Legal-Proofing LLMs: Investigating Legal Applications of Prover Verification Models

REDACTED
REDACTED
REDACTED

Abstract

Large Language Models (LLMs) have shown remarkable results on legal tasks, but their reasoning remains trapped inside the 'black box,' raising credibility concerns in high-stakes domains like the legal sphere. While broad research on obtaining better results from LLMs in legal applications is ongoing, a lack of a verifiable component to these models leaves the precision-based legal industry hesitant to accept models that may be unreliable, even to a small degree. This paper explores a novel approach that bridges natural-language legal reasoning with formal verification by leveraging *prover LLMs*, inspired by recent successes in automated theorem proving. Proof-verification frameworks such as Apple's Hilbert and DeepSeek-Prover have demonstrated that combining informal reasoning with formal proof checking can dramatically improve reliability in mathematical problem solving: [Varambally et al. 2025] [Ren et al. 2025] I propose an adaptation of this paradigm to legal reasoning. I outline a dual LLM architecture comprising a general-purpose legal "reasoner" and a formal logic "prover," augmented by a verifier and legal knowledge context. This system can translate legal problems into a form amenable to formal proof, verify each step for correctness, and thereby produce interpretable, verifiable legal conclusions. I describe the design of such a system, situate it in the context of prior legal AI efforts, and present initial experiments showing that a prover-augmented model achieves higher accuracy on certain legal benchmarks than standard LLMs. These results indicate that applying formal theorem-proving techniques to legal reasoning is a promising direction to increase the rigor and trustworthiness of AI in the legal domain.

CCS Concepts

• **Applied computing** → **Law**; • **Computing methodologies** → **Natural language processing**; *Machine learning*; *Artificial intelligence*; Modeling and simulation.

Keywords

legal reasoning, theorem proving, verification, large language models, prover LLMs

ICAIL '26, City, Country

ACM ISBN 978-1-4503-XXXX-X/26/XX
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
REDACTED. 2026. Legal-Proofing LLMs: Investigating Legal Applications of Prover Verification Models. In *Proceedings of International Conference on Artifical Intelligence and Law (ICAIL '26).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Modern legal practice involves complex reasoning over statutes, regulations, and precedents expressed in natural language. The prospect of AI systems that can reason "like a lawyer" holds great promise for improving access to justice and legal decision support. Large Language Models have recently been applied to legal tasks with some success ([Hu et al. 2025], [Kant et al. 2025], [Guha et al. 2023]), yet significant challenges remain. Even state-of-the-art LLMs tend to falter on advanced legal reasoning problems, struggling with domain-specific knowledge, multi-step logical deductions, and accuracy in application of rules ([Guha et al. 2023] [Joshi et al. 2024] [Chlapanis et al. 2024] [Fei et al. 2024]). Critically, their outputs lack verifiability: an LLM might produce a confident-sounding legal conclusion, but without an explicit proof or logical justification, it is difficult to trust such an answer in real courtroom settings. This absence of transparency and formal correctness limits the credibility of black-box LLMs for broader application in legal use, where precision can cost millions.

Academic research into how to bridge this gap in legal reasoning has not given up on this endeavor, however. Attempts at formalizing legal logic into computer-parse-able syllogisms have been attempted for decades, and some standard practices have been found. An early example is the representation of the *British Nationality Act* as a Prolog logic program by Sergot et al. [1986]. This seminal work demonstrated that entire statutes can be encoded as formal rules, enabling automated inference that is provably sound and explainable. Such logic-based approaches offer clear advantages in terms of transparency and the ability to explain and modify laws. However, manual formalization of law is labor-intensive and these systems historically lacked the flexibility and linguistic capabilities of modern LLMs.

Meanwhile, in the field of automated theorem proving, a new generation of prover LLMs has emerged. Models like Hilbert [Varambally et al. 2025] and DeepSeek-Prover [Ren et al. 2025] have achieved state-of-the-art results on challenging mathematical benchmarks by combining natural language reasoning with formal verification. Hilbert, for example, solves 99.2% of problems on the MiniF2F math benchmark and 70.0% of Putnam competition problems, dramatically outperforming prior methods by integrating an LLM "reasoner" with a Lean theorem prover and recursive proof checking. DeepSeek-Prover-V2 (heretofor DSP for brevity) similarly leverages a two-phase reasoning (informal and formal) to set new

records on formal math datasets. These successes suggest that a hybrid approach, using the creative problem-solving ability of LLMs together with the rigor of formal proof systems, can significantly improve the reliability of AI reasoning.

I posit that such prover-augmented LLM frameworks can be transformative for legal reasoning tasks, an area where both natural language understanding and logical consistency are paramount. To date, however, there has been no systematic application of formal theorem-proving LLMs in the legal domain. In this paper, we take a first step toward bridging that gap. I outline a multi-tiered legal reasoning agent inspired by Hilbert and DSP, consisting of a general-purpose legal reasoning model and a specialized prover model that work in tandem. By encoding legal problems into formal representation and verifying each deductive step, such a system can ensure that its conclusions follow rigorously from given laws and facts. This approach could dramatically reduce hallucinations and logical errors, providing the kind of step-by-step justification a human judge or attorney would expect. It also offers a pathway to integrate vast natural-language legal corpora (cases, statutes) with formal reasoning engines, potentially yielding AI that not only *reads* legal text but also *proves* legal conclusions from it.

In the following sections, I survey related work at the intersection of AI and law, delineate the design of our proposed prover-augmented legal reasoning model, and present preliminary experiments. I evaluate a proof-of-concept using DSP on several LegalBench tasks to gauge the viability of this approach [Guha et al. 2023]. My results indicate that even without domain-specific tuning, the prover-based model can outperform a general LLM on many legal reasoning problems, especially those requiring strict logical inference. Finally, I will discuss the broader implications of formal proof-guided LLMs for legal AI, highlighting how this paradigm could enhance the transparency, accuracy, and ultimately the credibility of automated legal reasoning systems.

2 Related Work

Before the rise of LLM research, legal AI researchers pursued formal systems that could represent statutes and apply them transparently. A canonical example is Sergot et al. [1986], which translated large portions of the British Nationality Act into a Prolog (logic-verifier programming language) program. They proved that legal rules can be executed mechanically and that a logic-based encoding can preserve the structure of legislation closely enough to be auditable and maintainable, while also producing step-by-step explanations of derived conclusions. In parallel, case-based reasoning systems (e.g., Ashley's *HYPO* [Ashley 1988]) modeled legal argument through a comparison of precedents, offering another structured route to interpretable conclusions.

Yet building and updating hand-crafted representations of the law is expensive, as the nuanced, evolving character of legal language complicates comprehensive and precise coverage. This lack of scalability motivated the shift toward statistical NLP and machine learning, which can absorb large legal corpora without requiring explicit rule-by-rule encoding. The resulting tradeoff, however, is familiar: learned models may be powerful, but they often struggle to provide the kind of explicit, verifiable reasoning that lawyers demand.

The recent wave of legal benchmarks has made this tradeoff measurable. Guha et al. [2023] introduced LegalBench, a large suite of expert-authored English tasks spanning statutory reasoning, case analysis, and legal text understanding. Their evaluations show that strong general-purpose models can perform well on many subtasks, but also leave substantial headroom on multi-step reasoning and difficult application questions. Complementing this, Fei et al. [2024] proposed LawBench for Chinese law and emphasized a civil-law setting where applying codified rules is central; their results likewise suggest that higher-order application remains challenging even for top models. Across benchmarks and jurisdictions, a consistent picture emerges: LLMs handle surface-level understanding and some forms of retrieval effectively, but they remain untrustworthy where precise rule application, exception handling, and logically consistent inference are required. What is missing is a mechanism that can *enforce* correctness on a model, and outline where its logic first falters.

In response, a growing literature explores hybrid methods that combine neural language understanding with logical constraints. Kant et al. [2025] offers a representative view: it argues that trustworthy legal AI requires repeatability, interpretability, and verification, and it surveys methods that structure model reasoning through explicit logical decomposition and rule-guided pipelines. One prominent theme is to break complex legal questions into smaller logical sub-queries and recombine them, aiming to reduce leaps of inference and improve consistency. Another theme is controlled or semi-structured representations, such as domain-specific controlled natural languages for contracts, that preserve legal semantics while making downstream reasoning more systematic. A third theme couples LLM generation with logic engines like Prolog or Lean 4, using the symbolic component as a validator that can catch contradictions, missing premises, or unsupported inferences.

These approaches all share an implicit design philosophy: legal reasoning improves when we impose *structure* and *checks* on model outputs. Our proposal adopts the same philosophy but pushes it further by asking whether legal reasoning can be grounded in formal proof construction rather than only rule-guided prompting or post-hoc validation, so that the model's reasoning can be verifiable step-by-step.

2.1 Prover LLMs in mathematics as a template for "legal proofing"

The strongest evidence that proof-guided generation can transform legal LLM reliability comes from recent advances in formal theorem proving. DeepSeek-Prover [Ren et al. 2025] separates the problem solving process into two modes of thought: Chain-of-Thought (CoT) and non-Chain-of-Thought (non-CoT). Chain-of-Thought mode consists of an informal reasoning component which decomposes problems and proposes proof sketches. In non-CoT mode, a specialized prover component solves sub-goals in Lean 4 and a verifier checks correctness. Similarly, Hilbert [Varambally et al. 2025] frames theorem proving as an agentic workflow that orchestrates an informal reasoner, a formal prover, a verifier, and a retriever of relevant theorems. These systems matter for our purposes because they show a concrete route to narrowing the gap between

fluent natural-language reasoning and formally checkable inference: when verification is in the loop, the system can iteratively repair failures, decompose hard steps, and converge on proofs that are not merely persuasive but correct by construction.

Although these frameworks were developed for mathematics, their architectural lesson is domain-general: pair a model that can interpret and plan in natural language with a component that can express and verify reasoning in a formal system, and use verifier feedback to discipline the overall process. Our work adapts this template to law by asking whether we can build a 'Legal Hilbert' that retrieves relevant authorities (statutes, regulations, precedents or their formalizations), proposes a structured legal argument in natural language, and then encodes that argument into a formal representation where each step can be verified. If successful, this would address a central limitation surfaced by legal benchmarks: not only improving accuracy, but producing conclusions that are audit-able, defensible, and resistant to hallucination.

3 Proposed Framework: A Dual LLM for Legal Proofs

To my knowledge, there remains no end-to-end system that brings the recent prover-LLM paradigm into legal reasoning in a way that is both practically useful and formally verifiable. I therefore outline an idealized framework, what I will refer to as a *Legal-Proofing LLM* (LP LLM, for brevity), by borrowing the central architecture from mathematical theorem-proving systems and adapting it to a legal context.

3.1 System Architecture and Components

LP would be built around two tightly coupled models and two supporting modules. The first component would consist of a *legal reasoner* LLM which operates in natural language; it would read the user's question and fact pattern, identifies any legally-relevant issues, and sketches a structured route to an answer (for example, recognizing when analysis should proceed element-by-element, as in contract formation or a statutory test). This mirrors the role played by the informal reasoning component in theorem-proving agents, which produces high-level proof plans and decompositions that a formal system can attempt to discharge. The second component would be a *prover* LLM which would use a symbolic language (Lean-style proofs, logic programs, or another legal 'logic') so that the reasoning can be expressed as precise claims about rules and facts. It would focus on translating the reasoner's proposed steps into formal sub-goals and work to prove them in a logically-sound manner.

The remaining two modules enforce correctness and supply grounding. A *formal verifier* (e.g., a proof assistant or a logic inference engine) accepts or rejects proof steps, supplying concrete feedback when an attempt fails, which can be routed back to the models for correction and refinement. This verifier forces the system to confront missing premises, invalid implications, or misapplied rules. Finally, a *legal knowledge retriever* supplies legal reasoning 'authorities' such as relevant statutes, regulations, and cases. Conceptually, it plays the same role that theorem retrieval plays in systems like Hilbert: it turns a generic LLM's reasoning capability into inference specific to the legal field. In the long run, one could imagine building a law-analog of `mathlib`, seeded by classic formalizations like the *British Nationality Act* [Sergot et al. 1986] logic program and expanded through incremental formal encodings and extraction methods for tools like LP.

3.2 Algorithmic Approach

Operationally, LP proceeds as an iterative proof construction process. DeepSeek-Prover has the clearest algorithmic method that would serve well in legal analysis due to its recursive nature, breaking down each stance into smaller and smaller lemmas, until single components can be known to be true. The first mode that would be relevant is the Chain-of-Thought mode, using the 'reasoner' outlined in the previous sub-section; this mode would lay out a sketch of how to solve a legal task and recursively break down each problem into sub-problems to check, verifying that these sub-problems are feasible to solve before moving into the second non-CoT mode; this mode breaks down each recursed lemma into a feasible proof using the 'prover' LLM. This proof would be verifiable using a language like Lean 4.

Emphasizing legal formalism in this model is itself a practical design choice; the right encoding for a given task will largely depend on the target domain: highly rule-bound areas (eligibility determinations, compliance checks, benefit calculations) are often amenable to crisp encodings, while open-textured standards (reasonableness, proportionality, balancing tests) may resist full formalization. LP is therefore best understood as a formalized solver that hopes to help across the spectrum of tasks: even partial formalization could substantially improve reliability by shrinking the space where black-box hallucinations can hide.

4 Experiments

To test whether prover-style reasoning can transfer to law, I ran preliminary experiments using DeepSeek-Prover on a variety of LegalBench benchmarks. The goal was not to assume an ideal legal prover system, but to probe a concrete question: how well do formal prover models handle legal reasoning problems without any fine-tuning, and how do they compare to strong conventional LLM baselines that run on the same queries?

4.1 Experimental Setup

I evaluated **DeepSeek-Prover-V2 (671B)** [dee [n. d.]] as a proxy for a prover-augmented legal model. Although it is trained for Lean-based mathematical proof generation rather than legal doctrine, its emphasis on structured decomposition and stepwise correctness makes it a useful stand-in for my proposed paradigm. **GPT-4** served as a high-performing general-purpose model that demonstrated strong results LegalBench benchmarks, but lacks formal checking.

My benchmark tasks were drawn from **LegalBench** [Guha et al. 2023]. I focused on English-only tasks with objectively scorable outputs—multiple choice, binary decisions, or classification labels—so that performance could be measured automatically without subjective grading. I also prioritized tasks that require rule application or multi-step reasoning rather than simple retrieval, including statutory yes/no determinations, contract clause interpretation, and

other structured legal QA formats. After filtering, I tested the models on roughly 25 distinct tasks, sampling up to 100 queries per task when available.

All models were evaluated in a zero-shot (no-exposure) setting first. Each prompt consisted of the task description and a query, with an instruction to output only the final answer. DeepSeek-Prover required additional prompt shaping to cast legal questions as proof-like objectives (e.g., framing the task as proving whether the conclusion is "Yes" or "No" given stated facts and rules), while GPT-4 was prompted more directly. I then ran a small three-shot condition (three pre-exposures) to test whether showing even a limited number of exposures improved reliability on the prover model. Accuracy was computed by exact match against the benchmark. I limited my experiment to three pre-exposures to see how a 'bare' prover model operates on legal tasks.

5 Results

Aggregated results by reasoning category appear in Table 1, with per-task breakdowns in Table 2 (0-shot) and Table 3 (3-shot). Two caveats frame how these numbers should be read. First, I only tested a minimal exposure regime (n=0 and n=3 exposures), which likely leaves the prover-oriented model well below its ceiling. Second, DeepSeek-Prover was never fine-tuned to produce *legal* proofs; it is a math-trained prover used here as a proxy, so future legal finetuning would also likely increase performance.

Even under these constraints, the prover-based model (DSP) leads in the most logic-intensive categories. In Table 1, DSP achieves the highest accuracy all four of the reasoning types it attempted, with particularly large margins on issue spotting and rule selection, precisely the stages that resemble element-by-element proof checking in legal analysis. Interpretation, however, has the most promising results, as it was the category with the most number of tasks (23), and still outperformed the GPT baseline on all fronts. In the per-task analysis, DSP outperforms GPT across many baselines.

Where DSP underperforms, the errors are typically consistent with missing domain knowledge rather than incoherent logic. The trademark distinctiveness task (Abercrombie) is a clear example: GPT-4 likely benefits from memorized doctrinal categories and examples, while DSP, lacking legal fine-tuning, misapplies or misidentifies the relevant standard. Contract entailment tasks show a different dynamic: in zero-shot, DSP can appear behind in performance, but with only three exposures added (Table 3), it improves sharply on many NDA-style NLI subtasks, suggesting that small amounts of task-specific grounding help the prover model map legal language to the intended decision boundary. This supports a central claim of the paper: a prover-oriented reasoning engine has strong latent structure, but it needs legal semantics in order to reach full capacity.

Overall, the results should be read as a proof-of-concept rather than a final system comparison. I find it not insignificant that a math-trained prover model, evaluated with minimal prompting and without an actual legal proof environment, nonetheless performs strongly on rule-application-heavy legal tasks. To see potentially further significance, a combination of a legal-specific verifier, stronger legal grounding through retrieval or formal rule libraries, and training a prover explicitly for legal-logic lemmas would likely be a fruitful next step.

6 Discussion

This preliminary experiment suggests that using proof-style verification in legal reasoning LLMs can improve reliability in exactly the places where legal users care most: element-by-element rule application, improved consistency, and the ability to justify an output. My hope is that models forced into this disciplined workflow are less likely to produce plausible but unsupported rationales. This mirrors what has been observed in mathematical reasoning: without a verifier in the loop, even strong LLMs can drift into confident errors, whereas verification pressure grounds model reasoning.

Beyond raw accuracy, the central payoff is the ability to audit the prover model. A prover-augmented legal system can, in principle, expose its reasoning as a checkable chain: which rule was applied, what premises were required, where exceptions were considered, and why the conclusion follows. That quality addresses one of the main barriers to legal adoption – the black-box character of LLM outputs.

At the same time, my current evidence is preliminary and comes with clear constraints. The prover model I tested was not trained on legal doctrine, so some failures likely reflect missing domain knowledge rather than weakness in verification-style reasoning; a full system would need retrieval and a formal rule library to close those gaps. My choice of benchmark tasks and prompts also emphasized exact-format categorical answers, which does not measure the quality of explanations or performance on open-ended legal reasoning. And, the lack of a more 'modern' LLM (Akin to GPT-5 or DeepSeek-R1) alongside GPT-4 is due to primarily personal research funding limitations; however, this deficiency will be corrected in future testings of the LP paradigm. Finally, my experiment has certainly not yet demonstrated end-to-end formal legal verification: there is no mature proof-assistant library for law comparable to `mathlib`, and selecting an appropriate legal formalism (proof assistant, logic programming, or a restricted logic) remains a key engineering and research choice. While the LP aspirations outlined in section three are not yet tangible, there already exists a clear research path for its potential realization.

6.1 Future Directions

Several concrete research directions follow naturally from this work. The most immediate is building a domain-specific formal knowledge base (an executable library of statutes, definitions, and exceptions) for an area of law that is already highly codified and comparatively determinate, such as select portions of immigration or tax law. A focused repository of formal rules (in Lean, Prolog, or a custom legal DSL), developed in collaboration with legal experts, would enable true end-to-end evaluation: the system could retrieve authorities, translate them into formal premises, and produce verifiable proofs for questions that fall squarely within that domain.

In parallel, the prover component itself should be adapted to law. Our experiments rely on a math-oriented prover largely for availability; a dedicated legal prover LLM would likely need fine-tuning on legal reasoning patterns and formal representations of legal rules. A key obstacle is the scarcity of "gold" formal proofs in law. This suggests a hands-on strategy: generate proof-like supervision

Table 1: Performance Comparison on LegalBench Subsets

Model	Issue	Rule	Conclusion	Interpretation	Rhetorical
DeepSeek-Prover-v2	**98.6%**	**82.8%**	**97.3%**	**80.9%**	N/A
GPT-4	82.9%	59.2%	89.9%	75.2%	**79.4%**

from existing legal QA datasets by eliciting structured, element-by-element justifications (from experts or carefully constrained model pipelines) and treating those derivations as training targets for the prover and verifier loop.

Finally, the framework needs evaluation and governance that matches the domain. Testing on realistic legal problems—bar-style questions, multi-issue hypotheticals, or carefully framed advice scenarios—would reveal whether the system scales beyond single-step tasks and where it breaks (for example, whether it handles rule application well but struggles with open-textured interpretation). Alongside this, ethical and practical questions should be treated as first-class research problems: how to present proofs in lawyer-friendly form, how to keep formal rule libraries synchronized with changing law, how the system should communicate uncertainty or "not proven" outcomes, and what validation standards are required before deploying a mock LP in real-world legal environments.

7 Conclusion

I have presented a novel approach to legal AI that integrates large language models with formal proof verification, drawing inspiration from recent advances in theorem-proving LLMs. My approach aims to bridge the gap between natural language legal corpora and formal logical reasoning systems by using a two-tiered LLM framework: a general legal reasoner and a formal prover, supported by a verifier and retriever. This architecture is designed to ensure that legal conclusions generated by AI are not only correct, but accompanied by a verifiable chain of reasoning, addressing one of the most critical requirements for AI in the legal domain: trust and transparency.

Through rewriting and polishing the provided content, and integrating insights from a range of contemporary papers, I highlighted how this work builds on and differs from prior efforts. Benchmarks like LegalBench and LawBench have demonstrated both the potential and the shortcomings of current LLMs on legal tasks([Guha et al. 2023], [Fei et al. 2024]). My approach takes a further step by proposing a full embedding of an LLM within a formal proof loop.

The experimental results, albeit preliminary, give a glimpse of the potential of LP models. DeepSeek-Prover-V2 outperformed a similarly-sized black-box model on many legal reasoning tasks, especially where logical structure and multi-step deduction were key. It did so despite no adaptation to the legal domain, which bodes well for future improvements. I envision that a fully realized LP LLM could serve as a dependable assistant for lawyers, capable of checking the consistency of arguments, finding applicable rules, and even generating first drafts of arguments that a human can then review. In education, such a system could help train law students by providing automated feedback on whether their arguments logically follow from the law (acting as a sort of Socratic tutor that always checks each step).

Ultimately, I hope to narrow the long-standing divide between computational logic approaches to law and statistical NLP. By harnessing the power of both, I aim to create legal AI systems that are not only intelligent in an abstract sense, but also aligned with the rigorous reasoning standards of legal practice. This cross-pollination between CoT and non-CoT can lead to a new generation of legal AI that is both cutting-edge and practically useful. The path forward will require collaboration between computer scientists, legal scholars, and domain experts, but the reward will be systems that truly reason like a lawyer, backed by logic strong enough to withstand scrutiny and inspire professional confidence.

References

[n. d.]. My Experimental Model Source Location. https://huggingface.co/deepseek-ai/DeepSeek-Prover-V2-671B.

Kevin D. Ashley. 1988. *Modelling Legal Argument: Reasoning with Cases and Hypotheticals.* Ph. D. Dissertation. University of Massachusetts.

Odysseas Chlapanis, Dimitris Galanis, and Ion Androutsopoulos. 2024. LAR-ECHR: A New Legal Argument Reasoning Task and Dataset for Cases of the European Court of Human Rights. In *Proceedings of the Natural Legal Language Processing Workshop 2024.*

Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, and Fengzhe Zhou. 2024. LawBench: Benchmarking Legal Knowledge of Large Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing.*

Neel Guha, Julian Nyarko, Daniel E. Ho, and Christopher Ré. 2023. LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models. *arXiv preprint arXiv:2308.11462* (2023). http://arxiv.org/abs/2308.11462

Yinghao Hu, Yaoyao Yu, Leilei Gan, and Bin Wei. 2025. Evaluating Test-Time Scaling LLMs for Legal Reasoning: OpenAI o1, DeepSeek-R1, and Beyond. *arXiv preprint arXiv:2503.16040* (2025). https://arxiv.org/abs/2503.16040

Abhinav Joshi, Shounak Paul, Akshat Sharma, Pawan Goyal, Saptarshi Ghosh, and Ashutosh Modi. 2024. IL-TUR: Benchmark for Indian Legal Text Understanding and Reasoning. *arXiv preprint arXiv:2407.05399* (2024). http://arxiv.org/abs/2407.05399

Manuj Kant, Sareh Nabi, Manav Kant, Roland Scharrer, Ma Megan, and Marzieh Nabi. 2025. Towards Robust Legal Reasoning: Harnessing Logical LLMs in Law. *arXiv preview arXiv:2502.17638* (2025). http://arxiv.org/abs/2502.17638

Z.Z. Ren, Zhihong Shao, Junxiao Song, and Huajian Xin. 2025. DeepSeek-Prover-V2: Advancing Formal Mathematical Reasoning via Reinforcement Learning for Subgoal Decomposition. *arXiv preprint arXiv:2504.21801* (2025). http://arxiv.org/abs/2504.21801

M.J. Sergot, F. Sadri, R.A. Kowalski, F. Kriwaczek, P. Hammond, and H.T. Cory. 1986. The British Nationality Act as a Logic Program. *Commun. ACM* (1986).

Sumanth Varambally, Thomas Voice, Yanchao Sun, and Zhifeng Chen. 2025. Hilbert: Recursively Building Formal Proofs with Informal Reasoning. *arXiv preprint arXiv:2509.22819* (2025). http://arxiv.org/abs/2509.22819

A Appendix

Table 2: DeepSeek-Prover (DSP) vs Baseline Model on Legal Tasks (0-shot setting). Accuracy (%) of each model is reported.

Task	Samples	DSP (%)	GPT-4 (%)
hearsay	94	92.6	83.8
personal_jurisdiction	50	86.0	91.4
proa (professional responsibility)	95	93.7	99.0
consumer_contracts_qa	100	99.0	93.6
abercrombie (trademark distinctiveness)	95	37.9	85.3
corporate_lobbying	100	84.0	81.7
canada_tax_court_outcomes	100	98.0	98.9
privacy_policy_entailment	100	73.0	85.5
insurance_policy_interpretation	100	47.0	69.6
citation_prediction_classification	100	63.0	71.3
contract_qa (NDA QA)	80	98.8	96.2
contract_nli_confidentiality	82	50.0	96.3
contract_nli_explicit_id	100	26.0	82.4
contract_nli_inclusion_verbally	100	65.0	90.7
contract_nli_limited_use	100	86.0	86.6
contract_nli_no_licensing	100	50.0	92.5
contract_nli_notice_disclosure	100	74.0	97.2
contract_nli_permissible_acq	100	61.0	96.1
contract_nli_permissible_copy	87	26.4	80.4
contract_nli_permissible_dev	100	60.0	98.5
contract_nli_post_possession	100	33.0	94.6
contract_nli_return_of_info	66	59.1	95.6
contract_nli_sharing_employees	100	90.0	94.6
contract_nli_sharing_third_parties	100	71.0	93.3
contract_nli_survival	100	70.0	94.0
cuad_affiliate_license_licensee	100	92.0	90.9
cuad_affiliate_license_licensor	88	53.4	92.0
cuad_anti_assignment	100	70.0	91.4
cuad_audit_rights	100	96.0	97.9
cuad_cap_on_liability	100	71.0	95.6

Table 3: DSP Results with 3-shot prompting, compared to GPT-4. GPT-4 results are from [Guha et al. 2023] for the same tasks.

Task	Samples	DSP (3-shot)	GPT-4
hearsay	94	**97.8%**	83.8%
personal_jurisdiction	50	**99.3%**	91.4%
proa (professional responsibility)	95	93.3%	**99.0%**
consumer_contracts_qa	100	**98.3%**	93.6%
abercrombie	95	53.7%	**85.3%**
corporate_lobbying	100	**86.0%**	81.7%
canada_tax_court_outcomes	100	97.3%	**98.9%**
privacy_policy_entailment	100	82.0%	**85.5%**
insurance_policy_interpretation	100	49.3%	**69.6%**
citation_prediction_classification	100	50.3%	**71.3%**
contract_qa	80	98.8%	96.2%
contract_nli_confidentiality_of_agreement	82	75.2%	**96.3%**
contract_nli_explicit_identification	100	68.0%	**82.4%**
contract_nli_inclusion_of_verbally_conveyed_information	100	63.3%	**90.7%**
contract_nli_limited_use	100	**98.7%**	86.6%
contract_nli_no_licensing	100	82.7%	**92.5%**
contract_nli_notice_on_compelled_disclosure	100	93.3%	**97.2%**
contract_nli_permissible_acquirement_of_similar_information	100	94.0%	**96.1%**
contract_nli_permissible_copy	87	56.3%	**80.4%**
contract_nli_permissible_development_of_similar_information	100	86.7%	**98.5%**
contract_nli_permissible_post-agreement_possession	100	70.3%	**94.6%**
contract_nli_return_of_confidential_information	66	77.3%	**95.6%**
contract_nli_sharing_with_employees	100	92.0%	**94.6%**
contract_nli_sharing_with_third-parties	100	79.3%	**93.3%**
contract_nli_survival_of_obligations	100	83.3%	**94.0%**
cuad_affiliate_license-licensee	100	**95.7%**	90.9%
cuad_affiliate_license-licensor	88	75.8%	**92.0%**
cuad_anti-assignment	100	**97.7%**	91.4%
cuad_audit_rights	100	93.7%	**97.9%**
cuad_cap_on_liability	100	**98.0%**	95.6%