured legal text, where exhaustive formalization is often brittle and prone to hallucination. Our method yields average relative performance improvements of 58% over direct LLM reasoning and 16% over formal LLM-based translation and Prolog inference pipelines.

1 Introduction

Legal reasoning is a complex and high-stakes task that requires multi-step reasoning over interdependent legal conditions, where errors may lead to severe real-world consequences (Richmond et al., 2023). Before large language models (LLMs), statutory reasoning was commonly approached through translating natural language into formal

logic, enabling verification and explainability (McCarty, 1977). However, such approaches struggled to scale because legal cases contain noisy and unstructured information, while legal rules are often ambiguous, context-dependent, and continuously evolving (Ashley, 2017; Sartor, 2010). Recent advances in LLMs have enabled applications in legal drafting, case analysis, and compliance monitoring, but their reasoning remains unreliable for high-stakes legal analysis (Chalkidis et al., 2020; Guha et al., 2023). Key limitations include hallucination, bias, and the lack of interpretable or verifiable reasoning traces (Dehghani et al., 2025; Dahl et al., 2024; Magesh et al., 2025). More fundamentally, LLMs do not explicitly execute formal deductive procedures, making them unstable for tasks requiring compositional reasoning and systematic rule application (Nguyen et al., 2025). Recent neurosymbolic approaches attempt to address these issues by translating legal text into formal logical representations (Kordjamshidi et al., 2026). However, translating large legal documents is itself prone to hallucination, inconsistency, and sensitivity to noisy information (Jurayj et al., 2025). Moreover, these systems often need engineered ontologies and canonical representations that are difficult to maintain in real-world domains (Sadowski and Chudziak, 2025). Developing more sophisticated neurosymbolic approaches that combine formal logic with the capabilities of LLMs is an active research direction (Kazemi et al., 2023; Lee and Hwang, 2025; Li et al., 2026). However, these approaches provide limited support for complex first-order logical representations and rely on facts formalization within a closed set of facts and rules. We extend this line of research and propose a new lightweight Prolog implementation, called PercepLog, that operates over formal Prolog rules and queries together with raw-text documents. Rather than fully translating legal cases into symbolic logic, PercepLog performs backward chaining directly over raw text

*This project was conducted while the author was on sabbatical at Bloomberg.

by delegating primitive operations, such as substitution and unification, to LLMs. This enables query-driven information extraction during proof search, allowing the system to incrementally retrieve only the information necessary to resolve the current reasoning goal. As a result, PercepLog avoids exhaustive semantic parsing of large and noisy legal documents while preserving explicit multi-step symbolic reasoning. We implement the framework through a custom Python-based first-order logic parser and backward-chaining engine integrated with an LLM-based extraction component. Although the framework still depends on rule quality, our results show that even automatically translated rules can achieve strong performance when combined with query-driven reasoning, with further gains when rules are curated through human oversight. We evaluate PercepLog on statutory reasoning tasks from DeonticBench (Dou et al., 2026) across domains including tax, housing, immigration appeals (USCIS), and airline regulations. The results show PercepLog outperforms both direct LLM reasoning and fully translation-based neurosymbolic pipelines while providing more robust and interpretable multi-step reasoning. Our contributions are summarized as follows: **1)** We propose PercepLog, a formal deductive reasoning framework that *combines LLM-based perception with Prolog-inspired symbolic reasoning*. It performs backward chaining directly over raw text by delegating primitive substitution operation to LLMs. This query-driven approach avoids exhaustive formalization by incrementally extracting only information relevant to reasoning goals; **2)** We provide *extensive experiments across five legal domains and five backbone LLMs*, demonstrating consistent improvements over both direct LLM reasoning and fully translation-based neurosymbolic pipelines; **3)** In contrast to prior neurosymbolic systems that operate over closed symbolic contexts and fact translation, PercepLog performs over raw textual documents, enabling more *general reasoning over noisy and unstructured text* and making the framework potentially applicable to a broad range of logical reasoning domains.

2 Related Work

Existing approaches to LLM-based logical reasoning vary along three axes: degree of formalization, role of external solvers, and division of labor between neural and symbolic components. Sev-

eral studies have used LLMs primarily as translators into formal logic for both general logical reasoning (Pan et al., 2023) and legal reasoning benchmarks (Jurayj et al., 2025). Although recent work continues to follow this paradigm (Jiang et al., 2024; Sehgal and Liu, 2024), accurate translation into formal representations remains challenging, as reflected by the low performance on recent benchmarks (Dou et al., 2026). Several recent studies guide LLMs toward explicit logical reasoning by combining natural language and formal representations. SymbCoT (Xu et al., 2024) extends the translation-to-formal-representation paradigm with planning, formal solving, and verifier feedback, an approach that has become common in recent agentic systems (Ali et al., 2025). FLARE (Arakelyan et al., 2025) similarly uses LLMs to plan solutions, formalize queries into logic programs, and simulate execution without an external solver. Formalizing intermediate chain-of-thought (CoT) reasoning steps has emerged as another promising direction for complex reasoning tasks. VeriCOT (Feng et al., 2026), for example, iteratively verifies and formalizes reasoning steps against contextual evidence, reducing hallucination by constraining formal representations to remain grounded in the input context. Beyond inference-time reasoning, some approaches use formal solver outputs to generate training data for LLMs. For example, Yang et al. (2025) apply solver-in-the-loop reinforcement learning to jointly optimize chain-of-thought reasoning and formal program generation for geometric reasoning. FACT (You et al., 2026) similarly alternates between text and code to encourage logical consistency during training.

Closely related to our work, several recent neurosymbolic approaches have drawn inspiration from backward chaining for deductive reasoning with LLMs. LAMBADA (Kazemi et al., 2023) was the first to decompose backward chaining into prompted reasoning modules, including fact checking, rule selection, goal decomposition, and sign agreement, demonstrating strong improvements over Chain-of-Thought reasoning on synthetic benchmarks. SymBa (Lee and Hwang, 2025) subsequently extended this direction by incorporating symbolic SLD resolution with explicit unification and backtracking. HBLR (Li et al., 2026) further extends this line of work with a selective symbolic translation mechanism that dynamically decides whether to reason in natural language or translate into FOL and invoke a symbolic solver.

All of these systems, however, operate over *closed theories* in which facts and rules are provided upfront as enumerated statements. The LLM therefore primarily serves either to check entailment (e.g., LAMBADA) or to translate natural-language statements into formal symbolic representations (e.g., SymBa) including facts formalization. Moreover, existing logical reasoning benchmarks typically contain relatively clean and pre-structured factual information.

In contrast, PercepLog follows a more general Prolog-inspired reasoning framework with support for parsing first-order logical expressions, symbolic backward chaining, and selected built-in operations in its initial implementation, PercepLog-0¹. More importantly, our framework differs from prior work in two key ways. First, rules are represented formally in Prolog, making rule selection and goal decomposition exact symbolic operations rather than LLM calls. Second, the LLM acts as a demand-driven information extraction by variable substitution operating directly over raw case documents and substituting goal variables based on unstructured legal text rather than reasoning over pre-enumerated formalized facts. This hybrid integration of symbolic rules with raw-text reasoning is particularly well suited for legal domains, where statutes evolve slowly and can be curated, while case documents arrive continuously and vary substantially in structure and content.

3 PercepLog Framework

Problem Setting. The logical reasoning problem considered in this paper consists of three components: a query Q , a set of logical rules R , and a context C . The goal is to infer the answer to the query according to the rules of reasoning, where all inputs and outputs are expressed in natural language. An example is shown in Figure 1. In this case, the output is a binary Entailment/Contradiction label indicating whether the query follows from the tax exemption rule. The query may also ask for a tax amount, in which case the output is numerical.

A common approach is to translate the case, rules, and query into a formal language such as Prolog and perform symbolic inference (Pan et al., 2023). However, such pipelines may hallucinate incorrect facts while missing necessary information. To address this issue, we introduce *PercepLog*, a generic deductive reasoning framework inspired

Case Text: Alice and Bob have been married since 2 Feb 2015. Bob has no income for 2015. **Query:** Alice can receive an exemption for Bob under section 151(b) for the year 2015. **Rules:** **Statute (IRC §151(b)).** An exemption of the exemption amount for the taxpayer; and an additional exemption of the exemption amount for the spouse of the taxpayer if a joint return is not made by the taxpayer and his spouse, and if the spouse, for the calendar year in which the taxable year of the taxpayer begins, has no gross income and is not the dependent of another taxpayer. **Label:** Entailment

Figure 1: Example from the SARA Binary dataset. Only the relevant statute/rule is displayed.

by Prolog that performs query-driven information extraction directly over raw case text. The key innovation is that PercepLog queries only the factual information required at each step of the backward-chaining inference algorithm (Lloyd, 1987). We rely on LLMs to obtain the Prolog-formatted query and rules, but perform iterative reasoning directly over the raw case text to determine whether the query is entailed. The algorithm is detailed below.

LLM-augmented Backward Chaining. Given a goal $G = p(t_1, \dots, t_n)$, a knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R})$ consisting of facts ($\mathcal{F}$) and rules ($\mathcal{R}$), and access to an LLM $\mathcal{L}$ with legal scenario context $\mathcal{C}$, the LLM-augmented backward-chaining procedure operates as shown in Algorithm 1. The algorithm queries the LLM to extract grounded goals or, when the goal contains unbound variables, to generate all substitutions (θ), i.e., all possible groundings of the goal predicate variables from the textual context $\mathcal{C}$, followed by unification. If the unification is successful, the LLM-derived substitutions θ are returned, otherwise, the algorithm continues with the formal rule expansion step. Next, the algorithm attempts to unify G with the rule head and recursively proves each body goal $b_i\theta$. The original backward-chaining algorithm is included in Appendix A. The key insight is that the LLM serves as a *demand-driven fact oracle*, returning substitutions θ only for predicates that require grounding. This combines the rigor of symbolic inference with the flexibility of neural fact extraction in the PercepLog framework.

Rule and Query Formalization. To apply the

¹Claude Code was used to assist with the implementation.

Rules:

```

spouse_exemption_allowed(Taxpayer, Spouse) :-
    marital_status(Taxpayer, married),
    not_joint_return(Taxpayer, Spouse),
    no_gross_income(Spouse),
    not_dependent_of_another(Spouse).

alice_eligibility_for_bob_exemption :-
    married_to(alice, bob),
    spouse_exemption_allowed(alice, bob).

```

Facts (extracted from text):

```

marital_status(alice, married).
married_to(alice, bob).
no_gross_income(bob).
not_joint_return(alice, bob).
not_dependent_of_another(bob).

```

Query:

```

:- alice_eligibility_for_bob_exemption.

```

Figure 2: Example from the SARA Binary dataset (IRC §151(b)). The statute defines conditions for a spousal exemption. PercepLog proves the claim by matching the rule head, then recursively proving each subgoal against facts extracted from the case text.

LLM-augmented backward-chaining algorithm, the rules and query must first be translated into formal representations. For this step, we follow the prompting approach proposed in DeonticBench (Dou et al., 2026), where LLMs generate case-specific Prolog rules and queries. In this setting, the prompt includes the case, query, and rules in natural language. The LLM is instructed to select the relevant rules and generate consistent formal Prolog representations. The resulting rules are then provided to the backward-chaining engine, which operates directly over the original case text. In some experimental settings, the full translated case is also used as the initial knowledge base, although query-driven LLM extractions are prioritized when they conflict with the pre-extracted facts. Figure 2 shows the ground-truth Prolog code provided in the DeonticBench dataset, while Figure 3 illustrates one step of the backward-chaining process, where the LLM is queried to produce the substitution θ for a formal query. Our experiments evaluate both ground-truth and LLM-generated rules to analyze the effectiveness of LLM-augmented backward chaining in each setting. Figure 4 presents an example proof tree generated by PercepLog during LLM-based fact extraction.

Case Text: Alice and Bob have been married since 2 Feb 2015. Bob has no income for 2015.

LLM Extraction Prompt:

```

## Goal to Extract: married_to(X1, X2)

Each tuple MUST have exactly 2 arguments. Extract ALL instances of predicate married_to from the legal scenario. Return a JSON array of argument tuples. Use lowercase for atoms. If this predicate cannot be extracted from the text, return [].

```

LLM Response: `[["alice", "bob"]]`

Figure 3: Example of LLM EXTRACTION during backward chaining. When proving `married_to(alice, bob)`, the LLM extracts the fact from the case text. The result is unified with the goal and added to $\mathcal{F}$.

4 Experiments

In this section, we study the following empirical questions: **Q1)** Can PercepLog outperform state-of-the-art LLMs on the DeonticBench legal reasoning benchmark? **Q2)** Can PercepLog outperform the pipeline of translation to Prolog and calling Prolog inference engine? **Q3)** Given ground-truth rules, how does query-driven fact extraction compare to exhaustive case translation for Prolog execution? **Q4)** How PercepLog perform on general logical reasoning benchmarks? ²

4.1 Datasets

DEONTICBENCH (Dou et al., 2026) contains 6,232 legal cases in five domains: U.S. federal income tax (SARA Numeric and Binary (Holzenberger et al., 2020)), airline baggage fee policies, state housing and eviction law, and USCIS immigration appeals. Each instance includes a natural-language question, relevant statutory text, a ground-truth label, and a verified SWI-Prolog program encoding the applicable rules and facts. Table 1 summarizes the domain statistics, including label distribution (Balance, i.e., percentage split for binary tasks and number of unique values for numeric ones) and approximate context length in tokens.

Distractor-augmented SARA Binary Hard Dataset A key advantage of PercepLog is its ability to operate directly over large and noisy legal documents without requiring complete formalization. We hypothesize that query-driven backward

²Our code and data will be publicly released through this GitHub link: <https://github.com/HLR/PercepLog>.

Algorithm 1 LLM-Augmented Backward Chaining

Require: Goal g , rule base $\mathcal{R}$, fact base $\mathcal{F} \leftarrow \emptyset$, substitution θ , LLM $\mathcal{L}$, context $\mathcal{C}$

Ensure: Set of substitutions under which g is derivable

```
1: function PROVELLM( $g, \theta$ )
2:    $found \leftarrow \text{FALSE}$ 
3:    $\mathcal{F} \leftarrow \mathcal{F} \cup \text{LLMEXTRACT}(\mathcal{L}, g, \mathcal{C})$  ▷ Phase 1: LLM Fact Extraction
4:   for each  $f \in \mathcal{F}$  do
5:      $\theta' \leftarrow \text{UNIFY}(g, f, \theta)$ 
6:     if  $\theta' \neq \text{fail}$  then yield  $\theta'$ ;  $found \leftarrow \text{TRUE}$ 
7:   if  $found$  then return
8:   for each  $(h \leftarrow b_1, \dots, b_k) \in \mathcal{R}$  do ▷ Phase 2: Rule Expansion
9:      $\theta' \leftarrow \text{UNIFY}(g, h, \theta)$ 
10:    if  $\theta' \neq \text{fail}$  then
11:      for each  $\sigma \in \text{PROVELLMBODY}(b_1, \dots, b_k, \theta')$  do yield  $\sigma$ 
12: function PROVELLMBODY( $b_1, \dots, b_k, \theta$ )
13:   if  $k = 0$  then yield  $\theta$ ; return
14:   for each  $\theta' \in \text{PROVELLM}(b_1\theta, \theta)$  do
15:     for each  $\sigma \in \text{PROVELLMBODY}(b_2, \dots, b_k, \theta')$  do yield  $\sigma$ 
```

Domain	Whole	Hard	Label	Balance	Tokens
SARA Num.	100	35	Int (\$)	96 uniq.	5.9k
SARA Bin.	276	30	0/1	50/50	5.9k
Airline	300	80	Int (\$)	285 uniq.	3.2k
Housing	5,314	78	yes/no	67/33	2.2k
USCIS-AAO	242	28	Acc./Dis.	50/50	0.8k
Total	6,232	251			

Table 1: Overview of DEONTICBENCH domains.

chaining becomes even more beneficial in realistic legal settings, where documents are longer and contain substantial irrelevant or distracting information not captured by existing benchmarks. To evaluate this hypothesis, we construct a distractor-augmented version of the 30-case SARA Binary hard split using 53 validated (Prolog fact, natural-language sentence) pairs drawn from other cases in the same tax domain. We augment the textual part of each case with additional sentences (e.g. the Case Text shown in Figure 1). For each case we augment, we first extract every predicate referenced in the case’s ground-truth rules and verification query (e.g. the predicates extracted from the text in Figure 2). Any sentence in the pool whose corresponding predicates overlap with this set is excluded, ensuring that no distractor can participate in SWI-Prolog’s proof search. We select 18–26 distractor entries per case and shuffle them uniformly at random with the original facts and sentences. Examples are provided in Appendix B. The ground-

truth rules and queries remain unchanged. All 30 augmented cases are verified with SWI-Prolog, confirming identical entailment results to their unaugmented counterparts.

4.2 Evaluation Methodology

Metrics. We report accuracy for all five domains. For SARA Numeric and Airline, where the output is a dollar amount, a prediction is counted as correct if it falls within $\pm\$1$ of the ground-truth real value. SARA Binary, USCIS-AAO, and Housing require an exact match with a binary output, that is, Entailment/Contradiction, Accepted/Dismissed, and yes/no, respectively. Abstentions, that is, unparseable output, SWI-Prolog errors, or PerceLog timeouts count as incorrect.

Confidence Intervals (CI). For our experiments, values are mean_{-CI}^{+CI} for 95% confidence intervals across 5 independent runs. Although temperature 0 is nominally deterministic, API-level non-determinism (batching, kernel selection, internal chain-of-thought sampling) introduces inter-run variation that this design captures.

4.3 Baselines and Model Variations

Direct QA. We reproduce the *Direct* baseline from (Dou et al., 2026), where the LLM receives the statutes and query in a single prompt and generates a free-text answer parsed via regex.

Prolog. This baseline uses LLMs to generate self-contained SWI-Prolog programs which are then

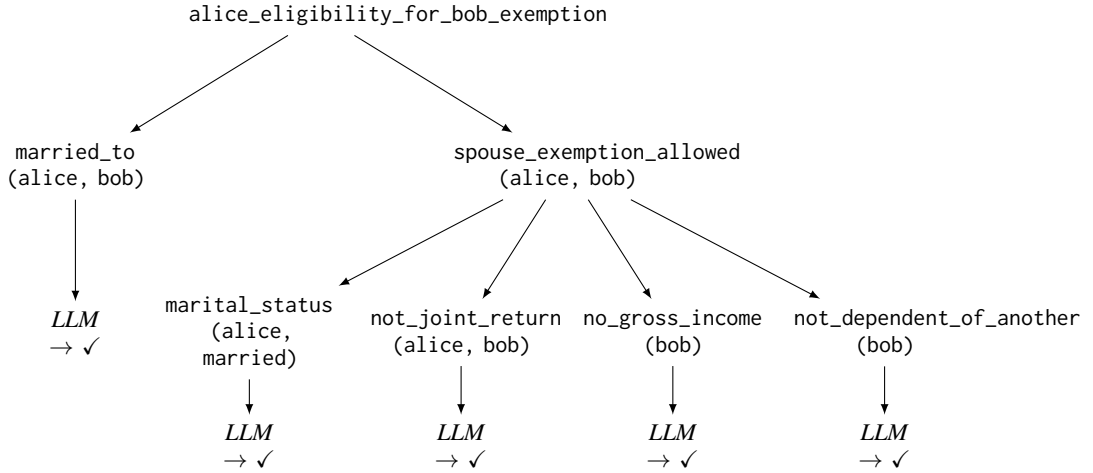


Figure 4: Proof tree for the claim in Figure 1. The root goal is resolved by rule expansion into two subgoals. `spouse_exemption_allowed` is further expanded into four conditions. Each leaf is resolved by LLM extractions from the case text, yielding Entailment.

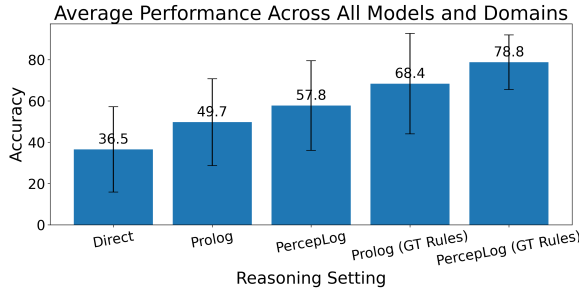


Figure 5: Overall performance averaged over Table 2 results on 5 DeonticBench domains and 5 LLMs.

executed by SWI-Prolog solver.

PercepLog with Predicted Rules. This setting is the end-to-end automated PercepLog pipeline. The LLM first translates the rules and query into formal representations, which are then passed to PercepLog together with the raw case text. We expect this approach to outperform the full-translation Prolog baseline while preserving explainability and explicit reasoning traces.

PercepLog with GT Rules. In this setting, PercepLog is provided with ground-truth Prolog rules. The goal is to isolate the effect of rule-generation errors and evaluate the advantage of query-driven fact extraction directly from raw case text.

Prolog with GT Rules. To compare with PercepLog using ground-truth rules, we use a setting where the LLM translates the full case into formal facts in a single step, which are then combined with the GT rules and executed using a standard Prolog solver. To maintain consistency between facts and rules, the ground-truth Prolog rules are included in the translation prompt. This provides a strong

comparison baseline for isolating the advantage of dynamic fact extraction in PercepLog independent of rule-generation errors. In all PercepLog experiments, we use the LLM-translated case facts as a noisy backup knowledge base. PercepLog may therefore start from either an empty or translated fact KB, while prioritizing dynamically extracted facts during LLM-augmented backward chaining.

4.4 Results and Analysis

Table 2 compares all the model variations and baselines. The overall trend is shown in Figure 5 which averages over all five domains and five different LLMs. The trend shows a significant relative improvement of about 58% when comparing PercepLog with Direct QA. It also shows a relative improvement of about 16% when comparing PercepLog with full translation using standard Prolog. The relative gain with PercepLog is significant (about 15%) when the ground-truth rules are given compared to standard Prolog.

Direct QA vs. Structured Reasoning. Direct QA exhibits highly unstable behavior across domains and models. While stronger reasoning-oriented models such as GPT-5.4 and Claude Opus 4.6 achieve competitive performance on some tasks, performance remains inconsistent overall, particularly on Housing and Airline. For example, GPT-5.4 achieves only 3.2% accuracy on Airline despite strong performance on USCIS-AAO, highlighting the difficulty of reliably performing implicit symbolic reasoning directly from text. Similar instability is observed across other models, suggesting that end-to-end prompting alone is insufficient for ro-

Model	Setting	SARA Num.	Airline	SARA Bin.	USCIS-AAO	Housing
GPT-4.1	Direct	18.3 ^{+5.4} _{-5.4}	5.8 ^{+3.4} _{-3.4}	33.3 ^{+5.1} _{-5.1}	64.3	20.5 ^{+1.1} _{-1.1}
	Prolog	19.4 ^{+5.3} _{-5.3}	79.5 ^{+3.6} _{-3.6}	64.0 ^{+3.5} _{-3.5}	40.7 ^{+4.0} _{-4.0}	41.5 ^{+3.1} _{-3.1}
	PercepLog	76.7	79.5 ^{+3.6} _{-3.6}	64.7 ^{+4.7} _{-4.7}	42.9 ^{+3.1} _{-3.1}	42.8 ^{+3.3} _{-3.3}
	Prolog (GT Rules)	25.7 ^{+2.5} _{-2.5}	35.5 ^{+3.6} _{-3.6}	76.7	90.0 ^{+2.0} _{-2.0}	73.6 ^{+1.4} _{-1.4}
	PercepLog (GT Rules)	76.0 ^{+3.2} _{-3.2}	35.8 ^{+3.6} _{-3.6}	80.0	90.0 ^{+2.0} _{-2.0}	80.8 ^{+1.9} _{-1.9}
GPT-5.4	Direct	32.0 ^{+5.8} _{-5.8}	3.2 ^{+1.8} _{-1.8}	60.0 ^{+7.2} _{-7.2}	77.1 ^{+2.4} _{-2.4}	22.8 ^{+3.1} _{-3.1}
	Prolog	19.4 ^{+11.0} _{-11.0}	93.5 ^{+2.0} _{-2.0}	54.7 ^{+6.3} _{-6.3}	60.7 ^{+8.3} _{-8.3}	28.2 ^{+4.1} _{-4.1}
	PercepLog	28.6	93.5 ^{+2.0} _{-2.0}	75.3 ^{+3.7} _{-3.7}	63.6 ^{+5.8} _{-5.8}	28.2 ^{+4.1} _{-4.1}
	Prolog (GT Rules)	27.4 ^{+1.9} _{-1.9}	97.0 ^{+0.8} _{-0.8}	79.3 ^{+1.9} _{-1.9}	72.1 ^{+4.9} _{-4.9}	76.2 ^{+1.8} _{-1.8}
	PercepLog (GT Rules)	73.7 ^{+1.6} _{-1.6}	97.0 ^{+1.4} _{-1.4}	80.0	72.1 ^{+4.9} _{-4.9}	79.5 ^{+1.9} _{-1.9}
Claude Opus 4.6	Direct	56.6 ^{+5.3} _{-5.3}	21.7 ^{+4.2} _{-4.2}	43.3 ^{+27.8} _{-27.8}	75.0	27.4 ^{+2.1} _{-2.1}
	Prolog	53.7 ^{+3.9} _{-3.9}	79.8 ^{+2.3} _{-2.3}	62.0 ^{+3.7} _{-3.7}	47.9 ^{+2.4} _{-2.4}	25.4 ^{+2.1} _{-2.1}
	PercepLog	76.7	77.2 ^{+3.0} _{-3.0}	88.7 ^{+3.7} _{-3.7}	67.9 ^{+3.1} _{-3.1}	26.9 ^{+1.1} _{-1.1}
	Prolog (GT Rules)	28.6	97.5	87.3 ^{+3.5} _{-3.5}	96.4	65.9 ^{+1.4} _{-1.4}
	PercepLog (GT Rules)	70.9 ^{+1.6} _{-1.6}	98.2 ^{+1.8} _{-1.8}	85.3 ^{+2.3} _{-2.3}	96.4	75.6
Claude Sonnet 4.5	Direct	45.1 ^{+5.8} _{-5.8}	4.5 ^{+3.7} _{-3.7}	37.3 ^{+3.5} _{-3.5}	60.7	42.1 ^{+0.7} _{-0.7}
	Prolog	28.6 ^{+2.5} _{-2.5}	86.2 ^{+2.2} _{-2.2}	59.3 ^{+4.5} _{-4.5}	61.4 ^{+5.8} _{-5.8}	23.8 ^{+2.4} _{-2.4}
	PercepLog	32.0 ^{+3.0} _{-3.0}	86.2 ^{+2.2} _{-2.2}	72.7 ^{+3.5} _{-3.5}	63.6 ^{+3.7} _{-3.7}	23.8 ^{+2.4} _{-2.4}
	Prolog (GT Rules)	28.6	96.2 ^{+1.9} _{-1.9}	82.7 ^{+1.9} _{-1.9}	92.9	76.7 ^{+1.7} _{-1.7}
	PercepLog (GT Rules)	69.1 ^{+3.9} _{-3.9}	97.2 ^{+3.0} _{-3.0}	82.7 ^{+1.9} _{-1.9}	92.9	81.5 ^{+1.4} _{-1.4}
Gemini 2.5 Flash	Direct	45.7 ^{+5.0} _{-5.0}	20.8 ^{+3.2} _{-3.2}	23.3	50.0 ^{+5.4} _{-5.4}	22.3 ^{+2.4} _{-2.4}
	Prolog	26.3 ^{+5.3} _{-5.3}	48.4 ^{+4.6} _{-4.6}	65.3 ^{+3.7} _{-3.7}	39.3 ^{+3.1} _{-3.1}	33.8 ^{+3.7} _{-3.7}
	PercepLog	30.9 ^{+3.0} _{-3.0}	46.4 ^{+4.5} _{-4.5}	70.0 ^{+4.1} _{-4.1}	47.9 ^{+5.1} _{-5.1}	37.2 ^{+3.4} _{-3.4}
	Prolog (GT Rules)	28.0 ^{+1.6} _{-1.6}	66.5 ^{+3.7} _{-3.7}	80.0	60.7	67.4 ^{+2.4} _{-2.4}
	PercepLog (GT Rules)	72.6 ^{+1.9} _{-1.9}	68.2 ^{+3.2} _{-3.2}	80.0	60.7	72.8 ^{+1.3} _{-1.3}

Table 2: PercepLog compared to baselines across legal reasoning domains with various LLM backbones.

Setting	GPT-5.4	GPT-4.1	Claude Opus 4.6	Claude Sonnet 4.5	Gemini 2.5 Flash
Direct	60.0 $\pm$ 4.1	65.3 $\pm$ 7.0	69.3 $\pm$ 3.5	65.3 $\pm$ 2.3	68.7 $\pm$ 2.3
Prolog	36.6 $\pm$ 3.6	37.6 $\pm$ 4.8	38.8 $\pm$ 2.5	36.0 $\pm$ 3.7	41.5 $\pm$ 1.3
PercepLog	69.3 $\pm$ 3.5	72.0 $\pm$ 3.7	75.3 $\pm$ 8.6	69.3 $\pm$ 3.5	62.0 $\pm$ 2.3
Prolog (GT Rules)	68.9 $\pm$ 3.3	69.3 $\pm$ 2.7	68.3 $\pm$ 2.6	68.0 $\pm$ 2.3	73.3 $\pm$ 1.4
PercepLog (GT Rules)	90.7 $\pm$ 1.9	90.0 $\pm$ 0.0	90.0 $\pm$ 0.0	90.7 $\pm$ 1.9	90.0 $\pm$ 0.0

Table 3: Results on the distractor-augmented SARA Binary hard split. Error bars denote 95% CI over 5 runs.

bust legal reasoning. Results for additional LLMs, including open-weight models, are reported in Appendix C.

Generated Rules. When reasoning over LLM-generated rules, PercepLog generally improves over standard Prolog execution. This setting is particularly important because it evaluates robustness under noisy and imperfect formalizations. Since the generated rules and facts may contain missing predicates, hallucinations, or inconsistent representations, symbolic execution alone often fails. In contrast, PercepLog can recover from partial or incomplete representations by performing query-driven extraction directly from the original text during backward chaining. Notable gains include an increase for Claude Opus 4.6 from 62.0% to 88.7%

on SARA Binary and for GPT-5.4 from 54.7% to 75.3%. Similar gains are observed on USCIS-AAO across nearly all models. These results suggest that the LLM-guided extraction process substantially mitigates brittleness in fully translated pipelines. Interestingly, for Airline, the gap between Prolog and PercepLog is relatively small. This is likely because the Airline rules are already translated with high accuracy by most models, leaving less room for recovery through query-driven extraction. In contrast, domains with more compositional logical dependencies benefit more strongly from PercepLog’s incremental reasoning strategy.

Ground-Truth Rule Setting. This setting ablates the impact of incorrect rules and shows the quality of query-driven fact extraction compared to

full translation of cases. Under this setting, PercepLog achieves consistently high performance while remaining competitive with or outperforming standard Prolog execution across nearly all domains and backbone models. For example, GPT-4.1 improves from 25.7% to 76.0% on SARA Numeric and from 73.6% to 80.8% on Housing, while Claude Opus 4.6 improves from 65.9% to 75.6% on Housing. For Airline and USCIS-AAO, in some cases, both approaches achieve similarly high performance once accurate symbolic rules are available. These results demonstrate that even when symbolic rules are correct, exhaustive fact extraction remains a major bottleneck for traditional formal pipelines. PercepLog alleviates this issue by replacing global case formalization with demand-driven extraction during proof search. This setting is applicable in realistic scenarios where formal rule libraries can be curated and maintained with human-in-the-loop because legal rules evolve relatively slowly. In contrast, legal cases arrive continuously and at scale, making robust query-driven extraction from noisy case documents the primary practical challenge.

Robustness Across Backbone Models. Another important observation is that PercepLog consistently improves performance across OpenAI, Anthropic, and Gemini models, suggesting that the gains primarily stem from the reasoning framework itself rather than from a specific model family. This indicates that explicit solver-guided reasoning provides a robust inductive bias for LLM reasoning.

4.4.1 Robustness to Distractor Facts

As motivated in Section 4.1, we hypothesize that PercepLog should perform even better when applied to more realistic, noisy cases in which much of the information may not be relevant to deciding a legal case. Table 3 reports results on the distractor-augmented SARA Binary split. PercepLog remains substantially more robust to noisy and irrelevant information than both direct QA and fully translated symbolic pipelines. In both generated-rules and ground-truth rule settings, PercepLog consistently outperforms standard Prolog execution across all backbone models, with particularly large gains under GT rules where accuracy approaches 90%. These results support our hypothesis that query-driven backward chaining is especially beneficial in realistic legal settings containing large amounts of distracting but topically related information.

4.4.2 Failure Modes

Failures of PercepLog may arise from *rule generation errors*, *fact extraction errors*, or *limitations in the current implementation* and the expressivity of the logical programs it can handle. According to our experiments, rule generation errors appear to be a more significant bottleneck in Housing and SARA Binary than in Airline and SARA Numeric, where the rules are more formalized and closely related to numerical computations.

Fact extraction errors, on the other hand, are more dominant in SARA Numeric due to two main challenges. First, the queries themselves are compositional and require aggregation over textual evidence, such as `findall` and summation operations over wages or exemptions. Second, some required facts are implicit rather than explicitly stated in the text. For example, the ground-truth knowledge base may specify that Sara is unmarried or does not file taxes jointly, even though these conditions are not directly mentioned in the case description. In such cases, the LLM-based extraction component may fail to produce the required predicates.

The third category of failures stems from limitations in the current PercepLog implementation, including the lack of optimized search procedures that may lead to timeouts, as well as incomplete support for certain Prolog constructs such as nested disjunctions inside `findall`. These limitations suggest that more mature implementations of LLM-augmented backward chaining could achieve substantially stronger performance in the future.

4.5 General Logical Reasoning Benchmarks

To demonstrate PercepLog’s generalization to abstract logical reasoning, we conducted some additional experiments on two commonly-used logical reasoning benchmarks outside the legal domain. Specifically, we evaluated PercepLog on the ProofWriter (Tafjord et al., 2021) and CLUTRR v2 (Sinha et al., 2019) datasets and compared it against recent neurosymbolic methods Symba and HBLR where applicable, while also reporting runtime. Table 4 (a) shows the comparison between the three frameworks on ProofWriter. The Symba paper reports 79% overall accuracy on the same test split using GPT-4 Turbo as the backbone LLM. However, GPT-4 Turbo has since been deprecated and is no longer available for use with PercepLog. Therefore, we reran all frameworks using GPT-4.1 instead. Overall, PercepLog improves absolute

(a) ProofWriter Depth-5						(b) CLUTRR v2			
Depth	N	HBLR	Symba	PercepLog		Method	Model	Acc. (%)	Time (m)
		4.1	4.1	5.4	4.1				
0	103	54.4	96.1	100.0	100.0	PercepLog	GPT-5.4	89.0	22.2
1	51	70.6	94.1	100.0	96.3	PercepLog	GPT-4.1	70.0	18.6
2	43	58.1	86.0	87.5	81.2	Symba	GPT-4.1	89.0	56.6
3	38	86.8	78.9	100.0	78.6	Symba	GPT-5.4	84.0	61.2
4	35	77.1	74.3	90.0	80.0				
5	30	56.7	43.3	97.7	94.4				
Overall	300	64.7	84.3	97.7	93.7				
Time (m)	300	~150	~55	~21	~17				

Table 4: Results on general logical reasoning benchmarks: (a) ProofWriter depth-5 results by reasoning depth on 300 examples; (b) CLUTRR v2 results on 100 examples. Accuracy is reported in percentage and runtime in minutes. HBLR (Li et al., 2026); Symba (Lee and Hwang, 2025).

accuracy by about 10% using GPT-4.1 backbone, while remaining largely independent of reasoning depth, whereas Symba’s performance drops dramatically at depth 5. At the same time, the runtime of the experiments is significantly reduced with PercepLog. Table 4 (b) demonstrates our results on CLUTRR v2, comparing to Symba. PercepLog achieves 89% accuracy with GPT-5.4, compared to 84% for SymBa, while reducing runtime from 56.6 to 22.2 minutes (2.5× faster). With GPT-4.1, PercepLog achieves lower accuracy indicating some dependence on the underlying LLM. PercepLog provides broad logical expressiveness, supporting most Prolog functionalities, whereas the compared baselines support much more limited languages and are not directly applicable to our complex legal Prolog rules. We therefore evaluate them on the test sets supported by their formalisms and report results accordingly. HBLR did not originally report results on CLUTRR v2, reflecting similar limitations in benchmark coverage and expressivity, so we do not include it in this comparison.

The logical reasoning benchmarks proved easier than the legal benchmarks, as they follow simpler syntactic templates. These complementary experiments demonstrate that PercepLog significantly improves accuracy while substantially reducing runtime, highlighting its potential for complex real-world applications.

5 Conclusion

We presented PercepLog, a deductive reasoning framework for legal reasoning that combines classical symbolic inference with the flexibility of perception in large language models. By following structured reasoning procedures such as backward chaining while delegating the primitive reasoning

operations to LLMs, the framework enables interpretable step-by-step reasoning directly over raw legal text without requiring formalization of legal documents. Our query-driven reasoning paradigm incrementally extracts only information relevant to the current reasoning goal, avoiding exhaustive semantic parsing of long and noisy legal documents. This design improves scalability and robustness while preserving the advantages of symbolic reasoning. Experiments on DeonticBench across multiple legal domains demonstrate that our approach consistently outperforms prior methods under both ground-truth and automatically translated rule settings, highlighting the effectiveness of solver-guided reasoning compared to direct LLM reasoning and fully translation-based neurosymbolic pipelines. Overall, our results suggest that combining symbolic reasoning structures with LLM-based local reasoning offers a promising direction for scalable and reliable legal AI systems.

Acknowledgments

We sincerely thank Bloomberg’s CTO Office for its support of the Visiting Faculty Program, and David Rosenberg for his thorough review and guidance.

Limitations

PercepLog still depends on the quality of the underlying symbolic rule representations. Although query-driven extraction improves robustness to noisy and incomplete formalizations, incorrect or hallucinated rules can still propagate errors during reasoning. In addition, LLM extraction errors can propagate without checks or verification. The iterative LLM calls during backward chaining may also increase inference cost for deep proof trees

and large rule spaces. Furthermore, PercepLog-0 is currently a lightweight implementation of Prolog and does not yet support the full range of Prolog functionalities, optimized search and execution mechanisms available in mature symbolic solvers. Finally, our evaluation focuses on statutory reasoning benchmarks and may not fully capture the complexity of real-world legal reasoning involving conflicting statutes, temporal dependencies, and evolving legal interpretations. While our methods aim to improve robustness and verifiability, they are not intended for deployment in real-world legal decision-making without human oversight. **Use of LLMs:** LLMs were used to improve wording, perform grammar checks, and enhance the overall clarity of the text. Coding agents were also used to assist in implementing the experimental setup.

References

- Mohamad Abou Ali, Fadi Dornaika, and Jinan Charafeddine. 2025. [Agentic ai: a comprehensive survey of architectures, applications, and future directions](#). *Artificial Intelligence Review*, 59(1):11.
- Erik Arakelyan, Pasquale Minervini, Patrick Lewis, Pat Verga, and Isabelle Augenstein. 2025. [FLARE: Faithful logic-aided reasoning and exploration](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 23385–23403, Suzhou, China. Association for Computational Linguistics.
- Kevin D. Ashley. 2017. *Artificial Intelligence and Legal Analytics: New Tools for Law Practice in the Digital Age*. Cambridge University Press.
- Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. [LEGAL-BERT: The muppets straight out of law school](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.
- Matthew Dahl, Varun Magesh, Mirac Suzgun, and Daniel E Ho. 2024. [Large legal fictions: Profiling legal hallucinations in large language models](#). *Journal of Legal Analysis*, 16(1):64–93.
- Fatemeh Dehghani, Roya Dehghani, Yazdan Naderzadeh Ardebili, and Shahryar Rahnamayan. 2025. [Large language models in legal systems: A survey](#). *Humanities and Social Sciences Communications*, 12(1):1977.
- Guangyao Dou, Luis Brena, Akhil Deo, William Jurayj, Jingyu Zhang, Nils Holzenberger, and Benjamin Van Durme. 2026. [Deonticbench: A benchmark for reasoning over rules](#). Preprint, arXiv:2604.04443.
- Yu Feng, Nathaniel Weir, Kaj Bostrom, Sam Bayless, Darion Cassel, Sapana Chaudhary, Benjamin Kiesl-Reiter, and Huzefa Rangwala. 2026. [Vericot: Neuro-symbolic chain-of-thought validation via logical consistency checks](#). In *The Fourteenth International Conference on Learning Representations*.
- Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya Narayana, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N. Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, and 21 others. 2023. [Legal-bench: a collaboratively built benchmark for measuring legal reasoning in large language models](#). In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS ’23*, Red Hook, NY, USA. Curran Associates Inc.
- Nils Holzenberger, Andrew Blair-Stanek, and Benjamin Van Durme. 2020. [A dataset for statutory reasoning in tax law entailment and question answering](#). In *Proceedings of the 2020 Natural Language Processing (NLLP) Workshop*, pages 1–10, San Diego, US. CEUR-WS.org.
- Dongwei Jiang, Marcio Fonseca, and Shay Cohen. 2024. [LeanReasoner: Boosting complex logical reasoning with lean](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7497–7510, Mexico City, Mexico. Association for Computational Linguistics.
- William Jurayj, Nils Holzenberger, and Benjamin Van Durme. 2025. [Language models and logic programs for trustworthy tax reasoning](#). In *AAAI Conference on Artificial Intelligence*.
- Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. 2023. [LAMBADA: Backward chaining for automated reasoning in natural language](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6547–6568, Toronto, Canada. Association for Computational Linguistics.
- Parisa Kordjamshidi, Samer Aslan, Madhavan Seshadri, Leslie Barrett, and Enrico Santus. 2026. [Reasoners or translators? contamination-aware evaluation and neuro-symbolic robustness in tax law](#). Preprint, arXiv:2605.16052.
- Jinu Lee and Wonseok Hwang. 2025. [SymBa: Symbolic backward chaining for structured natural language reasoning](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2468–2484, Albuquerque, New Mexico. Association for Computational Linguistics.

- Qingchuan Li, Mingyue Cheng, Zirui Liu, Daoyu Wang, Yuting Zeng, and Tongxuan Liu. 2026. [From hypothesis to premises: Llm-based backward logical reasoning with selective symbolic translation](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(37):31671–31679.
- John W. Lloyd. 1987. *Foundations of Logic Programming*. Springer.
- Varun Magesh, Faiz Surani, Matthew Dahl, Mirac Suzgun, Christopher D. Manning, and Daniel E. Ho. 2025. Hallucination-free? assessing the reliability of leading ai legal research tools. *Journal of Empirical Legal Studies*, 22.
- L. Thorne McCarty. 1977. [Reflections on "taxman": An experiment in artificial intelligence and legal reasoning](#). *Harvard Law Review*, 90(5):837–893.
- Ha Thanh Nguyen, Wachara Fungwacharakorn, May Myo Zin, Randy Goebel, Francesca Toni, Kostas Stathis, and Ken Satoh. 2025. [Llms for legal reasoning: A unified framework and future perspectives](#). *Computer Law and Security Review*, 58:106165.
- Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. [Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.
- Karen McGregor Richmond, Satya M. Muddamsetty, Thomas Gammeltoft-Hansen, Henrik Palmer Olsen, and Thomas B. Moeslund. 2023. [Explainable ai and law: An evidential survey](#). *Digital Society*, 3(1):1.
- Albert Sadowski and Jaroslaw A. Chudziak. 2025. [On verifiable legal reasoning: A multi-agent framework with formalized knowledge representations](#). In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management, CIKM '25*, page 2535–2545, New York, NY, USA. Association for Computing Machinery.
- Giovanni Sartor. 2010. Legal reasoning: A cognitive approach to the law. In Enrico Pattaro, editor, *A Treatise of Legal Philosophy and General Jurisprudence*. Springer.
- Sanskar Sehgal and Yanhong A. Liu. 2024. [Logical lease litigation: Prolog and llms for rental law compliance in new york](#). In *Proceedings 40th International Conference on Logic Programming, ICLP 2024*, University of Texas at Dallas, Dallas Texas, USA, October 14-17 2024, EPTCS, pages 59–68.
- Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. 2019. [CLUTRR: A diagnostic benchmark for inductive reasoning from text](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515, Hong Kong, China. Association for Computational Linguistics.
- Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. [ProofWriter: Generating implications, proofs, and abductive statements over natural language](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.
- Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. [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)*, pages 13326–13365, Bangkok, Thailand. Association for Computational Linguistics.
- Tian Yang, Yunwen Li, Ziniu Li, Zhihang Lin, Ruoyu Sun, and Tian Ding. 2025. [Bridging formal language with chain-of-thought reasoning to geometry problem solving](#). *ArXiv*, abs/2508.09099.
- Xinxin You, Qixin Sun, Chenwei Yan, Xiao Zhang, Chen Ning, Xiangling Fu, Si Liu, Guoping Hu, Shijin Wang, Ji Wu, and Xien Liu. 2026. [FACT: Mitigating inconsistent hallucinations in LLMs via fact-driven alternating code-text training](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

A Formal Background

This appendix gives the first-order backward-chaining procedure used as the symbolic background for PercepLog. We first state the two standard notions used by the algorithm and then present the classical procedure in Algorithm 2.

Definition A.1 (Substitution). A substitution $\theta = \{X_1 \mapsto t_1, \dots, X_n \mapsto t_n\}$ is a finite mapping from variables to terms. Applying θ to a term t , written $t\theta$, simultaneously replaces each variable X_i in t by the corresponding term t_i . The composition $\theta \circ \sigma$ is the substitution such that $t(\theta \circ \sigma) = (t\theta)\sigma$ for all terms t . The empty substitution ϵ satisfies $t\epsilon = t$ for all t .

Definition A.2 (Unification). Two terms s and t are unifiable if there exists a substitution θ such that $s\theta = t\theta$. Such a θ is called a unifier of s and t . A unifier θ is a most general unifier (MGU) if for every other unifier σ there exists a substitution γ such that $\sigma = \theta \circ \gamma$. When it exists, the MGU is unique up to variable renaming. We write $Unify(s, t) = \theta$ if θ is the MGU of s and t , and $Unify(s, t) = fail$ if no unifier exists.

In LLM-augmented backward chaining, a prompt such as Figure 3 returns candidate groundings for the current goal. These candidates are then passed to unification inside the backward-chaining loop.

Algorithm 2 Backward Chaining (First-Order)

Require: Goal g , rule base $\mathcal{R}$, fact base $\mathcal{F}$, substitution θ

Ensure: Set of substitutions under which g is derivable

```

1: function BC( $g, \theta$ )
2:   for all facts  $f \in \mathcal{F}$  do
3:      $\theta' \leftarrow \text{UNIFY}(g, f, \theta)$ 
4:     if  $\theta' \neq fail$  then
5:       yield  $\theta'$ 
6:   for all rules  $(h \leftarrow b_1, \dots, b_k) \in \mathcal{R}$  do
7:      $\theta' \leftarrow \text{UNIFY}(g, h, \theta)$ 
8:     if  $\theta' \neq fail$  then
9:       for all  $\sigma \in \text{BCBODY}(b_1, \dots, b_k, \theta')$  do
10:        yield  $\sigma$ 

11: function BCBODY( $b_1, \dots, b_k, \theta$ )
12:   if  $k = 0$  then
13:     yield  $\theta$ 
14:   return
15:   for all  $\theta' \in \text{BC}(b_1\theta, \theta)$  do
16:     for all  $\sigma \in \text{BCBODY}(b_2, \dots, b_k, \theta')$  do
17:       yield  $\sigma$ 

```

B Distractor Augmentation Example

Each augmented case adds 18–26 distractor Prolog facts and corresponding natural-language sentences

drawn from other SARA Binary cases. Distractors use predicates absent from the ground-truth rules, so SWI-Prolog ignores them. Rules and queries remain unchanged. Figure 6 shows a representative case before and after augmentation. In this example, the ground-truth rules reference only five predicates: `marital_status/2`, `married_to/2`, `no_gross_income/1`, `not_joint_return/2`, and `not_dependent_of_another/1`. All distractor facts use predicates outside this set.

C Additional Results

Table 5 reports additional reference baselines across the five DeonticBench domains. These runs supplement the main results by including model variants. A subset of these models was used as PercepLog backbones in the primary comparison in the main content of the paper. Moreover Table 6 reports more experiments with open-weight models.

D Examples Logical Reasoning Trees

For further qualitative analysis, we include examples of the logical reasoning trees generated by PercepLog and Prolog across several problems and domains in Figures 7–16. Some of the figures are partially drawn due to their large sizes.

Case: s151_b_pos **Label:** Entailment **Question:** Alice can receive an exemption for Bob under §151(b) for the year 2015.

Original case narrative (2 sentences)

Alice and Bob have been married since 2 Feb 2015. Bob has no income for 2015.

Original Prolog facts (5 facts)

```
marital_status(alice, married).
married_to(alice, bob).
no_gross_income(bob).
not_joint_return(alice, bob).
not_dependent_of_another(bob).
```

Ground-truth rules (unchanged in both conditions)

```
spouse_exemption_allowed(Taxpayer, Spouse) :-
    marital_status(Taxpayer, married),
    not_joint_return(Taxpayer, Spouse),
    no_gross_income(Spouse),
    not_dependent_of_another(Spouse).
alice_eligibility_for_bob_exemption :-
    married_to(alice, bob),
    spouse_exemption_allowed(alice, bob).
```

Augmented case narrative (18 sentences, 16 distractors)

Alice has paid wages of \$3200 to Bob for domestic service done in her home from Feb 1st, 2017 to Sep 2nd, 2017. In 2015, Alice's income was \$312. Mexico has a tax agreement with the US as a contiguous country. Bob had no income in 2015. Alice and Bob got married on Feb 3rd, 1992. Alice died on July 9th, 2014. **Bob has no income for 2015. Alice and Bob have been married since 2 Feb 2015.** Alice claims her tax is \$3538. Alice and Bob have been married since Feb 3rd, 2017, and they file a joint return for 2017. The exemption amount for Alice under §151(d) for the year 2015 was \$2000. Alice paid Bob \$12980 because she had to terminate their contract due to Bob's disability. Bob is a taxpayer. The standard deduction for a married individual filing jointly or a surviving spouse is \$6000. Alice has paid \$3200 to Bob for agricultural labor done from Feb 1st, 2017 to Sep 2nd, 2017. Alice is not married, is not a surviving spouse, and is not a head of household in 2017. Alice's taxable income for the year 2017 is \$236422. The standard deduction for a head of household is \$4400.

Augmented Prolog facts (28 facts, 23 distractors)

```
married(alice,bob,date(1992,2,3)).
taxable_income(alice, 236422).
standard_deduction(joint_or_surviving_spouse, 6000).
marital_status(alice, married). % *original
death_year(alice, 2014).
not_head_of_household(alice).
standard_deduction(head_of_household, 4400).
person(bob).
claim_tax(3538).
income(bob, 0).
married(alice, bob, 1992).
not_married(alice).
exemption_amount(2015, 2000).
not_dependent_of_another(bob). % *original
has_agreement_with_us_contiguous_country(mexico).
not_joint_return(alice, bob). % *original
performs_agricultural_labor(bob, alice, 2017).
domestic_service_in_private_home(alice, 2017).
spouse(bob, alice).
alice_files_joint_return(true).
married_to(alice, bob). % *original
no_gross_income(bob). % *original
alice_married(true).
payment(payment_of_12980).
not_surviving_spouse(alice).
alice_income(2015, 312).
paid_wages_to_alice_bob(2017, 3200).
agricultural_cash_wages_paid(alice, 2017, 3200).
```

Figure 6: Example of distractor augmentation for one SARA Binary case. The left column shows the original case; the right column shows the augmented version. The two original sentences are bolded in the augmented narrative, and original facts are marked with comments. The ground-truth rules are unchanged.

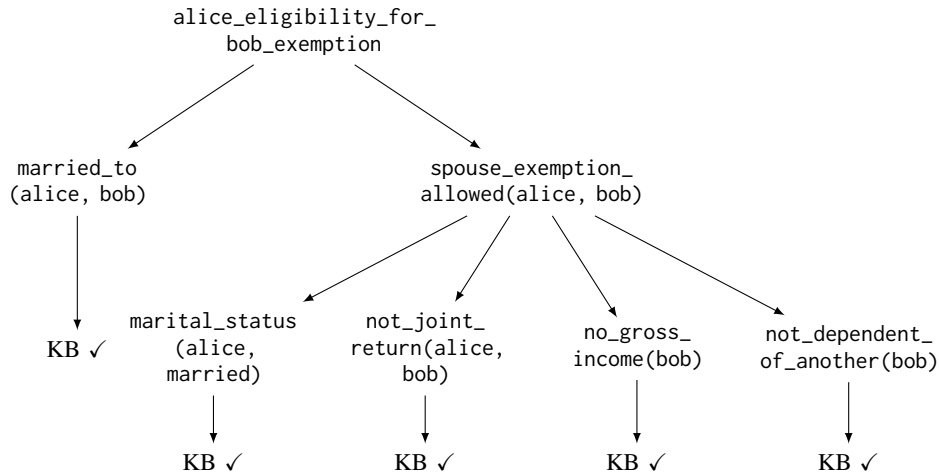


Figure 7: Proof tree of Prolog for the claim in Figure 1. The root goal is resolved by rule expansion into two subgoals. spouse_exemption_allowed is further expanded into four conditions. Each leaf is resolved by going back and forth between goals, formal KB facts and formal rules, yielding Entailment.

Model	Setting	Accuracy (%)				
		SARA Num.	Airline	SARA Bin.	USCIS-AAO	Housing
GPT-4.1	Few-Shot	22.0 ± 2.6	63.9 ± 2.8	44.9 ± 6.8	55.1 ± 3.5	40.6 ± 1.5
	Zero-Shot	8.9 ± 3.2	0.8 ± 0.8	35.8 ± 4.1	54.1 ± 3.5	24.9 ± 1.5
	Direct	18.3 ± 5.4	5.8 ± 3.4	33.3 ± 5.1	64.3 ± 0.0	20.5 ± 1.1
O3	Few-Shot	24.4 ± 1.3	91.2 ± 1.0	39.3 ± 2.4	56.7 ± 5.2	45.3 ± 1.9
	Zero-Shot	47.1 ± 4.6	19.2 ± 4.1	37.0 ± 5.2	60.2 ± 3.7	19.6 ± 1.5
	Direct	37.7 ± 6.3	38.8 ± 5.9	23.3 ± 0.0	62.1 ± 2.4	18.2 ± 5.0
GPT-5.1	Few-Shot	10.5 ± 6.1	68.0 ± 3.2	46.6 ± 3.7	62.6 ± 3.5	26.5 ± 1.7
	Zero-Shot	17.0 ± 2.9	1.3 ± 1.3	45.0 ± 6.6	71.5 ± 3.5	22.6 ± 1.3
	Direct	12.6 ± 6.4	2.2 ± 1.3	22.0 ± 3.7	77.9 ± 3.7	22.1 ± 2.6
GPT-5.2	Few-Shot	14.5 ± 2.6	75.5 ± 5.1	34.8 ± 4.2	40.3 ± 5.3	25.4 ± 2.9
	Zero-Shot	17.7 ± 2.0	2.5 ± 0.8	30.3 ± 2.8	46.6 ± 7.3	21.4 ± 1.6
	Direct	18.9 ± 9.2	1.8 ± 0.8	19.3 ± 3.5	73.6 ± 2.4	25.6 ± 1.1
GPT-5.4	Few-Shot	15.3 ± 2.3	86.2 ± 1.6	35.6 ± 2.3	59.0 ± 2.9	28.9 ± 2.0
	Zero-Shot	19.1 ± 3.5	2.6 ± 0.6	29.6 ± 2.6	61.3 ± 2.9	22.8 ± 4.3
	Direct	32.0 ± 5.8	3.2 ± 1.8	60.0 ± 7.2	77.1 ± 2.4	22.8 ± 3.1
GPT-5.4 High	Few-Shot	64.7 ± 4.4	94.9 ± 0.3	59.7 ± 2.3	66.1 ± 2.9	23.7 ± 1.8
	Zero-Shot	56.9 ± 7.6	50.9 ± 14.8	58.2 ± 6.7	67.5 ± 2.0	20.8 ± 2.3
	Direct	60.0 ± 6.6	88.2 ± 5.9	77.3 ± 5.4	75.0 ± 4.4	19.2 ± 3.7
Claude Opus 4.6	Few-Shot	52.5 ± 3.8	73.8 ± 8.4	32.6 ± 3.3	55.5 ± 3.3	25.1 ± 2.2
	Zero-Shot	64.9 ± 1.5	12.1 ± 1.7	49.1 ± 2.9	66.4 ± 4.8	22.8 ± 1.2
	Direct	56.6 ± 5.3	21.7 ± 4.2	43.3 ± 27.8	75.0 ± 0.0	27.4 ± 2.1
Claude Sonnet 4.5	Few-Shot	31.4 ± 3.0	82.5 ± 1.0	46.1 ± 4.4	57.5 ± 3.8	25.1 ± 0.4
	Zero-Shot	24.3 ± 4.2	5.0 ± 1.2	52.7 ± 5.5	60.8 ± 1.7	28.0 ± 2.6
	Direct	45.1 ± 5.8	4.5 ± 3.7	37.3 ± 3.5	60.7 ± 0.0	42.1 ± 0.7
Gemini 2.5 Flash	Few-Shot	27.4 ± 1.9	49.2 ± 1.1	45.0 ± 4.7	42.5 ± 3.0	29.5 ± 1.9
	Zero-Shot	15.9 ± 3.7	5.1 ± 0.7	35.9 ± 1.8	41.0 ± 4.4	25.0 ± 1.3
	Direct	45.7 ± 5.0	20.8 ± 3.2	23.3 ± 0.0	50.0 ± 5.4	22.3 ± 2.4

Table 5: Additional baseline accuracy results across five DeonticBench domains. Values are mean accuracy with 95% confidence intervals over five runs.

Note: Few-Shot and Zero-Shot denote settings in which the full input is translated to Prolog and solved with a symbolic solver. These results are included as reference baselines; the main paper focuses on the five models used as PercepLog backbones.

Model	Setting	SARA Num.	Airline	SARA Bin.	USCIS-AAO	Housing
gpt-oss-120b	Direct	20.6 ^{+9.8} _{-9.8}	9.0 ^{+3.0} _{-3.0}	75.3 ^{+3.7} _{-3.7}	50.0 ^{+3.1} _{-3.1}	15.4 ^{+1.9} _{-1.9}
	Prolog	13.3 ^{+2.4} _{-2.4}	12.1 ^{+1.5} _{-1.5}	50.9 ^{+1.1} _{-1.1}	52.4 ^{+4.2} _{-4.2}	31.3 ^{+3.7} _{-3.7}
	PercepLog	15.4 ^{+3.2} _{-3.2}	80.2 ^{+5.4} _{-5.4}	56.0 ^{+9.9} _{-9.9}	60.7 ^{+6.3} _{-6.3}	59.5 ^{+2.7} _{-2.7}
	Prolog (GT Rules)	29.1 ^{+1.6} _{-1.6}	43.2 ^{+2.1} _{-2.1}	80.0	75.7 ^{+2.0} _{-2.0}	60.3 ^{+3.6} _{-3.6}
	PercepLog (GT Rules)	50.4 ^{+6.2} _{-6.2}	77.2 ^{+5.3} _{-5.3}	70.0 ^{+2.9} _{-2.9}	77.9 ^{+4.9} _{-4.9}	64.6 ^{+3.7} _{-3.7}
Qwen3.5-35B-A3B	Direct	34.3 ^{+8.3} _{-8.3}	0.5 ^{+0.8} _{-0.8}	78.0 ^{+2.3} _{-2.3}	56.4 ^{+4.9} _{-4.9}	26.7 ^{+4.7} _{-4.7}
	Prolog	6.7 ^{+1.3} _{-1.3}	32.4 ^{+4.5} _{-4.5}	34.4 ^{+8.8} _{-8.8}	34.0 ^{+3.4} _{-3.4}	—
	PercepLog	8.6	41.2	63.3	53.6	41.0
	Prolog (GT Rules)	—	82.5	43.3	82.1	61.5
	PercepLog (GT Rules)	68.6	83.8	56.0 ^{+4.5} _{-4.5}	42.9	73.1

Table 6: PercepLog compared to baselines across legal reasoning domains with various open weights LLM backbones.

Note: Qwen’s Housing/Prolog and SARA Numeric/Prolog (GT Rules) cells are — because every run exceeded the 50% abstention threshold and was excluded.

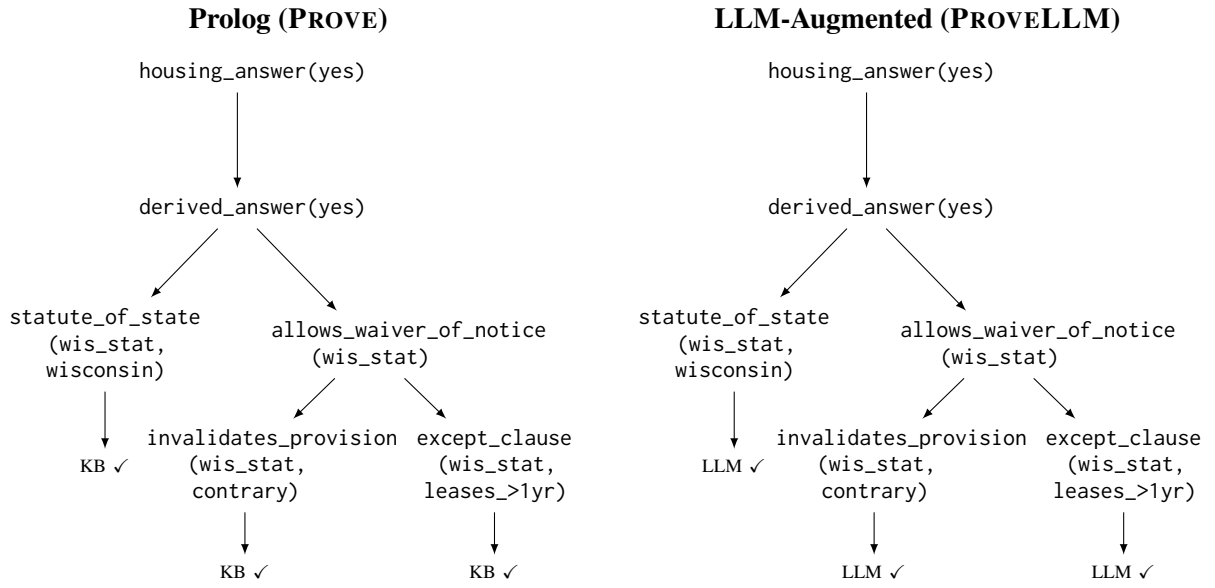


Figure 8: Proof trees for the Housing case (WIS. STAT. §704.17(5)(A)). Both trees follow the same backward chaining path: `housing_answer` → `derived_answer` → `allows_waiver_of_notice` → two statute property checks. In PROVE, leaves are resolved from pre-loaded KB facts. In PROVELLM, leaves are resolved by querying the LLM against the statute text at runtime.

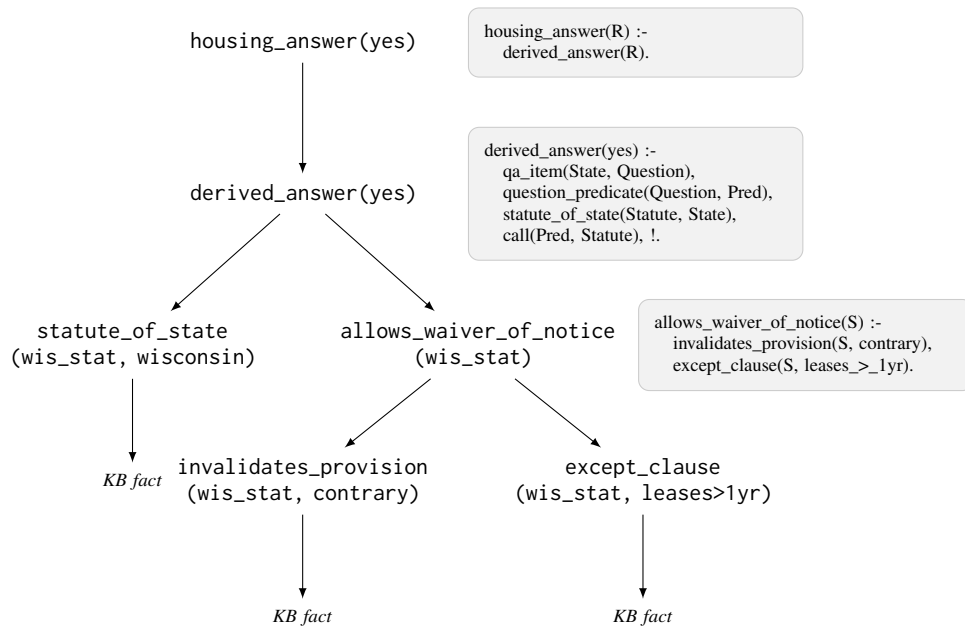


Figure 9: Proof tree for the Housing case (WIS. STAT. §704.17(5)(A)) with the rule applied at each level. Leaf nodes are ground facts resolved from the KB (Prolog) or extracted by the LLM at runtime (`prove_llm`).

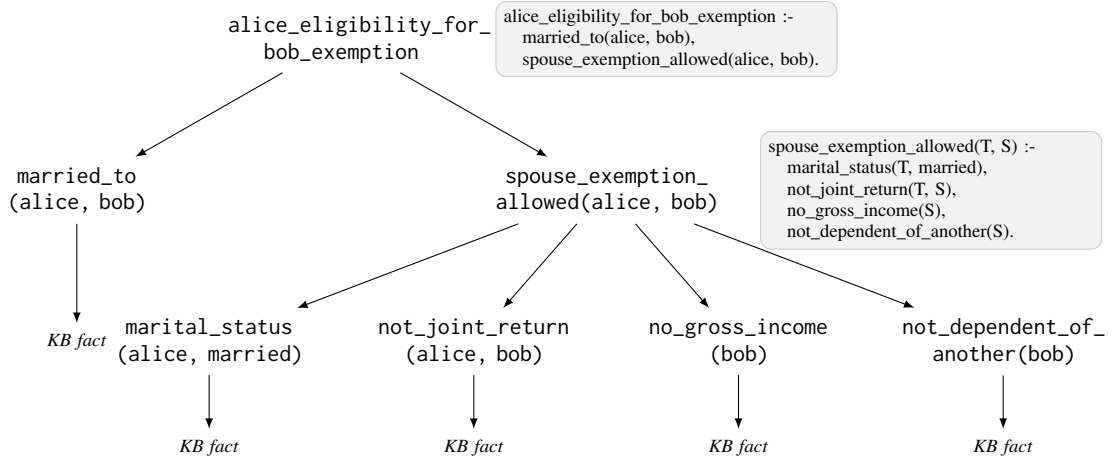


Figure 10: Proof tree for the SARA Binary case (IRC §151(b)) with the rule applied at each level. Leaf nodes are ground facts resolved from the KB (PROVE) or extracted by the LLM from the case text at runtime (PROVELLM).

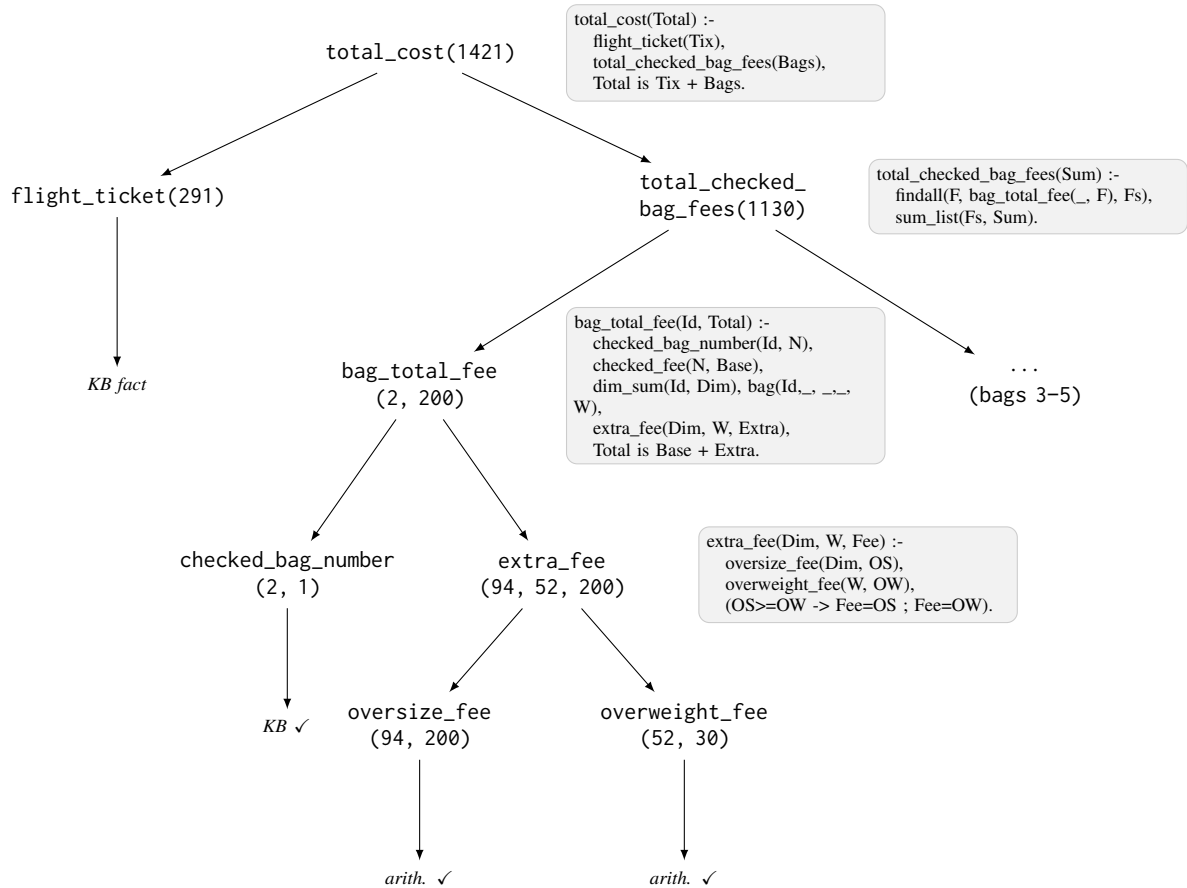


Figure 11: Proof tree for the Airline case (Emily, Ottawa to Dallas, \$1421 total). The backward chainer computes the total cost by recursively proving fees for each checked bag. Leaf nodes marked “KB fact” are extracted directly from the case text (bag dimensions, weights, flight ticket). Arithmetic leaves are resolved by the built-in is evaluator.

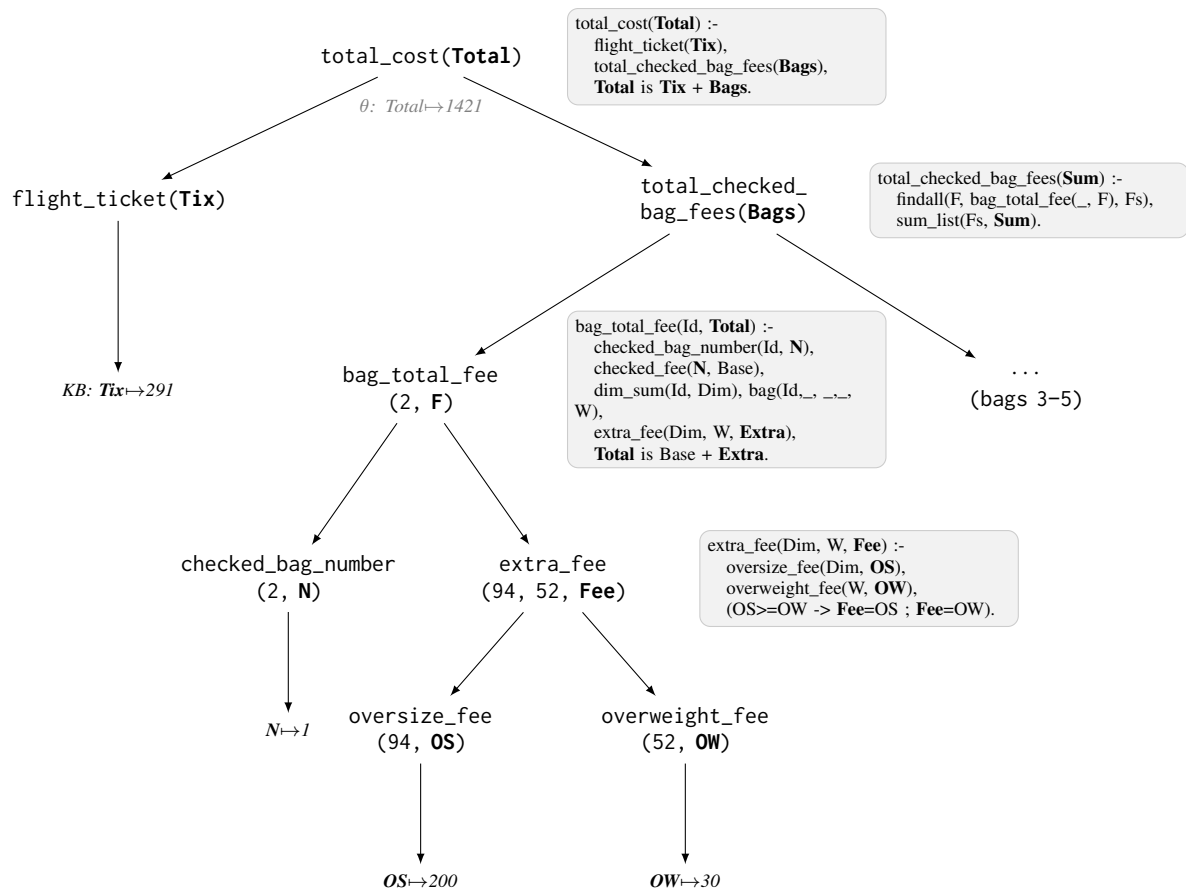


Figure 12: Proof tree for the Airline case (Emily, Ottawa to Dallas). Starting from the goal `total_cost(Total)` with an unbound variable, the backward chainer propagates substitutions bottom-up: leaf nodes bind **Tix** $\mapsto 291$, **OS** $\mapsto 200$, **OW** $\mapsto 30$, yielding **Fee** $\mapsto 200$, then **Bags** $\mapsto 1130$, and finally **Total** $\mapsto 1421$.

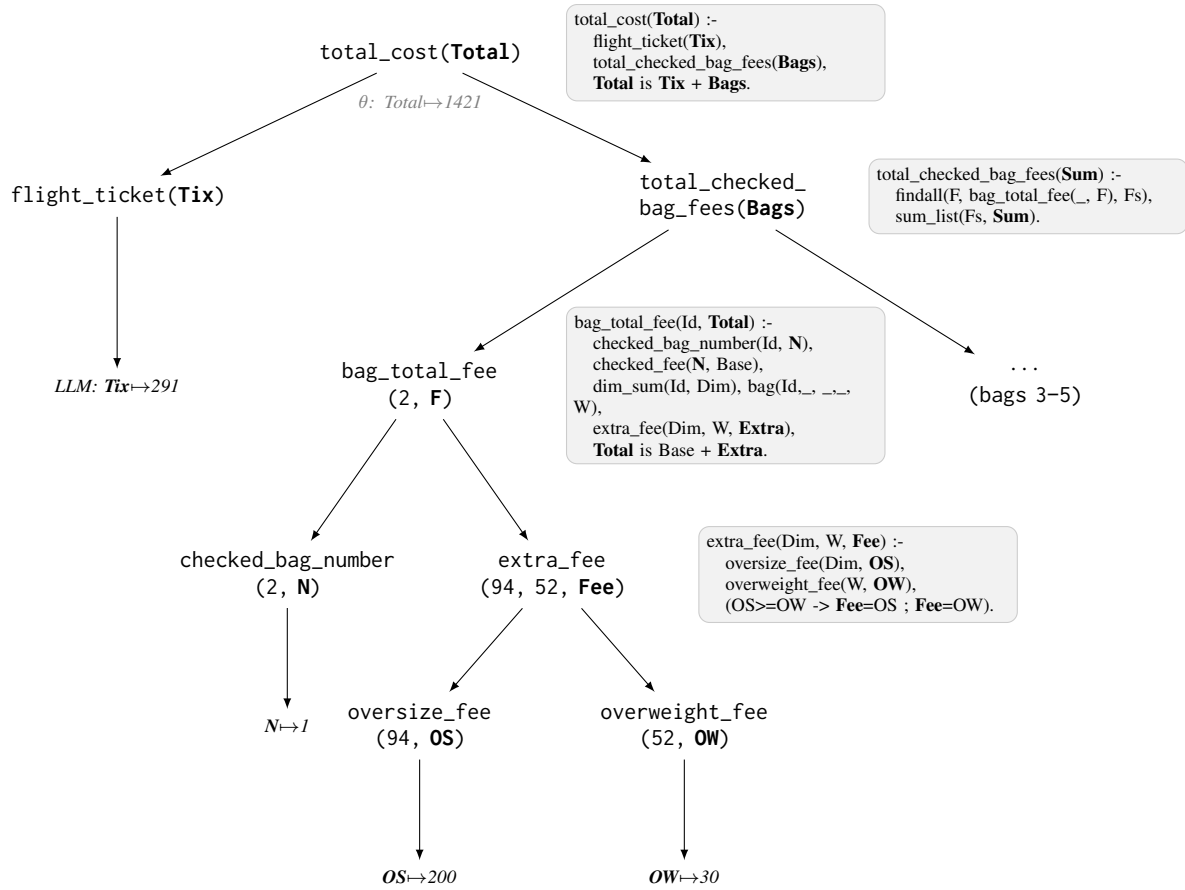


Figure 13: Proof tree for PROVELLM on the same Airline case. The tree structure is identical to PROVE (Figure 11), but leaf nodes marked “LLM” are resolved by querying the LLM against the case text at runtime rather than from a pre-loaded KB. For `flight_ticket(Tix)`, the LLM extracts \$291 from the sentence “Emily’s flight ticket is \$291.” Computed nodes (fee thresholds, arithmetic) remain unchanged.

PROLOG (WRONG: Dismissed)

eligibility_met
 $\downarrow$
 meets_marital_requirement
 $\downarrow$
 petitioner_unmarried

FAIL: fact missing from KB

PERCEPLOG (CORRECT: Accepted)

eligibility_met
 $\downarrow$
 meets_marital_requirement
 $\downarrow$
 petitioner_unmarried

LLM extracts from text:

Case: The petitioner seeks classification as a special immigrant juvenile. According to 2022 New York Family Court orders, the court appointed his biological mother as his guardian and found reunification with his father not viable due to neglect and abandonment.

Missing fact: `petitioner_unmarried`.

Rule: `meets_marital_requirement :- petitioner_unmarried.`

LLM prompt: “Does the scenario support `petitioner_unmarried`?”

LLM response: `[[[]]]` (yes — SIJ requires petitioner to be unmarried)

Figure 14: USCIS case where PROVE fails due to a missing fact in the predicted KB, while PROVELLM recovers by querying the LLM at runtime. Both use the same GT rules and predicted facts. The LLM correctly infers `petitioner_unmarried` from the SIJ context.

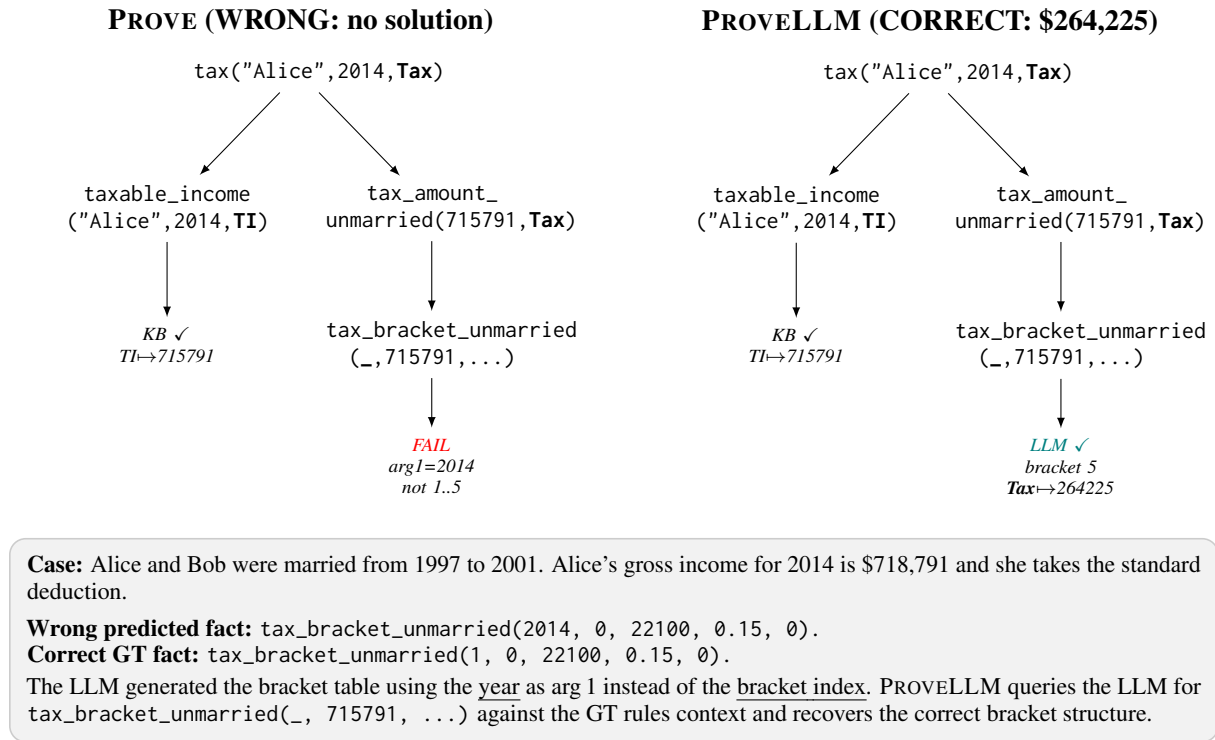


Figure 15: SARA Numeric case (IRC §1) where the predicted KB has a wrong argument structure in tax_bracket_unmarried. PROVE fails because no bracket matches index 5. PROVELLM queries the LLM which reads the GT rules context and returns the correctly indexed bracket, yielding the right tax of \$264,225.

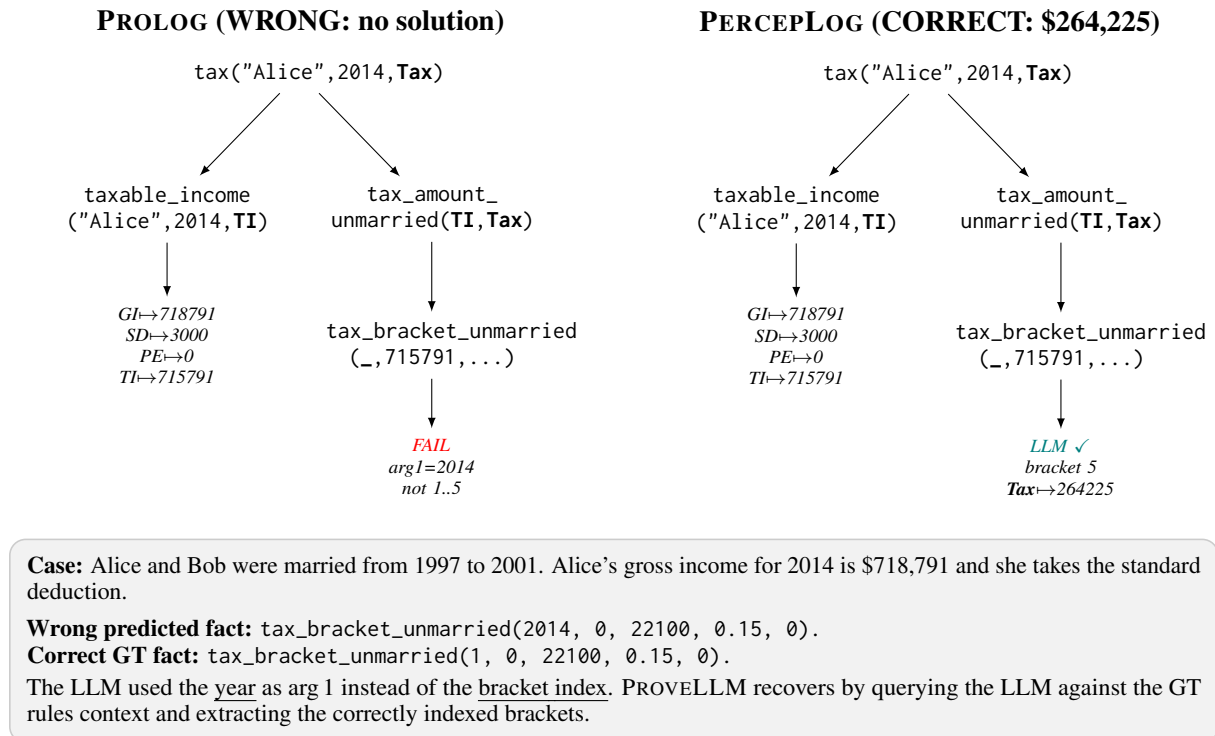


Figure 16: SARA Numeric case (IRC §1) where the predicted KB has a wrong argument structure in tax_bracket_unmarried. TI is computed as GI (\$718,791) minus standard deduction (\$3,000) minus personal exemption (\$0, fully phased out). PROVE fails because no bracket matches. PROVELLM recovers the correct bracket index, yielding \$264,225.