

NORTHWESTERN UNIVERSITY

Large Language Models as Legal Reasoners

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

by

Sergio Servantez

Technical Report Number: NU-CS-2026-19

EVANSTON, ILLINOIS

June 2026

© Copyright by Sergio Servantez, 2026

All Rights Reserved

Dissertation Committee:

Professor Kristian Hammond, Chair

Professor Sarah Lawsky

Professor Douglas Downey

Professor Larry Birnbaum

Abstract

The legal system is built upon complex bodies of written text that function as operative rules governing rights, obligations, and outcomes. Recent advances in natural language processing have created new opportunities to automate aspects of legal analysis using language models (LMs). However, these models remain fundamentally probabilistic systems optimized for statistical prediction rather than faithful rule application. As a result, they often generate fluent legal language while failing to perform the precise, multi-step reasoning required to correctly interpret and apply legal rules. This dissertation investigates how language models can be improved and rigorously evaluated as legal reasoners through the integration of symbolic representations and methods grounded in legal domain knowledge.

Focusing on a tractable subset of legal tasks termed computational legal reasoning, this work develops neurosymbolic approaches that structure legal interpretation, rule application, and evaluation. First, it introduces a system for transforming natural language contracts into machine-readable representations through the extraction of Obligation Logic Graphs (OLGs), enabling contractual obligations to be represented as structured logical relationships and translated into executable code. Second, it presents Chain of Logic, a prompting method designed to improve rule-based reasoning in language models by decomposing compositional rules into their constituent elements before recombining the results to reach a final conclusion. Third, it introduces OpenExempt, a dynamic framework and benchmark that generates natural language legal tasks and their solutions from expert-crafted encodings of statutes and case facts, enabling fine-grained diagnostic evaluation of model behavior across controlled variations in reasoning complexity and task

structure. Together, these contributions advance the study of language models as legal reasoners by introducing methods that improve interpretability, reliability, and diagnostic evaluation.

Table of Contents

Abstract	3
Table of Contents	5
List of Figures	10
List of Tables	11
1 Introduction	12
1.1 What is Legal Reasoning?	14
1.2 Why is Legal Reasoning a Challenge for Language Models?	15
1.3 Thesis Statement	17
1.4 Research Contributions	18
1.5 Scope and Responsible Use	19
1.6 Roadmap	19
1.6.1 Automating Computable Contracts through Graph-Based Extraction	19
1.6.2 Enhancing Rule-Based Reasoning on Compositional Rules	20
1.6.3 Diagnosing Legal Reasoning through Dynamic Task Generation . .	21
2 Automating Contract Logic by Extracting Obligation Logic Graphs	23

2.1	Clause to Code Translation	23
2.2	Related Work	26
2.2.1	Information Extraction	26
2.2.2	Legal NLP on Contracts	27
2.2.3	Obligation Extraction	27
2.2.4	Computable Contracts	28
2.3	Obligation Logic Graphs	29
2.3.1	OLG Entity Types	30
2.3.2	OLG Relation Types	31
2.3.3	OLG Formulas	32
2.4	The Contract-OLG Dataset	32
2.4.1	Preprocessing	33
2.4.2	Annotation Task	33
2.4.3	Dataset Statistics	35
2.4.4	Inter-Annotator Agreement	36
2.5	OLG Extraction Task	37
2.5.1	Entity Extraction	37
2.5.2	Relation Extraction	38
2.6	Code Generation	39
2.7	Experiments	41
2.7.1	Entity and Relationship Models	41
2.7.2	Evaluation	43
2.7.3	Entity Extraction Performance	43
2.7.4	Relation Extraction Performance	44
2.7.5	Contract Pre-training	45
2.8	Limitations	46

2.9	Chapter Summary	46
2.10	From Structured Extraction to Structured Reasoning	47
3	Rule-Based Reasoning with Large Language Models	49
3.1	Rule-Based Reasoning	49
3.2	Prompting Methods for Reasoning	52
3.3	Chain of Logic	53
3.4	Experiments	56
3.4.1	Tasks	56
3.4.2	Baseline Methods	57
3.4.3	Language Models	58
3.5	Results and Discussion	59
3.5.1	Diversity Jurisdiction Series	61
3.5.2	Ablation Study	63
3.6	Related Work	64
3.7	Limitations	65
3.8	Chapter Summary	65
3.9	Toward More Rigorous Evaluation of Legal Reasoning	66
4	Diagnostic Evaluation of Legal Reasoning through Dynamic Task Generation	68
4.1	Challenges in Benchmarking Legal Reasoning	68
4.2	Related Work	71
4.2.1	Legal Reasoning Benchmarks	71
4.2.2	Computable Statutory Reasoning	72
4.3	The OpenExempt Framework	73
4.3.1	Asset Exemption in Bankruptcy	74

4.3.2	Structured Legal Knowledge	74
4.3.3	Dynamic Task Generation	77
4.3.4	Computing Gold Solutions	78
4.4	The OpenExempt Benchmark	80
4.4.1	Tasks	80
4.4.2	Benchmark Suites	83
4.5	Results and Discussion	86
4.5.1	Experimental Setup	86
4.5.2	Models	87
4.5.3	Evaluation Protocol	88
4.5.4	Competency Evaluation	89
4.5.5	Diagnostic Evaluation	90
4.6	Limitations	94
4.7	Chapter Summary	94
5	A Unified Perspective on Language Models as Legal Reasoners	96
5.1	The Unified Perspective	96
5.2	Core Insights	97
5.3	Future Directions	98
5.4	Concluding Remarks	100
	References	101
A	Chain of Logic Appendix	113
A.1	LegalBench Task Examples	113
A.2	Model Performance by Rule	117

B	OpenExempt Appendix	119
B.1	Modular Task Prompts	119
B.2	Response Format Compliance	120
B.3	Response Validation	120
B.4	Configuration Parameters	122
B.5	Benchmark Suite Composition	123
B.6	Diagnostic Suite Results	124
B.7	Task Prompt Examples	134
B.8	Statute Source by Jurisdiction	137

List of Figures

2.1	Example of OLG Annotation Task.	24
2.2	OLG Schema and Instance.	25
2.3	Screenshot of OLG Annotation Interface.	34
2.4	Entity Type Distribution Across Contract-OLG Dataset.	36
2.5	Relation Type Distribution Across Contract-OLG Dataset.	37
2.6	Code Generation Example Mapping Extracted OLG to Accord DSL.	39
3.1	Example of Compositional Rule.	50
3.2	One-Shot Examples of Prompting Methods on Personal Jurisdiction Task.	53
3.3	Comparison of Model Reasoning on Personal Jurisdiction Task.	55
3.4	Accuracy (%) Across All 6 Diversity Jurisdiction Tasks Using GPT-3.5.	61
3.5	GPT-4 Output Demonstrating Common Errors in One-Shot Methods.	62
4.1	Asset Exemption Overview.	69
4.2	OpenExempt Framework Architecture.	73
4.3	Benchmark Suite Distribution.	80
4.4	OpenExempt Task Pipeline.	81
4.5	Example of OpenExempt Task Prompt.	83
4.6	Impact of Distractor, Sycophancy, and Obfuscation Perturbations.	89
4.7	Model Performance ($F1$) on Temporal Reasoning and Asset Scaling Suites.	90
4.8	Obfuscation Robustness on Optimal Exemption Task.	91
4.9	Reasoning Decomposition on Optimal Exemption Task.	93
B.1	Example of Allowable Exemption Task Prompt and Solution.	134
B.2	Example of Exemption Classification Task Prompt and Solution.	135
B.3	Example of Optimal Exemption Task Prompt and Solution.	137

List of Tables

2.1	Contract-OLG Dataset Statistics.	35
2.2	Deontic Modality Distribution for Contract-OLG Dataset.	36
2.3	Precision, Recall and F1 for Entity Extraction Task on Contract-OLG. . . .	44
2.4	Precision, Recall and F1 for Relation Extraction Task on Contract-OLG. . .	46
3.1	Aggregated Performance (Accuracy) Across LegalBench Reasoning Tasks.	60
3.2	Ablation Study of Performance Loss from Removing Reasoning Steps. . .	63
4.1	OpenExempt Constraints and Dependencies.	76
4.2	Model Performance (Sample-Based $F1$) Across Competency Suites.	87
A.1	GPT-3.5 Performance (Accuracy) by LegalBench Rule.	117
A.2	GPT-4 Performance (Accuracy) by LegalBench Rule.	117
A.3	Llama-2-70b Performance (Accuracy) by LegalBench Rule.	118
A.4	MistralOrca-7b Performance (Accuracy) by LegalBench Rule.	118
B.1	Frequency and Percentage of Model Responses with Malformed JSON. . .	120
B.2	OpenExempt Configuration Parameters.	122
B.3	Summary of Configuration Settings Across Evaluation Suites.	123
B.4	Model Performance ($F1$) on Temporal Reasoning Suite.	124
B.5	Efficient Model Performance ($F1$) on Reasoning Decomposition Suite. . .	125
B.6	Large Model Performance ($F1$) on Reasoning Decomposition Suite. . . .	126
B.7	Reasoning Model Performance ($F1$) on Reasoning Decomposition Suite. .	127
B.8	Efficient Model Performance ($F1$) on Robustness Suites.	128
B.9	Large Model Performance ($F1$) on Robustness Suites.	129
B.10	Reasoning Model Performance ($F1$) on Robustness Suites.	130
B.11	Efficient Model Performance ($F1$) on Asset Scaling Suite.	131
B.12	Large Model Performance ($F1$) on Asset Scaling Suite.	132
B.13	Reasoning Model Performance ($F1$) on Asset Scaling Suite.	133
B.14	Sources of Statutes by Jurisdiction.	137

Chapter 1

Introduction

The legal profession is inherently document-centric. From statutes and judicial opinions to contracts and regulations, the legal system continuously produces vast volumes of unstructured text. Yet, unlike general prose, legal text is operative; it functions as a rigorous set of instructions that dictate rights, obligations, and outcomes. Because of this complexity, society relies on human experts—lawyers and judges—to navigate the intricate, technical language of the law. This heavy reliance on written text positions the legal field as a prime candidate for transformation by emerging technologies in natural language processing (NLP).

The recent revolution in NLP, driven by deep neural networks, presents a path toward automating the understanding and generation of legal text at scale. However, the integration of these models into legal practice faces a fundamental friction. Modern language models (LMs) are probabilistic engines that excel at pattern matching and statistical prediction. Law, conversely, is a prescriptive domain that requires strict adherence to rules where a single word (e.g., shall vs. may) can invert an outcome.

While deep learning has achieved remarkable success in general domains, it faces unique hurdles in law. First, the data scarcity problem is acute: while raw legal text is

abundant, the expert demonstrations and feedback signals required for effective post-training, particularly instruction tuning and reinforcement learning, are rare and expensive to create as it requires highly specialized labor. Second, even frontier language models struggle with the reasoning gap: the distance between generating fluent, legal-sounding text and performing the precise, multi-step logic required to resolve a legal issue correctly. Recent work has shown these models can struggle with basic legal tasks [Blair-Stanek, Holzenberger, and Durme 2023], and frequently generate hallucinations in a legal setting [Dahl et al. 2024; Magesh et al. 2024].

This dissertation confronts these challenges. Positioned at the intersection of computational law (historically dominated by symbolic logic) and modern NLP (driven by transformer-based language models), it introduces neurosymbolic methods to bridge the gap between probabilistic models and rule-bound legal reasoning by drawing inspiration from legal domain knowledge and conventions. Valid legal judgment requires more than predicting an outcome; it demands the faithful application of authoritative rules that dictate rights, obligations, and consequences, often across extended reasoning paths. Therefore, a robust framework for computational legal reasoning must (i) represent legal meaning in a way that exposes conditions and dependencies, (ii) apply rules through a procedure that makes intermediate judgments and resulting consequences explicit, and (iii) be evaluated with methods capable of isolating where failures occur and why. These requirements form the backbone of this dissertation, which advances extractive methods for legal interpretation, generative approaches to structured rule application, and robust evaluation methodologies that diagnose model behavior under controlled variation.

1.1 What is Legal Reasoning?

Reasoning is the process of drawing conclusions based on established facts and premises [Lohman and Lakin 2011]. When applied to the legal domain, however, this definition becomes the subject of debate. Despite a century of discussion, there remains no scholarly consensus on what precisely constitutes legal reasoning, although legal scholars often converge on two cognitive processes: deductive (rule-based) reasoning, which applies legal rules to facts, and analogical (case-based) reasoning, which resolves new disputes by comparing them to patterns in prior cases [Ellsworth 2005].

Yet if we examine the mechanics of these frameworks, we find that legal reasoning is not monolithic, but instead decomposes into a diverse set of cognitive tasks. Rule-based reasoning is often framed through a legal syllogism, dictating that consequences follow if conditions are met. In practice, however, determining whether a condition is satisfied or understanding the resulting consequence can itself require many other forms of reasoning. For example, a single rule application may require temporal reasoning about sequences of events, compositional reasoning to parse the boolean structure of a complex statute, and mathematical reasoning to quantify damages or eligibility thresholds. Because the law touches nearly every facet of human activity, the reasoning required to practice it must be equally multifaceted. Therefore, in this dissertation we adopt a functional definition of legal reasoning as encompassing the spectrum of inferential processes a lawyer must perform to effectively discharge their duties.

Because legal reasoning is so expansive, any empirical study must define a tractable and principled scope. This dissertation focuses on a distinct subset that is both foundational to legal practice and amenable to rigorous evaluation: **computational legal reason-**

ing. We define this as the class of legal tasks for which there exists a definitive solution with a high degree of confidence, given (i) a specified set of facts and (ii) a governing legal authority whose application can be verified. This focus creates a setting in which success and failure can be measured precisely, task difficulty can be carefully controlled, and error signals can be meaningfully interpreted; a necessary setting for making credible claims about language models as legal reasoners.

Accordingly, we set aside forms of legal reasoning that resist objective evaluation, including tasks that depend primarily on normative judgment, open-textured standards (e.g., determining what is “reasonable”), or persuasive argumentation. When the right answer is subjective, evaluation must rely on proxy methods, such as LLM-as-a-judge. However, recent work in the legal domain has shown these proxy methods exhibit only moderate correlation (Spearman ≈ 0.75) with expert evaluation even under careful alignment [Shi et al. 2026]. This scoring variance obscures the signal needed for fine-grained assessments of legal reasoning. Therefore, our exclusion of these tasks is methodological rather than normative; it is not a claim about what is most important in law, but rather a reflection of the difficulty in scoring such forms objectively without collapsing evaluation into preferences about style and ideology.

1.2 Why is Legal Reasoning a Challenge for Language Models?

Even within the tractable scope of computational legal reasoning, legal tasks remain difficult for language models for several reasons. First, legal text is not merely descriptive, it is operative. Statutes, contracts, and regulations function as instruments that create rights and obligations, and correctness often depends on nuanced semantics. Legal au-

thorities encode conditions, exceptions, cross-references, and definitions whose meaning depends on context, and whose consequences often require the application of complex rules. Yet modern language models are optimized to generate text that is statistically probable given their training distribution, not text constrained by an authoritative rule system [Grimmelmann et al. 2025]. This mismatch produces fluent and persuasive outputs that often fail to reflect the disciplined process of legal reasoning [Grimmelmann et al. 2025].

These weaknesses become especially clear in deductive reasoning (often framed as legal syllogism). In this setting, rule-based reasoning requires explicitly connecting governing rules to specific case facts while maintaining a coherent logical state across multiple steps. Language models struggle to construct and sustain such reasoning paths [K. Zhang et al. 2025]. They frequently misinterpret legal nuances, conflate similar rule elements, and fail to prioritize relevant facts over distracting information [Chlapanis et al. 2025; Hu et al. 2025]. As a result, models often arrive at incorrect conclusions from valid premises or, conversely, predict the correct final answer based on false, incomplete, or overly generic rationales [V. Mishra et al. 2025].

Beyond logical brittleness, models exhibit a lack of robustness to extraneous information, making them unstable for high-stakes legal applications [Grimmelmann et al. 2025]. Their outputs are highly sensitive to irrelevant variables, such as minor changes in prompt phrasing, positional placement of information, or injected adversarial noise [Hu et al. 2025]. This fragility extends to their factual reliability. Language models are prone to hallucinating non-existent case details such as parties to a lawsuit [V. Mishra et al. 2025], or ignoring explicit constraints like jurisdictional scope [Trozze et al. 2024]. They also suffer from temporal bias, struggling to parse the archaic terminology and var-

ied citation structures of historical legal texts [L. Zhang, Savelka, et al. 2025]. Because these systems operate as opaque black boxes, such unpredictable and sometimes fabricated errors undermine the reliability required in legal settings.

Finally, addressing these shortcomings is complicated by the inherent difficulty of evaluating legal reasoning. Standard text-generation metrics, such as accuracy and ROUGE [C.-Y. Lin 2004], are insufficient because they cannot assess legal validity or faithful rule application. Meanwhile, existing benchmarks often rely on multiple-choice formats that fail to capture the complexity of open-ended legal analysis [Chlapanis et al. 2025]. These approaches are also vulnerable to data contamination, which can inflate performance by rewarding memorization rather than reasoning [Shojaee et al. 2025]. At the same time, open-ended evaluation is difficult to scale because it requires expensive expert scoring [Guha, Nyarko, Ho, and Ré 2025]. Together, these measurement limits make it difficult to determine whether models genuinely understand and apply legal rules or merely produce plausible legal language.

1.3 Thesis Statement

This dissertation argues that legal reasoning with language models is advanced by integrating symbolic methods that mimic the structured problem-solving strategies of legal professionals. By translating the cognitive frameworks lawyers use to interpret contracts, reason about rules, and evaluate statutory compliance into symbolic representations, we demonstrate that grounding neural methods in legal domain knowledge significantly improves the reliability, interpretability, and diagnostic evaluation of legal reasoning.

1.4 Research Contributions

This dissertation presents new methods for legal reasoning with language models and introduces a robust framework for evaluating and understanding their capabilities as legal reasoners. Collectively, this work advances legal reasoning with language models through contributions in extraction, generation, and evaluation. The research contributions of this dissertation are organized as follows:

- Chapter 2 presents the first system for converting natural language contracts into a code representation through the automatic extraction of obligation logic graphs (OLGs) and subsequent conversion into the Accord domain-specific language (DSL). We introduce OLG extraction as a new joint entity and relation extraction task, and present the Contract-OLG dataset, which consists of 1,876 contract provisions, 18,597 entities (7 unique categories), and 18,170 relations (24 unique categories).
- Chapter 3 introduces Chain of Logic, a prompting method that guides language models in reasoning about compositional rules by decomposing rule elements into a sequence of logical statements and then recomposing intermediate conclusions to resolve the overall task. We evaluate several common in-context learning approaches across 8 tasks and 3 distinct rules from LegalBench. Our experiments show that Chain of Logic consistently outperforms alternative prompting methods across these tasks and rules.
- Chapter 4 introduces OpenExempt, a dynamic framework and diagnostic benchmark for fine-grained evaluation of legal reasoning in language models. OpenExempt constructs natural language tasks and their solutions on demand from expert-crafted encodings of statutes and case facts, enabling controlled variation in task

scope and complexity without requiring direct annotation of solutions.

1.5 Scope and Responsible Use

This dissertation is not an argument that language models should practice law, replace legal professionals, or be relied upon as autonomous decision makers in high-stakes settings. On the contrary, this work underscores the profound challenges of modeling legal reasoning tasks and introduces methods for making that behavior more dependable and measurable. The deployment of generative models in real-world legal environments introduces significant ethical risks, including hallucination, opaque error modes, and the potential for over-reliance by non-experts. To mitigate these risks, this dissertation advances new representations, procedures, and evaluation methods that support transparency, interpretability, and diagnostic insight. These properties must be considered prerequisites for responsible deployment in any setting where machine outputs impact legal rights, obligations and outcomes.

1.6 Roadmap

The remainder of this introduction provides a roadmap of the chapters that follow, summarizing the central problem, method, and contribution of each.

1.6.1 Automating Computable Contracts through Graph-Based Extraction

Computable contracts are agreements represented in a machine-readable form, typically code, that can be automatically interpreted and executed. Representing contracts as code has the potential to significantly reduce the time and cost of contract management by

enabling automated verification and execution of contractual obligations. In response, several domain-specific languages have emerged for expressing contractual logic. However, adoption of computable contracts has remained limited, largely because converting existing natural language contracts into code remains costly and labor-intensive.

Chapter 2 addresses this challenge by introducing the first system for automatically converting natural language contracts into code through the extraction of structured representations of contractual obligations. Our approach extracts key entities, relationships, and formulas into a graph-based representation called the Obligation Logic Graph (OLG), which captures the semantic structure of contractual obligations and their dependencies. These graphs can then be mapped to executable representations in downstream systems. To support this task, we introduce OLG extraction as a new joint entity and relation prediction task for legal contracts and present the Contract-OLG dataset, consisting of 1,876 contract provisions, 18,597 entities, and 18,170 relationships. We evaluate several state-of-the-art Transformer and graph-based models on this task and identify substantial performance gaps between human experts and current models, particularly for relation extraction.

1.6.2 Enhancing Rule-Based Reasoning on Compositional Rules

Rule-based reasoning is a fundamental type of legal reasoning in which conclusions are reached by applying a rule to a set of facts. Many legal tasks require reasoning over rules drawn from statutes, regulations, judicial decisions, and contracts. These rules are often compositional, with antecedents that consist of multiple rule elements connected through a logical expression. Reasoning over such rules is challenging because it requires executing multiple reasoning steps while preserving the logical relationships between elements.

Chapter 3 examines whether causal language models can perform this type of rule-based reasoning reliably. To address this challenge, we introduce Chain of Logic, a prompting method that elicits rule-based reasoning through decomposition and recombination. The method first evaluates individual rule elements as separate threads of reasoning and then recombines those intermediate conclusions to resolve the overall logical expression. Inspired by the IRAC (Issue, Rule, Application, Conclusion) framework used in legal analysis, Chain of Logic provides a structured way to reason over compositional rules. We evaluate the method across eight rule-based reasoning tasks derived from three compositional rules in LegalBench and show that it consistently outperforms alternative prompting methods, including Chain-of-Thought and Self-Ask, across both open-source and commercial language models.

1.6.3 Diagnosing Legal Reasoning through Dynamic Task Generation

Evaluating the reasoning capabilities of language models has become increasingly challenging as models improve on many complex tasks. Traditional benchmarks typically consist of static collections of question-answer pairs, which provide only a limited view of model performance. A failure on a complex problem reveals little about why a model failed or which reasoning steps caused the error. These limitations are particularly pronounced in the legal domain, where reasoning often requires applying multiple interdependent rules and carefully tracking how facts satisfy specific legal conditions. Understanding model capabilities therefore requires evaluation methods that can isolate distinct reasoning challenges and control task difficulty.

Chapter 4 introduces OpenExempt, a framework and benchmark designed to support diagnostic evaluation of legal reasoning in language models. OpenExempt constructs natural language legal tasks and their solutions on demand from expert-crafted encodings

of statutes and case facts. By combining statutory text with structured representations of legal logic, the framework enables the controlled generation of tasks that vary in scope and complexity without requiring direct annotation of solutions. Using this system, we construct the OpenExempt benchmark, consisting of nearly 10,000 samples, organized to evaluate model performance across a series of diagnostic and competency suites that probe different aspects of legal reasoning. Experiments across a diverse set of language models reveal sharp performance cliffs that emerge only under longer reasoning paths and in the presence of obfuscating statements.

Chapter 5 concludes the dissertation by synthesizing the findings of the preceding chapters, drawing final conclusions about language models as legal reasoners, and outlining directions for future research in computational legal reasoning.

Chapter 2

Automating Contract Logic by Extracting Obligation Logic Graphs

2.1 Clause to Code Translation

The nascent field of computable contracts focuses on representing contracts in symbolic form, typically code, which can be easily understood by machines. Enabling machines to understand legal contracts unlocks tremendous value, including automating execution, validation and scenario planning. The promise of computable contracts has led to the emergence of domain-specific languages (DSL) - programming languages specifically designed to represent legal documents or statutes. Yet despite the existence of several of these DSLs [[Governatori and Wong 2023](#); [Merigoux et al. 2021](#); [Roche et al. 2021](#)], few contracts or other legal documents have been converted into these code representations due to the high cost of conversion. The conversion process can be described as pair programming between a lawyer and a developer. The lawyer understands the contract, the developer understands the code and they must communicate to ensure the code is a faithful representation of the contract. Since this process requires a high degree of technical and legal skills, and a significant time commitment from both, it does not scale beyond commonly used templates to the number of agreements created and revised every day.

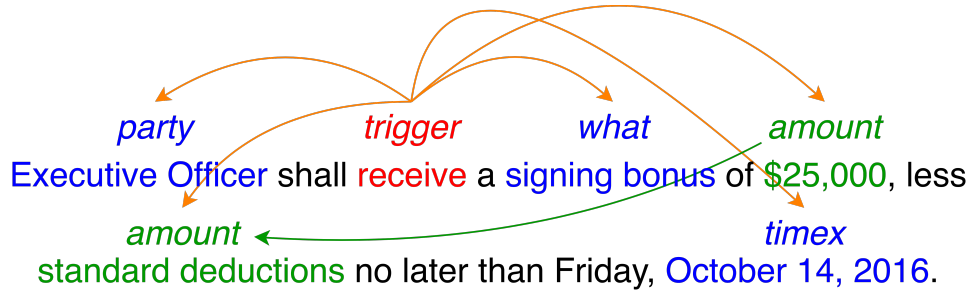


Figure 2.1: A simple obligation from our dataset with annotations for the OLG extraction task. Obligation trigger is red, obligation attributes are blue, and formula entities are green. Edges represent relations (orange edges are obligation argument roles and the green edge is a subtraction relation).

Our work is motivated by the need to simplify and work towards automating this conversion process to assist legal professionals in moving towards computational contracts. We choose to model this problem as an information extraction task in which we extract the key entities, relationships, and formulas needed to create a computational representation of the business logic in a clause, centered around obligations in the contract. We call this graph representation the Obligation Logic Graph (OLG), and also demonstrate how the representation can be easily converted to a popular computational contract language (Accord) [Roche et al. 2021]. While recent advances in natural language processing (NLP) [Chen et al. 2021] have enabled the direct conversion between language and code, we believe this intermediate representation shown in Figure 2.2 is easier for legal professionals to understand and work with. Additionally, we believe enabling a deeper semantic representation of natural language contracts will benefit both lawyers and non-legal professionals in reading, understanding, and authoring agreements.

Our research also explores how extracting these computational representations can be automated using state-of-the-art NLP methods. However, a major obstacle prior to modeling, and to legal NLP tasks in general, is the scarcity of richly annotated datasets

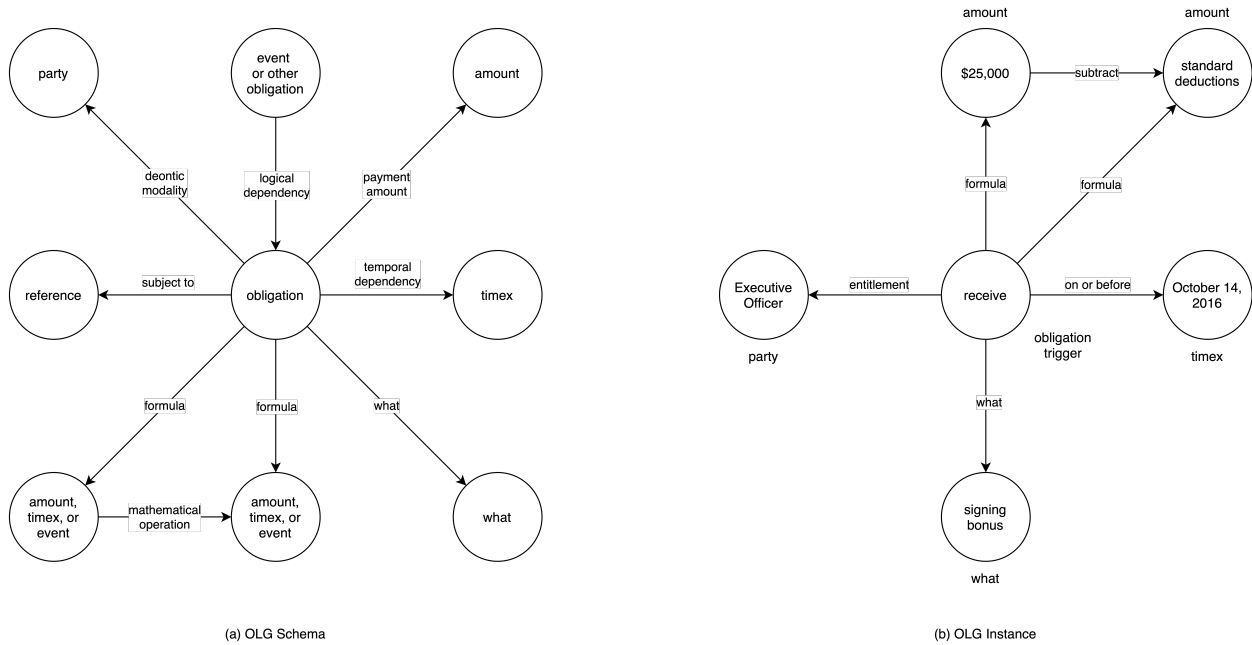


Figure 2.2: (a) OLG schema showing entity and relation types (b) OLG instance for obligation shown in Figure 2.1.

and the high cost of creating them. Unlike typical NLP datasets that can be annotated through crowdsourcing marketplaces like Mechanical Turk, annotating legal datasets requires legal expertise which narrowly restricts the pool of suitable annotators. The type of richly annotated contract data needed to perform the modeling task described above simply does not exist. And so taking on this task also required us to take on creating the necessary data, including defining a meaningful representation of entities and relations to capture the intent within the clauses. Capturing such rich annotations quickly and intuitively also required us to design a custom interface and assemble a group of legal professionals to create this dataset.

Our main research contributions are:

- We present the first system, to our knowledge, for converting natural language contracts to a code representation through the automatic extraction of obligation logic

graphs and conversion to the Accord language.

- We introduce OLG extraction as a new joint entity and relation extraction task, and introduce the Contract-OLG dataset, which consists of 1,876 contract provisions, 18,597 entities (7 unique categories) and 18,170 relations (24 unique categories). Our intent is to release this dataset to motivate future work into extracting rich semantic and computational representations from contract text. We also perform detailed experiments to understand the capabilities of state-of-the-art NLP algorithms at predicting obligation logic graphs.

2.2 Related Work

2.2.1 Information Extraction

Recent work in information extraction has used various deep learning approaches to jointly model the four tasks of entity, relation, event and argument extraction [Y. Lin et al. 2020; Van Nguyen et al. 2021; Wadden et al. 2019]. While each of these four tasks are required for OLG extraction, our task is not centered on events but rather obligations. While we still detect event triggers, obligation triggers are far more prevalent in the Contract-OLG dataset (see Figure 2.4). In addition, we also extract other common occurrences in contracts, including dependencies (both logical and temporal) and mathematical formulas (e.g. base salary minus deductions). Similar to Jie [Jie et al. 2022], we approach formula detection as a relation extraction task where mathematical operations link variables to construct a tree representation of the formula. Unlike Jie [Jie et al. 2022], our task seeks only to extract the formula representation, not solve the expression, since contract language commonly contains variables that change based on context (e.g. standard deductions). Existing work has approached condition detection as a sentence splitting task

where conditional sentences are split into antecedent and consequent using both rule-based [Wenzina and Kaiser 2013] and deep learning methods [Vo et al. 2021]. Similar to formula detection, we approach condition detection as a relation extraction task allowing us to combine simple conditions using *and/or* relations to express more complex logical dependencies.

2.2.2 Legal NLP on Contracts

Existing work has explored a broad range of legal NLP tasks on contract language, including clause classification [Hendrycks et al. 2021], contract summarization [Manor and J. J. Li 2019], and clause recommendation [Aggarwal et al. 2021]. More related to our work, contract element extraction methods have explored identifying and extracting key contract elements, like clause headings and legislation references [Chalkidis, Androutsopoulos, and Michos 2017; Z. Wang et al. 2021]. The scarcity of expertly annotated datasets has been a limiting factor for legal NLP tasks, especially in the contract domain. Tuggener [Tuggener et al. 2020] and Hendrycks [Hendrycks et al. 2021] have recently released large contract datasets to the public, but both are labeled at the clause or provision level. Our work builds on the provisions in Tuggener [Tuggener et al. 2020] to capture fine-grained entity and relation annotations.

2.2.3 Obligation Extraction

Prior work in obligation extraction has primarily focused on classification tasks at the sentential level [Chalkidis, Androutsopoulos, and Michos 2018; Glaser et al. 2018], while our approach extracts a graph representation of the obligation. Similar to an Event Logic Graph [Ding et al. 2019], our task constructs a graph of obligation nodes and conditional relations. We also go beyond the event logic graph architecture by capturing a rich set

of obligation arguments along with their respective relations. There have been several works that explore components of the OLG, but our work brings these efforts together and even adds new components to present a unified graph which captures much of the semantic meaning of an obligation. Sancheti [Sancheti et al. 2022] presented a new taxonomy for deontic types and, similar to our work, introduced a dataset for party-specific deontic modality detection with trigger span detection. Mathur [Mathur et al. 2022] presented a parser for extracting temporal dependency graphs from contracts at the document level. Funaki [Funaki et al. 2020] introduced a contract corpus annotated in XML format to identify obligations, parties, conditions and exceptions. However, their XML representation captures the entire span of an obligation or condition without capturing the many obligation arguments and relations we identify in the OLG, including obligation triggers and temporal dependencies. Their dataset is also not publicly available and is more than an order of magnitude smaller than our dataset. Additionally, we are the first, to our knowledge, to explore complex condition extraction and formula extraction as a subtask of obligation extraction.

2.2.4 Computable Contracts

Several DSLs for legal documents and statutes have emerged to address the need for computable contracts. A general survey [Christopher D Clack 2021] on computational law and contracts provides pointers to several systems as well as background on the field. Legal-RuleML [Palmirani et al. 2011] is an XML formatted representation for legal knowledge, including rules and norms, that has been used to automate compliance checking. Catala [Merigoux et al. 2021] is a DSL specifically designed to translate statutory laws into an executable representation. Stipula [Crafa et al. 2022] and L4 [Governatori and Wong 2023] are domain specific languages tailored for agreements. Stipula provides a formal specifi-

cation for logic, obligations, events, entitlements and prohibitions, and also provides an example of a rental agreement converted to this specification. L4 provides similar functionality and specializes in being more human interpretable. Similarly, the Accord Project [Roche et al. 2021] has designed their own DSL for representing computable contracts, automating contract logic and generating templated contract text (see Section 2.6 for a detailed description of the Accord technology stack). Our analysis of these languages found that they ranged from initial specification to well documented language guides. Though beyond a small number of examples, it was difficult to find publicly available codified clauses or contracts in large quantities. We chose to convert our OLG representation to the Accord language since it supported many of the data structures needed for our work.

2.3 Obligation Logic Graphs

The OLG consists of a rich set of entities (nodes) and relations (edges) to capture the semantic meaning and logic in contract text and represent it computationally. In particular, we designed it to capture the business logic needed for payment clauses, though the schema could be easily extended to capture other clause types. For example, we could easily adapt the schema to represent governing law clauses by adding an entity and relation for jurisdiction. Since the OLG must be captured through human annotations, we must balance between the schema being too simple or too complex - too simple and we do not capture enough of the obligation meaning and computational semantics, too complex and we make the annotation task onerous and time consuming. Specifically, the schema attends to the who (parties), what (subject) and when (temporal dependencies), as well as logical dependencies, formulas, and deontic modalities. Below, we enumerate these graph components and discuss their importance to the OLG. Figure 2.2 illustrates the OLG schema.

2.3.1 OLG Entity Types

Each entity is expressed as a node type and a token span of the contract text. There are 7 total entity types in the Contract-OLG dataset.

- **Obligation and Event Triggers:** Our approach centers around obligations, but we also capture events since an obligation can be conditioned on an event. We define an obligation trigger as the word span that identifies a commitment or duty. Similarly, an event trigger is a word span that identifies an occurrence, typically an action taken by a party. *Entity types: obligation trigger, event trigger.*
- **Party:** A party is a participant in the contract, typically an individual or a business. *Entity type: party.*
- **Timex:** A timex is a temporal expression, typically a specified date or a relational expression (e.g. 5 days before). *Entity type: timex.*
- **What:** The *what* entities represent the subject of an obligation. In Figure 2.1, “signing bonus” is a *what* entity since it represents what the party is receiving. *Entity type: what.*
- **Reference:** Often an obligation inherits part of its meaning by referencing an external document or a clause from the same contract (e.g. subject to Section 5.2). *Entity type: reference.*
- **Amount:** An amount is a numerical quantity, typically related to a payment amount. *Entity type: amount.*

2.3.2 OLG Relation Types

Each relation is expressed as a relation type and a directed edge between two entities. There are a total of 23 relation types in the Contract-OLG dataset.

- **Deontic Modality:** Deontic modals express the type of commitment or privilege being assigned to a party (entitlement, prohibition). These relations link obligation triggers to party entities. By representing modal expressions as a relation, we are able to express party-specific deontic modalities. *Relation types: entitlement, obligation, prohibition, permission.*
- **Logical Dependencies:** Obligations are not only interconnected, but also can be conditional on other obligations and events. We express these *if-then* relationships through logical dependency relations where the antecedent (event or obligation) triggers a consequent (obligation) based on a condition that is expressed as a relation from the former to the latter. Since obligations can be conditioned on events, we also capture event triggers in the OLG. *Relation types: if true, if false, if late.*
- **Subject To:** The *subject to* relation links obligation triggers to reference entities (e.g. subject to Section 5.2). *Relation type: subject to.*
- **Temporal:** These relations express temporal relationships from obligation and event triggers to timex entities and temporal formulas. *Relation types: before, after, on, during, reoccurring.*
- **Amount:** The amount relation links obligation triggers to amount entities and amount formulas. *Relation type: amount.*

- **What:** The *what* relation links obligation triggers to *what* entities. *Relation type: what.*
- **Formula:** For each operand in an equation, there is a formula relation linking the trigger to the operand entity. *Relation type: formula.*
- **Comparators (not used):** For completeness, we also intended to use comparison operators such as less than or equal to. However after the annotation process was complete, we found only 7 samples with these comparators and thus decided to drop these examples. *Relation types: <, <=, >, >=, ==, !=.*

2.3.3 OLG Formulas

We treat equations made of logical dependencies or mathematical operations as equation trees in order to maintain order of operations. We use equation trees for the following relation types:

- **Operation:** The operation relations express mathematical operations between entities. *Relation types: addition, subtraction, multiplication, division, maximum, minimum.*
- **And/Or:** These relations link logical dependencies to create complex conditions in antecedents. *Relation types: and, or.*

2.4 The Contract-OLG Dataset

In order to support experiments for the OLG extraction task, we construct the Contract-OLG dataset. We limit our dataset to payment-related provisions to scope the task.

2.4.1 Preprocessing

We started with provisions from LEDGAR [Tuggener et al. 2020] - a multi-label corpus of contract provisions that is available to the public. From these labeled provisions, we first identified classes with a high concentration of provisions containing payment and formula related keywords (e.g. pay, deduct). The corpus was then filtered to include only provisions that contained at least one of these classes. Note that all provisions from a class, not just those containing a keyword, were included in the filtered dataset to increase language variety. To make the annotation task more manageable, we also excluded all provisions with a length greater than 750 characters. To further increase language variety, we also removed near duplicate provisions using locality-sensitive hashing (Jaccard index > 0.7). The deduplicated dataset was then split into batches ($n = 100$) with each batch being assigned to an annotator.

2.4.2 Annotation Task

The annotators were tasked to capture all information needed to construct the OLG, and as shown in Section 2.3 the OLG schema is highly granular. Even at the provision level, a single example can contain multiple obligations and conditions. Capturing such a detailed representation of a contract provision requires a complex annotation task where annotators must identify a broad range of entities and relations for every example. To make this task more manageable, we designed an annotation interface shown in Figure 2.3 with the following goals: 1) reduce time to complete task by allowing annotators to quickly select OLG components; 2) reduce task complexity by inferring information from existing annotations; and 3) enable annotators to use the interface with minimal training. Using the interface, an annotator first identifies an obligation or event by highlighting the span of the trigger. This selection populates the interface with a list of possible obligation

The screenshot displays the OLG annotation interface. At the top, there are buttons for 'Run Model', 'Generate Code', 'Export to JSON', 'New Clause', and 'Choose File' (with 'data_json.json' selected). A navigation bar on the right contains tabs for 'Obligations / Prohibitions' (active), 'Events', 'Conditions', and 'Formulas'.

Clause Text: The main text area shows a clause: "Executive Officer * shall * receive * a * signing bonus * of * \$25,000 * , less * standard deductions * no later than Friday, * October 14, 2016 * .". The words "receive" and "signing bonus" are highlighted with star icons.

Properties: This section contains three questions:

- "What type of obligation, entitlement, or prohibition (i.e. pay, sign, review) ?" with a dropdown menu set to "pay".
- "Is it mandatory or optional or prohibited?" with a dropdown menu set to "shall".
- "Is this a bulk constraint - meaning it applies to all obligations of the same type?" with a dropdown menu set to "No".

 At the bottom, there is a prompt to "Highlight the word best represents the obligation, entitlement, or prohibition you are trying to represent?" with a star icon and the word "receive" selected.

Attributes: This section contains several fields:

- "Who:" with a dropdown menu set to "Executive Officer" and a button labeled "Entitled".
- "When:" with a dropdown menu set to "Before" and a date picker set to "October 14, 2016".
- "Subject To:" with a plus icon button.
- "Payment_Method:" with a plus icon button.
- "Payment_Amount:" with a dropdown menu set to "Formula: 0" and a plus icon button.
- "What:" with a dropdown menu set to "signing bonus" and a plus icon button.

Sidebar: The sidebar on the right shows a list of annotations:

- "Obligation / Prohibition" with a sub-item "Obligation 0: receive" and a dropdown menu labeled "Attributes".
- "Event" with a plus icon button.
- "Conditions" with a plus icon button.
- "Formulas" with a sub-item "Formula 0" and a dropdown menu labeled "Attributes".

Figure 2.3: Screenshot of our OLG annotation interface populated with annotations for the obligation shown in Figure 2.1.

arguments which can be quickly filled by the annotator based on which arguments exist in the provision. This approach was motivated by work in event argument mining [S. Li et al. 2021] and allows us to naturally pair every argument to its corresponding obligation and thereby infer all obligation argument relations. Inspired by Blockly and Scratch [Resnick et al. 2009], the interface also allows annotators to construct equation trees and conditions by dragging and dropping interlocking blocks of entities and formulas to create new relations and equations.

Since the annotation task requires the interpretation of complex contract language, we hired lawyers and law school students from Upwork to be annotators. Candidates were initially given an annotation guideline describing the task in detail, and were later given a demo of the annotation interface. Candidates were then tested by asking each to annotate the same 10 contract provisions using the interface. From the 6 candidates that took the test, we selected the top 4 who demonstrated the strongest understanding of the task. These annotators were then assigned batches of 100 provisions to complete at a time. Annotators were also given the option of skipping a provision if they felt like the schema

Table 2.1: Contract-OLG dataset statistics

Dataset	Provisions	Entities	Relations	Triggers	Arguments	Roles	Formula/ Logic
Train	1660	16231	15934	4178	11106	12683	3251
Dev	108	1122	1068	300	772	851	217
Test	108	1244	1168	324	852	979	189

would not capture its meaning or it contained no obligations. Of the data assigned, only 12 percent of provisions were skipped, providing evidence that the OLG schema could capture most of the provisions.

2.4.3 Dataset Statistics

There are 1,876 total provisions (see Table 2.1) with a mean of 1.49 (standard deviation $\sigma = 0.68$) sentences per provision and 366.18 ($\sigma = 168.21$) characters per provision. We collect 18,597 total span annotations with a mean of 9.9 ($\sigma = 4.78$) spans per provision, and 18,170 total relations with a mean of 9.69 ($\sigma = 5.59$) relations per provision. The dataset includes 30 total label categories, including 7 entity categories and 23 relation categories. Note that some related categories have been merged in Figure 2.5 to better visualize the distribution. For a complete list of relation categories, see Section 2.3. We chose to not have provisions annotated by more than one annotator, and instead focused on language variety by maximizing the number of annotated provisions. To produce inter-annotator agreement metrics, 108 provisions were randomly selected from the dataset and annotated independently again by one of the authors who is a legal expert. We chose to use these 108 provisions as the test set to minimize errors in the evaluation data.

Table 2.2: Deontic Modality Distribution

Obligation	Entitlement	Permission	Prohibition
1712	1860	271	59

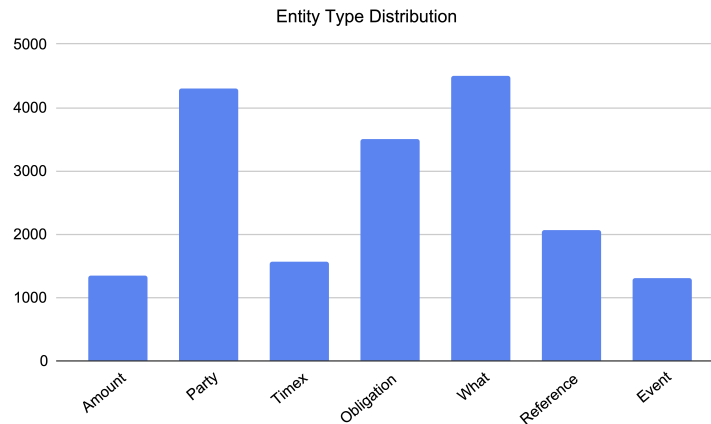


Figure 2.4: Distribution of entity types across the Contract-OLG dataset.

2.4.4 Inter-Annotator Agreement

We measure agreement between the training and test datasets and report micro-averaged precision, recall and F1. We use a hard (exact match) and soft (partial match) metric as defined in Section 2.7.2. The agreement can be found in Tables 2.3 and 2.4 under the row labeled Human.

We found high agreement for entities (Trigger=87.1 and Argument=80.9 F1), though for event arguments there was a 7% gap between the hard and soft metric indicating some challenges with determining argument boundaries. The relations had lower agreement (Role=69.8 and Formula=58.7 F1) though this is partially due to the compounded errors for missed entities. This is still roughly comparable to prior work on general joint entity and relationship detection [Yao et al. 2019].

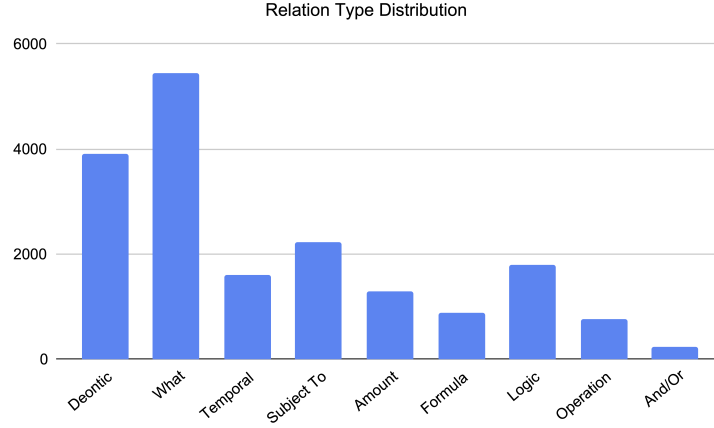


Figure 2.5: Distribution of relation types across the Contract-OLG dataset.

2.5 OLG Extraction Task

Given a contract provision, the task is to extract an OLG representation for each existing obligation. These graphs can be disjoint or, in the case of a logical dependency, be connected via a logical relation. We formulate the OLG extraction task as follows. For each obligation predict a graph $G=(V, E)$ where V is the node set and E is the edge set. Each node $v_i = (t_i, s_i, e_i) \in V$ represents a trigger, argument or entity where t is the node type, and s and e are the start and end token indexes, respectively. Each edge $e_{ij} = (t_{ij}, i, j) \in E$ represents an argument role or relation between nodes, where t is the edge type, and i and j are node indexes for the tail and head, respectively. We further discuss the subtasks of OLG extraction below.

2.5.1 Entity Extraction

This task seeks to detect entity spans (start and end tokens) in the text and classify them into entity types. For example, in Figure 2.1 we would want to classify “Executive Officer” as a party entity and identify the contiguous span of tokens that make up that party name.

2.5.1.1 Obligation and Event Extraction

This task identifies obligation and event triggers (words that most clearly evoke the obligation or event), and their arguments (words that describe event or obligation attributes - e.g. parties, amounts). Triggers and arguments are subsets of entities. Returning to the example in Figure 2.1, more specifically “receive” is an obligation trigger, and “Executive Officer” is an obligation argument.

2.5.2 Relation Extraction

Once entities have been identified in the text, relation extraction predicts relation types between ordered pairs of entities. For example, in Figure 2.1 we would want to assign an entitlement relation from the “receive” entity to the “Executive Officer” entity.

2.5.2.1 Argument Role Extraction

Argument roles are a subset of relations. The argument role extraction task identifies relation types between triggers and arguments. The entitlement relation from Figure 2.1 can be more specifically defined as an argument role since it links an obligation trigger to an obligation argument.

2.5.2.2 Logic and Formula Extraction

Relations can also express logical dependencies between events and obligations, and mathematical operations between entities. This task identifies logical and mathematical relation types between ordered pairs of entities. For example, in Figure 2.1 we need to identify the subtract relation type between the fixed bonus amount (“\$25,000”) and “standard deductions”.

OLG	Accord Model Code	Accord Ergo Code
Executive Officer shall receive a signing bonus of \$25,000, less standard deductions no later than Friday, October 14, 2016. Entities: party1: Executive Officer obligation1: receive what1: signing bonus amount1: \$25,000 amount2: standard deductions timex1: October 14, 2016 Relationships: subtract: amount1, amount2 entitled: obligation1, party1 before: obligation1, timex1 what: obligation1, what1 formula: obligation1, amount1 formula: obligation1, amount2 amount: obligation1, amount1 amount: obligation1, amount2	<pre>asset PaymentContract extends Contract { --> Party party1 o String what1 o DateTime timex1 } transaction clauseRequest extends Request { o Double amount1 o Double amount2 }</pre>	<pre>clause Obligation1Initialize(request : clauseRequest) : Any emits GeneralObligation { enforce isBefore(now(), contract.timex1) else throw failure("Obligation1"); emit GeneralObligation { entitled: [contract.party1], obliged: [], prohibited: [], permission: [], what: [contract.what1], subject_to: [], amount:[request.amount1 - request.amount2], }; return }</pre>

Figure 2.6: Example of code generation approach mapping the OLG extracted from Figure 2.1 to Accord model and logic code snippets. Highlighted fields map entities, underlines map relations.

2.6 Code Generation

To connect back to our motivation for this work (enabling the conversion of natural language agreements to computable contracts) - we build a proof of concept for this idea by converting our OLG into a subset of the Accord programming language [Roche et al. 2021], which is a domain specific language built specifically for legal contracts. It captures the business logic for contracts and contains existing data structures such as clauses, obligations, parties, that make it well suited for our effort. It is made up of three parts:

- **Templated Text:** Accord templatizes the text, representing the key entities and elements as computational types (e.g. Strings, Numeric, Parties, DateTime) using markup.

- **Data Model (Concerto):** Accord uses an object oriented model to represent schema for the clause. In particular you define a set of request functions that act as an interface between external events and the clause logic, and a clause object that contains the attributes defined in the templated text.
- **Logic (Ergo):** Accord represents the logic for the agreement as a separate code block, where you have if/then statements or can perform computation over the variables. This is also where you emit obligations or respond to requests.

For conversion from our OLG structure to code we take inspiration from the scratch programming language [Resnick et al. 2009] that converts blocks and relationships directly to code snippets. For example obligation triggers map directly to obligation emissions, event triggers to requests, obligation arguments to String (e.g. what), amounts to Double, party to Party, or timex to DateTime. The argument relationships define the object oriented structures for representing the code within the Concerto data model.

Once the Concerto data model is defined, the temporal, logical and formula relationships between obligations, events, and their arguments define any specialized logic or failure constraints that may need to be applied, and ultimately emits any obligations. For example, logical dependencies are converted to if/then statements, temporal dependencies are implemented as constraints, formulas to math equations and all obligations are emitted conditioned on these dependencies. The Accord language does not directly provide support for references so we treat them as arguments for our obligations.

To simplify the construction of these clauses we do not allow the Accord code to maintain states in the agreement and rather assume the system maintains state externally and is passed in through requests. Additionally we create an Accord clause for each

obligation. Finally we do not use responses to return the clause result, but rather limit to emitting new obligations or failures. To simplify the code construction we also assume any unresolvable variables, like standard deductions, are passed to the system. Figure 2.6 demonstrates how our OLG representation can directly map to Accord code using our running example.

2.7 Experiments

2.7.1 Entity and Relationship Models

We experiment on the OLG extraction task using a variety of state-of-the-art NLP models, including two extractive language models, SpERT [Eberts and Ulges 2019] and OneIE [Y. Lin et al. 2020], and a generative language model, GPT-3 [Brown et al. 2020]. We also break apart the OLG extraction task and experiment on entity extraction and relation extraction as separate tasks using BERT NER [Devlin et al. 2018] and SAIS [Xiao et al. 2021], respectively. For all models, except GPT-3, we experiment with two variants for the backbone: BERT Base and Legal BERT [Chalkidis, Fergadiotis, et al. 2020]. All models were fine-tuned on Contract-OLG using a single NVIDIA Titan V GPU.

2.7.1.1 SpERT [Eberts and Ulges 2019]

A Transformer language model that has demonstrated state-of-the-art performance on joint entity and relation extraction. We fine-tune SpERT for 80 epochs with a learning rate of $5e-5$ and weight decay of $1e-2$.

2.7.1.2 OneIE [Y. Lin et al. 2020]

A graph-based information extraction model that has demonstrated state-of-the-art performance on event, entity and relation extraction. We fine-tune OneIE for 50 epochs with a learning rate of $5e-5$ and weight decay of $1e-5$ for BERT parameters, and a learning rate of $1e-4$ and weight decay of $1e-3$ for other parameters.

2.7.1.3 GPT-3 [Brown et al. 2020]

Recently, generative prompt tuning approaches have shown state of the art performance for entity and relationship extraction due to pre-training. We follow the approach of DeepStruct [C. Wang et al. 2022] to generate entities and relationships using the prompts: "(Entity) instance of (Entity type)" and "(Entity A);(relationship); (Entity B)" sorted alphabetically. We use OpenAI for GPT-3 to build this model and train it for 4 epochs using default parameters. During inference we run with a temperature of 0 and a max generated length of 1000 tokens. This model does not output spans so it was only evaluated using the soft-metric.

2.7.1.4 BERT NER [Devlin et al. 2018]

A BERT model optimized for token classification for named entity recognition. We fine-tune BERT NER for 30 epochs with a learning rate of $1e-4$ and weight decay of $1e-5$ and perform inference by averaging token predictions.

2.7.1.5 SAIS [Xiao et al. 2021]

A language model that uses evidence based data augmentation and ensemble inference to achieve state-of-the-art performance on relation extraction. We fine-tune SAIS for 30 epochs with a learning rate of $5e-5$ for BERT parameters, and a learning rate of $1e-4$ for

other parameters.

2.7.2 Evaluation

We report micro-averaged precision, recall and F1 scores for all tasks, which are commonly used for entity and relationship tasks. We report both hard and soft variants of these metrics as defined below:

- **Hard Metric:** Entities must have the same type and span. Relations must have the same type and linked entities must match exactly.
- **Soft Metric:** Similar to prior work [Grishman and Sundheim 1996], credit is given for partial string overlap. Entities must have the same type, credit is given for partial overlaps if a sequence of greater than 50% of two entity strings match. Relations must have the same type and entities must pass the soft metric.

The hard metric measures whether entities and relations match exactly, while the soft metric measures whether entities and relations are semantically similar. The soft metric is meaningful because some entities, like party names, are often mentioned multiple times in a sentence. The soft metric finds agreements when annotators select the same party name, even if the spans do not match, since the selected entities are semantically equivalent. Additionally, algorithms were sometimes penalized for including adjectives or other articles, even though both predictions may have been acceptable.

2.7.3 Entity Extraction Performance

We report experiment results for the entity extraction task in Table 2.3. BERT NER (base) outperformed other models in the trigger extraction task for both hard (71.3 F1) and soft

Model	Trigger (Hard)			Trigger (Soft)			Argument (Hard)			Argument (Soft)		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SpERT _{base}	74.6	65.4	69.7	82.0	71.9	76.6	70.1	70.1	70.1	76.6	76.6	76.6
SpERT _{legal}	69.6	65.7	67.6	78.4	74.1	76.2	64.9	71.7	68.1	70.5	77.9	74.0
OneIE _{base}	55.4	68.2	61.1	61.9	76.2	68.3	70.2	71.1	70.6	77.6	78.6	78.1
OneIE _{legal}	55.7	67.9	61.2	61.8	75.3	67.9	71.9	72.2	72.0	78.7	79.0	78.9
BERT	75.5	67.6	71.3	83.8	75.0	79.2	69.8	72.5	71.1	76.1	79.0	77.5
NER _{base}												
BERT	74.2	64.8	69.2	82.7	72.2	77.1	71.1	71.4	71.3	78.7	79.0	78.9
NER _{legal}												
GPT-3 _{curie}	-	-	-	72.0	57.1	63.7	-	-	-	61.2	58.9	60.1
GPT-3 _{davinci}	-	-	-	78.3	67.0	72.2	-	-	-	76.5	60.0	67.2
Human	89.3	85.0	87.1	93.9	89.4	91.6	80.8	81.0	80.9	86.9	87.2	87.1

Table 2.3: Precision, recall and F1 reported as percentages for entity extraction task on the Contract-OLG dataset. Subscript *base* is for models using BERT base, *legal* is for Legal BERT. See Section 2.7.2 for a description of hard and soft evaluation metrics.

(79.2 F1) metrics. OneIE (legal) was the best performing model for argument extraction using the hard metric (72.0 F1), and tied BERT NER (legal) for the soft metric (78.9 F1). Since the GPT3 approach does not involve token spans, we cannot evaluate using the hard metric, and thus report only soft metrics for all GPT-3 results. While the GPT-3 results are generally lower than the other models, we consider these results to be promising considering the other models were specifically designed for information extraction tasks. While our baseline models show promise, we still observe considerable gaps between machine and human performance across all entity extraction tasks, indicating there is still significant room for improvement in modeling these tasks.

2.7.4 Relation Extraction Performance

We report experiment results for the relation extraction task in Table 2.4. For the argument role extraction task, SpERT (base) outperformed other models in both hard (47.1 F1)

and soft (54.3 F1) metrics. OneIE was the best performing model for formula and logic extraction for both hard (34.6 F1) and soft (43.2 F1) metrics (base variant and legal variant, respectively). Compared to the entity extraction task in Table 2.3, the relation extraction tasks proved to be more challenging for the baseline models, especially formula and logic relations. This is partly due to error propagation - in order for a relation to be predicted correctly, both entities in the relationship must first be predicted correctly. Formulas in particular have the largest gap between human and machine performance, which could be due to there being almost 4 times fewer training points than argument roles. To better understand relation extraction as a separate task we also experiment with SAIS using oracle entities. SAIS shows limited improvement on role extraction and is outperformed by the two-stage models in formula/logic extraction indicating that this task is challenging and may require improved modeling or a larger dataset. Note that SAIS models were not considered in determining the best performing models above since the use of oracle entities gives them an advantage over other approaches.

2.7.5 Contract Pre-training

Unlike prior work that found pre-training in-domain to provide a few point performance boost for NER tasks [Chalkidis, Fergadiotis, et al. 2020], we did not observe significant gains on our dataset. Despite being optimized on a corpus of legal documents, the performance of Legal BERT variants on entity extraction tasks is comparable to BERT base variants, sometimes increasing by a few points and other times dropping by a similar margin. We suspect it may be due to legal BERT being an uncased model, while BERT-base was cased.

Model	Argument Role (Hard)			Argument Role (Soft)			Formula / Logic (Hard)			Formula / Logic (Soft)		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SpERT _{base}	47.9	46.4	47.1	55.1	53.4	54.3	30.7	34.4	32.5	35.6	39.9	37.6
SpERT _{legal}	41.7	48.0	44.7	48.8	56.1	52.2	20.5	32.2	25.1	25.7	40.4	31.4
OneIE _{base}	45.0	40.4	42.6	52.6	47.2	49.8	35.4	33.9	34.6	40.6	38.8	39.7
OneIE _{legal}	48.3	42.1	45.0	56.4	49.2	52.6	33.7	31.1	32.4	45.0	41.5	43.2
GPT-3 _{curie}	-	-	-	45.0	28.6	35.0	-	-	-	22.9	21.3	22.1
GPT-3 _{davinci}	-	-	-	50.2	33.1	39.9	-	-	-	28.7	26.0	27.3
SAIS _{base} ^{oracle}	57.7	53.8	55.7	60.5	56.4	58.4	31.3	35.3	33.2	32.3	36.4	34.2
SAIS _{legal} ^{oracle}	53.3	47.3	50.1	56.0	49.7	52.7	23.5	29.0	25.9	23.9	29.5	26.4
Human	69.5	70.0	69.8	76.0	76.7	76.3	53.2	65.3	58.7	60.1	74.6	67.1

Table 2.4: Precision, recall and F1 reported as percentages for relation extraction task on the Contract-OLG dataset. Subscript *base* is for models using BERT base, *legal* is for Legal BERT. Superscript *oracle* is for models trained and evaluated using oracle entities. See Section 2.7.2 for a description of hard and soft evaluation metrics.

2.8 Limitations

While we capture a range of obligation types in this work, the OLG was designed to best represent payment related obligations. The provisions in our dataset have been specifically filtered to contain a high concentration of payment related obligations (see Section 2.4.1). There is no guarantee our experiment results would generalize to all contract provisions. We hope to explore a more generalized version of the OLG in future work.

2.9 Chapter Summary

In this chapter we present the first system and task for converting natural language contracts to a computable representation through the extraction of obligation logic graphs. We show how the OLG schema defines a meaningful representation of entities and rela-

tions to capture the intent of an obligation, and present a custom interface for creating OLG annotations. To enable modeling this new OLG extraction task, we also introduce the Contract-OLG dataset consisting of 1,876 contract provisions, 18,597 entities and 18,170 relationships. Our analysis of state-of-the-art NLP models shows that there is still room for improvement to close the gap between human expert and machine performance for several OLG extraction subtasks, especially formula extraction. We intend to release the Contract-OLG dataset with the hopes this work will bring the field of computational law and the broader NLP community closer together and motivate future research.

2.10 From Structured Extraction to Structured Reasoning

The OLG framework demonstrates that legal language becomes more tractable when its logical components are made explicit. By decomposing contract provisions into structured entities, relations, and formulas, this chapter showed how symbolic representations can help bridge the gap between natural language and contract logic. Our experiments also highlight the emerging capabilities of generative models. While task-specific models like SpERT and OneIE performed best overall, GPT-3 produced competitive results despite not being designed specifically for graph-based extraction. This performance demonstrated the nascent promise of generative language models to grasp complex legal semantics without the need for highly engineered, specialized architectures.

This observation motivates the next chapter of the dissertation. Rather than focusing on extracting symbolic structure from legal text, the following chapter considers how to improve reasoning within generative language models themselves. If general-purpose models can already approximate aspects of structured legal analysis, then a natural next question is whether they can be guided to reason more effectively through legal rules. In

particular, Chapter 3 explores whether prompting models to decompose complex rules into elements can improve rule-based legal reasoning. The dissertation thus shifts from imposing structure on legal text to imposing structure on how language models reason about that text.

Chapter 3

Rule-Based Reasoning with Large Language Models

3.1 Rule-Based Reasoning

The surge in reasoning capabilities of language models (LMs) has the potential to transform the legal industry, a sector inherently reliant on sophisticated reasoning. The development of models that can assist and validate the reasoning processes of legal practitioners promises to usher in a new legal era. Such advancements would not only enhance the efficiency of legal services but also expand the capacity of professionals to service more clients, thereby broadening access to justice. At present, language models are prone to hallucinations in a legal setting [[Dahl et al. 2024](#)], and even powerful LMs like GPT-4 struggle to perform basic legal tasks [[Blair-Stanek, Holzenberger, and Durme 2023](#)].

Legal tasks typically require sophisticated rule-based reasoning. These rules are written in natural language and expressed in many forms, including statutes, judicial holdings and even contract provisions. Similar to an if/then statement, a rule has an antecedent (a condition that can be evaluated to true or false) and a consequent (the outcome triggered if the antecedent is satisfied). Even the earliest recorded laws, written in

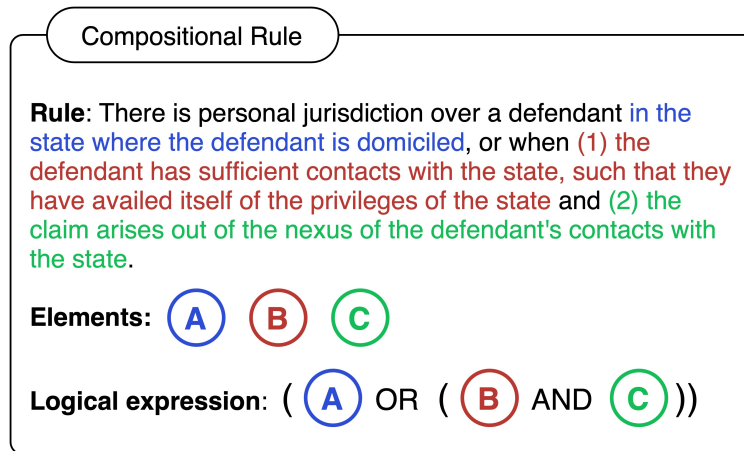


Figure 3.1: Example showing compositional structure of rule for Personal Jurisdiction task from LegalBench. Color coding is used to identify rule elements and illustrate how these elements form a complex logical expression. Reasoning about compositional rules requires not only correctly applying each element to a fact pattern, but also resolving the logical expression. If the logical expression evaluates to true, it triggers a consequence (personal jurisdiction exists).

Sumerian on clay tablets, have used this conditional structure [Roth 1995]. Rule-based reasoning allows us to draw conclusions by applying a rule to a set of facts to determine if these preconditions are satisfied. For example, if parking is prohibited (consequent) between 2pm and 4pm (antecedent), and we know it is currently 3pm, then we can conclude that parking is currently prohibited. Often rules, especially complex rules, are compositional in nature meaning the antecedent consists of multiple conditions joined by *and* and *or* operators forming a complex logical expression (see Figure 3.1). In law, these constituent conditions are called rule elements. Recent work demonstrates that models can struggle to reason about even basic compositional rules [Blair-Stanek, Holzenberger, and Van Durme 2023].

We explore large language models as rule-based reasoners, evaluating several common in-context learning methods across 8 tasks and 3 distinct rules from LegalBench, a collaboratively constructed legal reasoning benchmark [Guha, Nyarko, Ho, Ré, et al. 2023]. LegalBench tasks were designed by subject matter experts to evaluate useful le-

gal reasoning skills. Prior work on LegalBench has focused on evaluating the capacity of models to perform rule-based reasoning given in-context demonstrations of the same rule being applied to varying fact patterns. These few-shot prompts are valuable in law where annotated data is scarce, limiting the ability to fine-tune models for specific legal tasks. Yet requiring several reasoning examples for each rule in a real world setting poses a challenge in terms of both cost and scalability. Additionally, a same-rule setting allows the model to largely mimic the reasoning path of the in-context examples. In contrast, a different-rule setting enables us to examine a model’s ability to dynamically generate this reasoning path - the same way a lawyer would reason. Our work shifts focus to evaluating and improving a model’s ability to perform rule-based reasoning given only a *single* in-context demonstration of a *different* rule application, thus removing the need to store examples for each rule.

In this work, we introduce **Chain of Logic**, a new prompting approach to guide LMs in reasoning about compositional rules by decomposing rule elements into a series of logical statements (evaluate to true or false), before explicitly recomposing these element answers to resolve the underlying logical expression. Understanding natural language questions requires understanding how to break down a question into a set of steps yielding an answer [Wolfson et al. 2020]. The inspiration for our method is rooted in the IRAC Framework, an approach to legal reasoning where lawyers break down the reasoning process into four sequential steps: issue spotting, rule identification, rule application and conclusion. During the rule application step, rules are typically divided into their elements and addressed separately before recombining these threads to arrive at a conclusion. In devising our chain of logic approach, our interdisciplinary team combined this domain knowledge with a variety of insights from prior work on in-context learning.

Our experiments show chain of logic outperforms other prompting methods, including chain of thought [Wei et al. 2023] and self-ask [Press et al. 2023], across a variety of rule-based reasoning tasks using both open-source and commercial models. We demonstrate that given a single example of chain of logic, a model can learn to generalize this approach to a different rule and fact pattern, and thereby improve its reasoning capabilities.

3.2 Prompting Methods for Reasoning

Large language models have demonstrated strong capabilities as zero-shot [Kojima et al. 2023] and few-shot [Brown et al. 2020] reasoners. Chain of thought prompting enhanced problem solving performance even further by eliciting models to produce intermediate reasoning steps [Wei et al. 2023]. This approach proved effective at solving complex tasks, including arithmetic and commonsense reasoning tasks. Many other prompting methods followed with a similar aim of decomposing a complex task into a sequence of simpler subtasks. Self-ask improved multi-hop question answering performance beyond chain of thought prompting by guiding the model to explicitly pose and answer intermediate questions [Press et al. 2023]. Similar to decomposing a multi-hop question into a sequence of intermediate questions, compositional rules can be decomposed into rule elements. However, reasoning over these rules requires not only answering each element correctly, but also understanding the logical relationships that exist between these elements. For example, if a rule with two elements resolves to (true, false), then the final answer depends on whether the boolean relationship between those elements is *and* or *or*. In Section 3.5, we show existing prompting methods can resolve rule elements correctly while still getting the final answer incorrect. This motivates the need for a more robust prompting method which attends to both the element level answers and the underlying

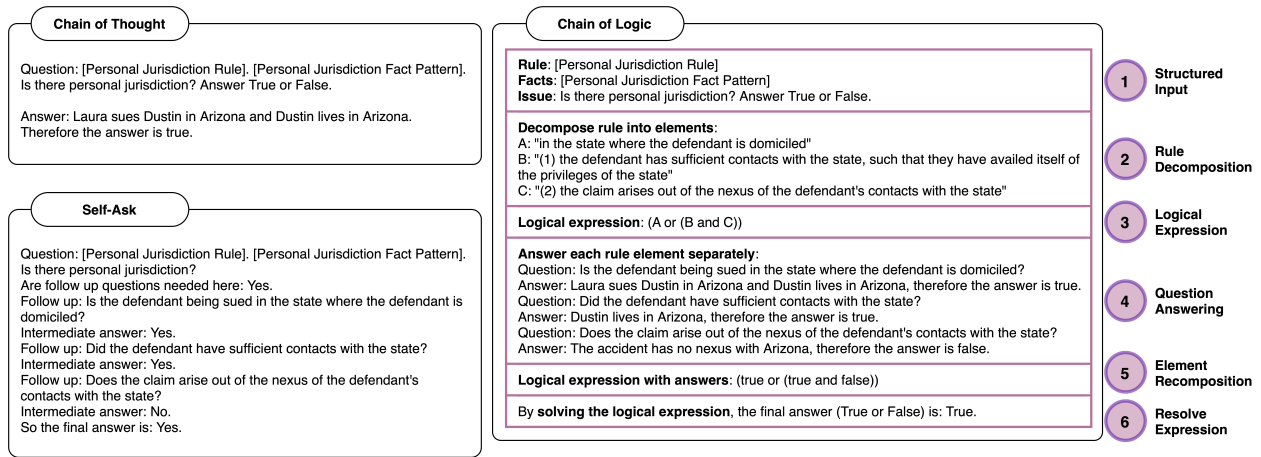


Figure 3.2: Comparing one-shot examples demonstrating chain of thought, self-ask and our chain of logic method on the Personal Jurisdiction task. Through a sequence of reasoning steps, chain of logic decomposes a rule into elements which are solved independently, before recomposing sub-answers to arrive at a final conclusion. See Section 3.3 for a detailed discussion on the chain of logic approach.

logical structure. Further motivating this work, zero-shot methods have been observed outperforming one-shot and few-shot approaches on legal reasoning tasks [Blair-Stanek, Holzenberger, and Van Durme 2023; Yu et al. 2022], suggesting models can struggle with in-context learning in a legal setting.

3.3 Chain of Logic

We introduce chain of logic, a new prompting approach to elicit rule-based reasoning in LMs through a series of instructive reasoning steps. Each step in this series helps inform the next, and enables the model to unravel the many reasoning tasks needed to arrive at the right conclusion. Our method builds on chain of thought and self-ask by not only considering problem decomposition, but also attending to relationships between subtasks that can dictate the final answer, which we later show improves performance. Our prompt begins with a single example demonstrating the chain of logic approach, appended with the inference-time problem. Figure 3.2 contains a one-shot example of our

approach depicting the sequence of reasoning steps which we describe below:

- **Step 1: Structured Input.** Clearly label the task inputs to identify rule, facts and issue, similar to the IRAC Framework.
- **Step 2: Rule Decomposition.** Decompose the rule into elements by identifying relevant text spans and assigning elements to variables. Similar to the least-to-most approach [Zhou et al. 2023], our method first identifies the subtasks before trying to solve them.
- **Step 3: Logical Expression.** Construct a complex logical expression representing the logical relationships that exist between these element variables.
- **Step 4: Question Answering.** Iterate through each rule element, rephrasing the element as a question before providing a rationale and answer. Like self-ask, we dynamically construct this series of sub-questions, but for our approach this construction is informed by the elements identified in step 2. The model needs only rephrase each element as a question.
- **Step 5: Element Recomposition.** Replace element variables in the logical expression with the corresponding sub-answers from the previous step.
- **Step 6: Resolve Expression.** Resolve the logical expression populated with sub-answers to arrive at the final solution.

During inference, the model performs each step in this process automatically given only a one-shot example and the inference task inputs (formatted as described in step 1). Since the one-shot task inputs (rule, facts, issue) are different from the task inputs being tested, the model is reasoning on its own at each step, including deciding how many rule

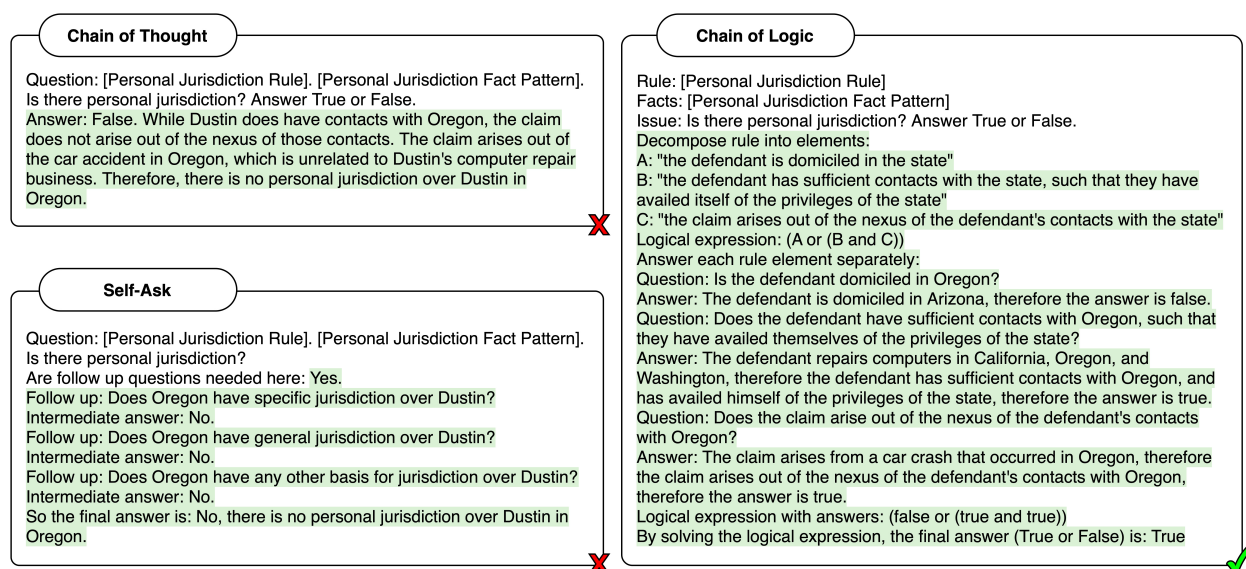


Figure 3.3: Comparing GPT-3.5 output for chain of thought, self-ask and our chain of logic method on the same Personal Jurisdiction task. Chain of logic prompting elicits large language models to reason about complex rules while also constructing an interpretable reasoning path. The prompts here are abridged, omitting a one-shot example for each method from the Diversity Jurisdiction task (see Section 3.4.1).

elements exist, the text span of each element and the logical relationships between them. This more closely resembles a real world setting where lawyers apply the IRAC framework to a rule and fact pattern being observed for the first time.

The intermediate steps in our approach provide a detailed accounting of each factor contributing to the final conclusion (see Figure 3.3 for an example of model output). This allows for transparent and explainable decision-making, which is particularly vital in the legal domain where the justification for an answer can be as important as the answer itself. Chain of logic also enables us to debug inaccurate conclusions by retracing the reasoning path to locate errors in rule application and logic.

3.4 Experiments

We conduct a series of experiments to compare our proposed chain of logic approach with existing prompting methods, and to evaluate the rule-based reasoning capabilities of LMs in a different-rule setting. While we sometimes observe an inverse relationship between prompt complexity and performance (simpler prompts perform better), we find that chain of logic consistently outperforms other prompting methods across a range of rule-based reasoning tasks.

3.4.1 Tasks

LegalBench [Guha, Nyarko, Ho, Ré, et al. 2023] is a legal reasoning benchmark designed and constructed by an interdisciplinary team of computer scientists and legal professionals. The tasks in this benchmark have been designed by subject matter experts to measure legal reasoning capabilities that are both interesting and useful. We consider 8 rule-based reasoning tasks involving 3 distinct compositional rules from this benchmark:

- **Personal Jurisdiction** (1 task): This task involves determining whether a court has the authority to preside over a dispute based on where the defendant is domiciled, and whether the defendant had sufficient contacts with the forum state and the claim arose out of the nexus of those contacts. There are 50 test samples for this task.
- **Diversity Jurisdiction** (6 tasks): This task involves determining whether a federal court can preside over a lawsuit pertaining to state law based on whether complete diversity exists (no plaintiff and defendant are citizens of the same state) and the amount in controversy exceeding \$75,000. There are 6 diversity jurisdiction tasks

with increasingly complex fact patterns, from Diversity Jurisdiction 1 (easiest) to Diversity Jurisdiction 6 (hardest). There are 300 test samples for each diversity jurisdiction task.

- **J.Crew Blocker** (1 task): This task involves determining whether a provision from a loan agreement contains a J.Crew Blocker restrictive covenant based on whether the provision prohibits transferring IP to an unrestricted subsidiary or requires lender consent for IP transfers to a subsidiary. There are 54 test samples for this task.

Each sample from all tasks contains a rule, fact pattern and question. To correctly answer the question, the rule must be applied to the fact pattern. The prompt for each task contains a one-shot example from another rule, followed by the test sample. The benchmark expects an answer of yes or no for each question. For our method we reformat the answer to true or false to more closely resemble formal logic. If the model’s response is ambiguous, we use a second prompt "Therefore the answer (true or false) is", following Kojima et al. [Kojima et al. 2023].

3.4.2 Baseline Methods

In addition to our chain of logic approach (see Section 3.3), we examine six prompting methods. We first explore zero-shot prompting where the prompt includes only the test sample (rules, facts, question) with no in-context demonstrations. Blair-Stanek et al. showed that zero-shot prompting can outperform few-shot methods for some legal reasoning tasks, even with chain of thought reasoning [Blair-Stanek, Holzenberger, and Van Durme 2023]. This suggests some models could have difficulty learning through in-context demonstrations in a legal setting. We include zero-shot prompting in our experiments to further explore this hypothesis. For completeness, we also include two zero-shot meth-

ods designed for legal reasoning tasks from prior work. Legal syllogism (LS) prompting is a zero-shot approach to legal judgement prediction where the prompt first defines legal syllogism before instructing the model to perform syllogistic reasoning [C. Jiang and Yang 2023]. Legal reasoning (LR) prompting [Yu et al. 2022] is also a zero-shot method where the prompt includes a simple approach description: "Approach: Issue, rule, application, conclusion".

For the remaining methods, we use a one-shot approach where a single demonstration with correct answer is included in the prompt. First, we use standard prompting [Brown et al. 2020] where the in-context demonstration includes only the sample and answer. Second, we explore chain of thought [Wei et al. 2023] prompting which includes a rationale written by a legal professional explaining the relevant reasoning before the demonstration answer. Last, we include the self-ask approach [Press et al. 2023] where the one-shot example demonstrates explicitly asking and answering an intermediate question for each rule element before the final answer. See Figure 3.2 for an illustration of these one-shot examples. While self-ask was originally evaluated on another compositional reasoning task, multi-hop question answering, its demonstrated ability to increase reasoning capacity by decomposing complex problems makes it a compelling method to explore in this work.

3.4.3 Language Models

We experiment with two commercial models, GPT-3.5-Turbo and GPT-4 [OpenAI 2023], and two leading open-source models, Llama-2-70b-chat [Touvron et al. 2023] and Mistral-7B-OpenOrca, which is a Mistral-7b [A. Q. Jiang et al. 2023] model fine-tuned with Orca [Mukherjee et al. 2023]. We select open-source models of disparate size to investigate the relationship between model size and rule-based reasoning abilities. For all language

models we set the temperature to 0.0 to make the output more focused and reproducible, and otherwise use default settings.

3.5 Results and Discussion

Table 3.1 presents results for baseline methods and our chain of logic approach macro-averaged across all rule sets (Personal Jurisdiction, Diversity Jurisdiction and J.Crew Blocker), using commercial and open-source models. We report the macro-average here to provide a balanced view across rule sets, though as shown later in Figure 3.4 and the appendix, Chain of logic performs best on the Diversity Jurisdiction task, which is downweighted with this metric. See the Appendix for more detailed model performance by rule. For each model, a zero-shot method is the best performing baseline. This is consistent with prior work finding that zero-shot methods can outperform one-shot and few-shot prompting on legal reasoning tasks [Blair-Stanek, Holzenberger, and Van Durme 2023; Yu et al. 2022]. Chain of logic significantly improves rule-based reasoning through in-context learning.

For each model, chain of logic outperforms all baseline methods, including zero-shot methods. The performance gap between chain of logic and the best performing baseline is 3.9% on average. This gap is smaller for open source models (0.3% and 0.4%) and larger for commercial models, with GPT-3.5 having the largest performance improvement of 10.7%. For this work we observe a large performance difference between the commercial and open source models, and hypothesize the performance gains may be larger for the stronger commercial models which can more easily follow the longer prompts. This can also be seen in the performance differences between the baseline zero and one-shot methods. For GPT-4 we believe the absolute performance gain may be lower due to ceil-

	GPT-3.5	GPT-4	Llama-2	Mistral	Average
Zero-Shot	76.3	90.4	74.2	60.8	75.4
Zero-Shot-LR [Yu et al. 2022]	57.2	90.6	65.3	62.8	69.0
Zero-Shot-LS [C. Jiang and Yang 2023]	75.5	90.1	58.4	52.2	69.1
Standard Prompting [Brown et al. 2020]	72.6	88.6	65.1	45.5	68.0
Chain of Thought [Wei et al. 2023]	70.1	89.1	71.8	46.4	69.4
Self-Ask [Press et al. 2023]	68.2	86.3	67.5	46.5	67.1
Chain of Logic (ours)	87.0	92.3	74.6	63.1	79.3

Table 3.1: Aggregated performance (accuracy) across all rules, including Personal Jurisdiction, Diversity Jurisdiction and J. Crew Blocker, using both open-source and commercial language models. Chain of logic consistently outperforms other prompting methods. All methods are one-shot, except zero-shot approaches.

ing effects of the model. The specialized zero-shot legal baselines demonstrate comparable performance to zero-shot for some models, but are also inconsistent. LR prompting slightly outperforms zero-shot using Mistral-7b and GPT-4, but also underperforms by 19.1% for GPT-3.5.

The one-shot baselines demonstrate the challenges to in-context learning in this setting by consistently underperforming zero-shot methods. The model output depicted in Figures 3.3 and 3.5 illustrates common errors we observe in rule-based reasoning using chain of thought and self-ask. First, chain of thought often generates fluent but illogical reasoning paths. The output in Figure 3.5 shows the model incorrectly concludes that personal jurisdiction does not exist because Dustin plans to move to New York, even though future residency is not an element of the rule. Second, the question/answer pairs generated by the self-ask approach commonly decompose the rule incorrectly (Figure 3.3) or incompletely (Figure 3.5). The output in the latter figure shows the model only asks inter-

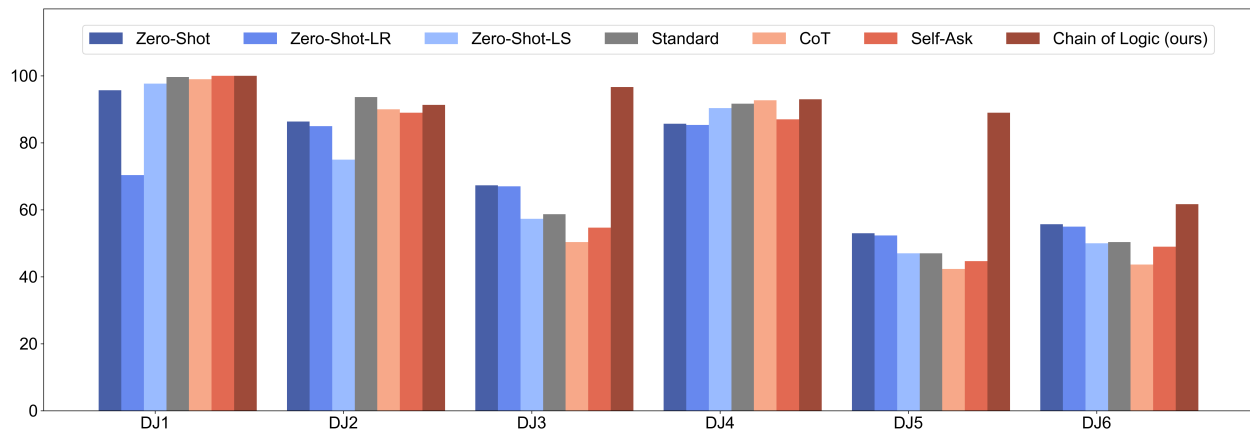


Figure 3.4: Accuracy (%) across all 6 Diversity Jurisdiction tasks using GPT-3.5. The fact patterns in these tasks are increasingly complex, from DJ1 (easiest) to DJ6 (hardest). Chain of logic particularly outperforms other prompting methods for tasks requiring arithmetic operations (DJ3, DJ5, DJ6). See Section 3.5.1 for a detailed discussion.

mediate questions for two out of three rule elements, omitting whether the defendant had sufficient contact with the forum state. Chain of logic prevents these omissions by first restating the rule elements (step 2 in the approach), similar to how System 2 Attention [Weston and Sukhbaatar 2023] first restates only the relevant portions of a context before attending to a task. In the ablation study, we show that removing this step significantly decreases performance. Last, Figure 3.5 also demonstrates how not recognizing logical relationships can lead to an incorrect solution, even when sub-questions are answered correctly. The model correctly determines the defendant is domiciled in the forum state, which is sufficient to conclude personal jurisdiction exists, but still reaches the wrong final answer.

3.5.1 Diversity Jurisdiction Series

The Diversity Jurisdiction series gives us a unique opportunity to examine how performance is affected by task complexity. Recall there are 6 tasks involving the diversity jurisdiction rule. The fact patterns in these tasks increase in complexity across two dimen-

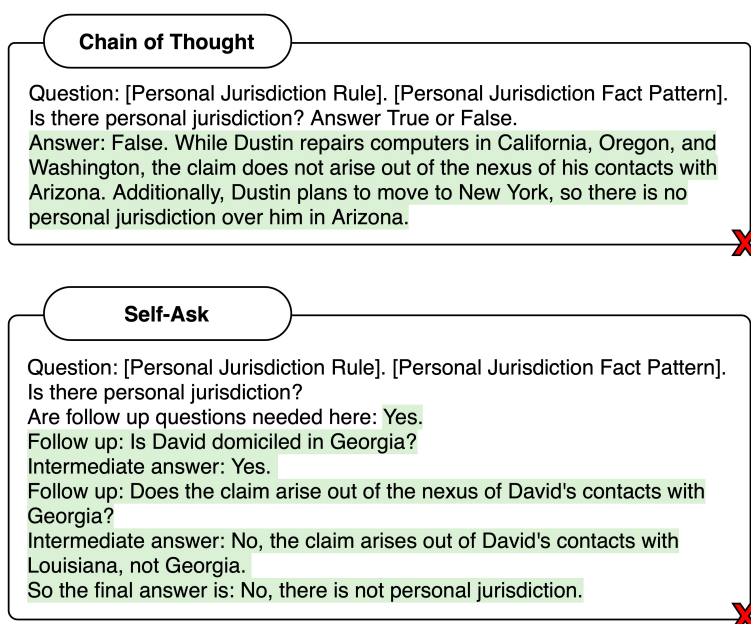


Figure 3.5: GPT-4 output demonstrating common errors in one-shot methods.

sions: 1) the number of parties (plaintiffs and defendants), and 2) the number of claims per plaintiff-defendant pair. The simplest task, Diversity Jurisdiction 1 (DJ1), contains one plaintiff, one defendant and one claim. The most complex task, Diversity Jurisdiction 6 (DJ6), contains two plaintiffs, two defendants, and 4 total claims.

Figure 3.4 shows results across all 6 Diversity Jurisdiction tasks using GPT-3.5. All methods, except zero-shot-LR, perform above 95% accuracy on DJ1, yet even the best performing method, chain of logic, only scores a 61.7% accuracy on DJ6. Chain of logic outperforms the baseline methods for 5 out of 6 tasks, with standard prompting outperforming our method by 2.3% for DJ2. Most notably, chain of logic significantly outperforms other methods on tasks requiring arithmetic reasoning to add dollar amounts from multiple claims (DJ3, DJ5, DJ6). The performance gap between chain of logic and zero-shot, the best performing baseline, is 36% for DJ5. All baselines perform near random chance for this task. For tasks involving arithmetic reasoning, the baselines also demonstrate an

Reasoning Step	PJ
Step 1: Structured Input	-25.9
Step 2: Rule Decomposition	-12.9
Step 3: Logical Expression	-14.8
Step 4: Question Answering	-49.3
Step 5: Element Recomposition	-5.5
Step 6: Resolve Expression	-3.7

Table 3.2: **Ablations.** Reduction in performance (absolute percentage) observed when removing the specified reasoning step from the chain of logic approach using GPT-3.5 on the Personal Jurisdiction task.

inverse relationship between prompt complexity and performance. That is to say, simpler prompts tend to perform better. Chain of logic is the clear exception to this trend being both the most complex prompt and the best performing model.

3.5.2 Ablation Study

Our experiments show chain of logic improves in-context learning for rule-based reasoning tasks. In this section, we assess the contribution of each reasoning step to the overall performance as shown in Table 3.2. We isolate these effects by removing a single reasoning step at a time. We evaluate on the most complex rule, personal jurisdiction, using the model with the best comparable performance, GPT-3.5. Unsurprisingly, removing step 4 drastically reduces overall performance since there is no clear path between the logical expressions in steps 3 and 5. Notably, the removal of step 1, switching from structured to unstructured task inputs, leads to a 25.9 absolute percentage point performance decrease. We believe this structured content is useful for the rule decomposition step that follows. Similarly, the ablations show that decomposing the rule into elements (step 2: -12.9%) and generating a logical expression representing the relationships between elements (step 3: -14.8%) are both critical to the performance gains we observe for this approach.

3.6 Related Work

There has been extensive work on improving the problem solving capabilities of LMs by enabling a model to use more computation for more complex tasks. Graves introduces an adaptive algorithm for dynamically selecting the number of computation steps a recurrent neural network should take based on the task complexity [Graves 2017]. Ling et al. first demonstrated the use of answer rationales, natural language explanations generated before the solution, to solve algebraic word problems [Ling et al. 2017]. Several approaches followed this work in exploring the use of intermediate reasoning steps [Nye et al. 2021; X. Wang et al. 2023; Wei et al. 2023]. More similar to our work, other methods have explored decomposing compositional questions into simpler sub-questions using supervised models [Khot et al. 2021; Min et al. 2019; Qi et al. 2019] and prompting techniques [S. Mishra et al. 2022; Press et al. 2023]. Talmor and Berant also recognized the importance of explicitly identifying relationships between sub-questions [Talmor and Berant 2018]. The authors train a supervised model to translate questions into a computation tree and answer sub-questions by querying a search engine. We extend these works with our decomposition-recomposition approach, exploring simple questions involving compositional rules where the solution depends not only on rule element answers, but also the logical relationships between elements.

Prior work in natural language processing has explored a broad range of legal tasks, including clause classification [Hendrycks et al. 2021] and recommendation [Aggarwal et al. 2021], contract summarization [Manor and J. J. Li 2019], legal judgment prediction [Chalkidis, Androutsopoulos, and Aletras 2019] and even answering bar exam questions [au2 and Katz 2022; Zhong et al. 2019]. Rule-based reasoning can involve rules from many sources. Saeidi et al. explored conversational question answering where the solution re-

quired rule-based reasoning about a collection of regulations [Saeidi et al. 2018]. Servantez et al. captured rules in contract text through graph-based extraction and converted them into code [Servantez, Lipka, et al. 2023]. Similar to our work, Holzenberger and Van Durme introduced an approach to reasoning about tax statutes by decomposing the reasoning process [Holzenberger and Van Durme 2021]. This approach first extracts key arguments (entities, dates, dollar amounts) from the statute text and fact pattern using fine-tuned BERT models, before arriving at a final answer. We do not explore this method since it requires a fine-tuned model and does not easily translate to the tasks in this work.

3.7 Limitations

While LegalBench tasks were crafted by subject matter experts, the rules have been simplified to ensure answers are objectively correct. This greatly improves evaluation, but also means the scope of complexity is narrower than many real world rules. Additionally, our current approach only addresses rules where the solution is based on whether the antecedent has been triggered. Real world rules can contain complex consequences which themselves require some form of reasoning. For example, calculating tax liability after determining the applicable tax bracket.

3.8 Chapter Summary

In this work, we explore causal language models as rule-based reasoners, and show these models can have difficulty learning from in-context demonstrations using common prompting approaches. We present a new prompting method, chain of logic, to elicit rule-based reasoning in language models through decomposition and recomposition. Our experiments show chain of logic consistently outperforms other prompting methods, including

chain of thought and self-ask, across a variety of rule-based reasoning tasks using both open-source and commercial language models. We show how chain of logic creates a coherent and interpretable reasoning path by unraveling the many reasoning steps required by compositional rules. These findings may also be a useful signal for future instruction tuning of language models to imbue them with reasoning. By enhancing the rule-based reasoning capabilities of LMs through in-context learning, chain of logic also reduces the need for annotated legal data, which has historically been a bottleneck for the legal domain

3.9 Toward More Rigorous Evaluation of Legal Reasoning

The results of this chapter show that legal rule application in language models is not fixed, but can be meaningfully improved through more structured reasoning. These findings suggest that language models possess latent capacity for rule-based reasoning that can be elicited through guidance. While chain of logic improved performance across a range of LegalBench tasks, our experiments reveal the importance of analyzing model behavior beyond aggregated benchmark scores. The Diversity Jurisdiction task series provided a unique opportunity to examine model performance across increasingly complex fact patterns. The aggregated results capture a broad performance delta across evaluated prompting methods, but analyzing these tasks individually revealed a more nuanced pattern. The largest performance improvements for chain of logic emerged on tasks requiring mathematical reasoning, such as aggregating claim values to determine whether the amount in controversy exceeded the statutory threshold. This progression across six tasks with increasing complexity allowed us to identify how different reasoning demands affected model performance in ways the aggregate metric alone obscured.

This observation highlights a broader challenge for evaluating legal reasoning. Static benchmarks provide valuable comparisons, but they often offer limited control over the complexity and variation of the reasoning tasks themselves. As a result, it can be difficult to determine precisely which capabilities models possess, where they fail, and how those failures relate to specific reasoning demands. These limitations motivate the next chapter. While Chapter 3 focused on improving rule-based reasoning through structured prompting, the following chapter turns to the question of how such reasoning should be evaluated. Building on the insight that structured variation in task complexity can reveal important differences in model behavior, the dissertation introduces OpenExempt, a dynamic framework for generating legal reasoning tasks designed to enable more controlled, scalable, and diagnostic evaluation of language models.

Chapter 4

Diagnostic Evaluation of Legal Reasoning through Dynamic Task Generation

4.1 Challenges in Benchmarking Legal Reasoning

Language models (LMs) now demonstrate remarkable performance on a wide array of complex tasks, from writing code to passing professional exams. Yet, recent work has begun to question these abilities, probing whether models are truly reasoning or relying on sophisticated forms of memorization and pattern matching [Mirzadeh et al. 2025; Shojaee et al. 2025]. This uncertainty has fueled a critical need for new evaluation methodologies that move beyond static evaluation [Hofmann et al. 2025], providing deeper, more diagnostic insights into the competencies and failure modes of these powerful systems.

This challenge is particularly acute in the legal domain where reasoning demands precision, consistency, and an understanding of intricate, interdependent rules [Servantez, Barrow, et al. 2024]. Consequently, creating benchmarks for legal tasks has historically relied on expert annotation of solutions, a process that is not only expensive and time-consuming but also results in static datasets of fixed question-answer pairs [Guha, Nyarko, Ho, and Ré 2025]. Such datasets struggle to keep pace with rapidly evolving mod-

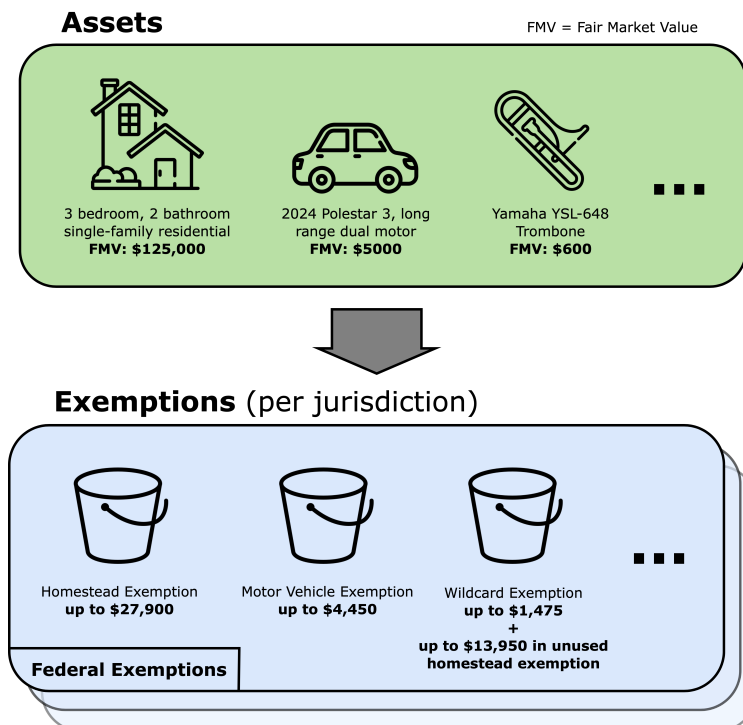


Figure 4.1: OpenExempt tasks center on U.S. bankruptcy law, primarily asset exemption where assets must be assigned to exemption statutes with dollar limits.

els and make it difficult to disentangle the many reasoning skills that a single legal problem may require. A model’s failure on a complex task provides only a single, opaque signal of error. Gaining deeper insight requires a more controlled evaluation approach, one where task complexity and scope can be systematically adjusted to reveal a model’s specific breaking points.

To address these limitations, we introduce OpenExempt, a framework and benchmark constructed through an interdisciplinary collaboration of computer scientists and legal professionals. The **OpenExempt Framework** enables fine-grained control over crafting complex legal reasoning tasks and their solutions on demand. This dynamic approach directly overcomes the constraints of static benchmarks by allowing users to vary many aspects of the task, including case details, jurisdictions, and the scope of the task itself,

thereby enabling the isolation of different types of reasoning. This makes it possible to disentangle performance across distinct reasoning processes, avoiding the conflation of errors that can obscure a model’s capabilities and limitations.

OpenExempt tasks center on the application of federal and state laws governing the exemption of assets under the United States Bankruptcy Code (U.S. Code Title 11)¹. Inspired by computable contracts where natural language clauses are paired with formal, machine-readable logic [Christopher D. Clack et al. 2017; Surden 2012], we combine statute text with structured, symbolic representations of their logic and dependencies, making solutions machine-computable. While our approach also requires legal interpretation, OpenExempt does not rely on direct annotation of task solutions, unlike prior work in dynamic benchmarking [Kiela et al. 2021]. Instead, we use legal knowledge to encode statutes and case assets into structured representations, from which we can construct an immense space of natural language tasks and solutions, removing a key bottleneck in legal reasoning benchmark construction while preserving the accuracy and depth of expert reasoning.

Using this system, we construct the **OpenExempt Benchmark**, a diagnostic benchmark for legal reasoning composed of nine evaluation suites: 6 diagnostic suites and 3 competency suites. Diagnostic suites isolate specific reasoning challenges by varying task complexity across a single axis, such as the number of assets. This allows us to go beyond single points of failure by precisely measuring the performance delta caused by each variation. Competency suites provide a more holistic assessment of a model’s legal reasoning capabilities at three levels of difficulty: basic, intermediate, and advanced.

We present OpenExempt. Our primary contributions are:

¹<https://uscode.house.gov>

1. *OpenExempt Framework*: We present a system capable of creating complex legal reasoning tasks and solutions on demand from expert-crafted encodings of statutes and facts. This framework gives users control over defining task scope and complexity, enabling targeted exploration of a vast problem space and diagnostic evaluation through controlled task variation.
2. *OpenExempt Benchmark*: We introduce a diagnostic benchmark for legal reasoning, consisting of 9 evaluation suites and nearly 10k samples. We perform detailed experiments across a diverse set of 13 language models to surface clear strengths and breaking points.
3. *Open and Extensible Public Release*: We release the OpenExempt Framework² and Benchmark³ to the public under a permissive license (CC BY 4.0). OpenExempt is intended to support further research in both the legal and NLP communities. Its modular architecture makes it possible for either field to build on this work.

4.2 Related Work

4.2.1 Legal Reasoning Benchmarks

A large body of prior work has examined the legal reasoning capabilities of language models using static datasets of fixed question-answer pairs. Large scale benchmarks like LegalBench [Guha, Nyarko, Ho, Ré, et al. 2023], LEXTREME [Niklaus, Matoshi, Rani, et al. 2023], LawBench [Fei et al. 2023], and LexGLUE [Chalkidis, Jana, et al. 2022] provide broad assessments across diverse sets of legal tasks. Beyond multi-task benchmarks, other works have targeted specific legal skills, including contract review [Hendrycks et al.

²Code: <https://github.com/servantez/OpenExempt>

³Data: <https://huggingface.co/datasets/SergioServantez/OpenExempt>

2021], legal information retrieval [Joshi et al. 2023; Zheng, Guha, Arifov, et al. 2025], legal exam question answering [Fan et al. 2025], case holding identification [Zheng, Guha, Anderson, et al. 2021], legal judgment prediction [Chalkidis, Androutsopoulos, and Aletras 2019], as well as legal datasets for domain adaptation through pretraining [Henderson et al. 2022; Niklaus, Matoshi, Stürmer, et al. 2024] and instruction tuning [Niklaus, Zheng, et al. 2025]. These benchmarks and datasets have significantly advanced legal reasoning evaluation, yet their static design narrows evaluation to a one-size-fits-all assessment that neither accounts for varying model capabilities nor isolates specific failure modes. OpenExempt introduces a new benchmark paradigm where the user is in control of dynamically crafting legal tasks and defining complexity across multiple dimensions based on their specific evaluation goals.

4.2.2 Computable Statutory Reasoning

Our work is grounded in the field of computational law, where statutes are modeled as executable logic programs. A prominent example is Catala [Huttner and Merigoux 2020; Lawsky 2022], a domain-specific programming language designed to encode real-world tax laws in an executable form using prioritized default logic [Lawsky 2017]. Related work has also demonstrated how natural language contracts can be converted into executable programs in the Accord programming language [Roche et al. 2021], using an intermediate layer of symbolic legal representations [Servantez, Lipka, et al. 2023]. While these works establish the feasibility of symbolic legal encodings, they primarily function as implementation languages or reasoning architectures rather than evaluation benchmarks. SARA [Blair-Stanek, Holzenberger, and Van Durme 2023; Holzenberger, Blair-Stanek, et al. 2020] is the seminal dataset for evaluating statutory reasoning in language models using Prolog encodings to compute gold solutions for tax problems, yet its approach re-

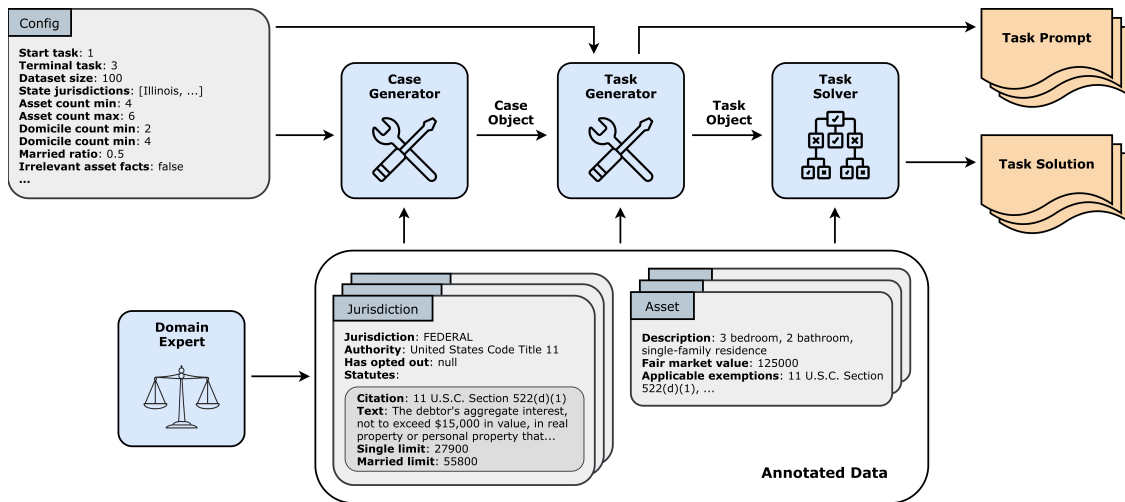


Figure 4.2: OpenExempt Framework Architecture. Dynamic task generation driven by user-defined configuration and grounded in structured legal knowledge.

quires that each scenario be hand-crafted, yielding a fixed dataset of only a few hundred examples. Other prior work has taken an important step toward evaluating hierarchical legal reasoning using case-based analogies, but is also limited by a small, static dataset and does not address statutory reasoning [L. Zhang, Grabmair, et al. 2025]. These threads of work point to the need for a benchmark that is simultaneously dynamic, configurable, diagnostic, grounded in legal knowledge, and scalable beyond hand-crafted datasets – a combination realized in OpenExempt.

4.3 The OpenExempt Framework

The OpenExempt framework is a dynamic task generation system that creates complex legal reasoning benchmarks in the domain of U.S. bankruptcy exemptions. This framework consists of three primary components: 1) a knowledge representation layer that encodes expert legal annotations; 2) a task generator that constructs paired representations in both symbolic and natural language forms; and 3) a deterministic solver that computes ground truth solutions using branch and bound optimization.

The bankruptcy exemption process is an ideal proving ground for controlled evaluation because it allows incremental adjustments to task complexity. Adding assets to a case increases complexity super-linearly: each new asset must not only be evaluated individually, but also in competition with others for shared statutory limits.

4.3.1 Asset Exemption in Bankruptcy

A person filing for bankruptcy, called the Debtor, is allowed to protect certain property from seizure by creditors. An exemption defines a category of property which can be protected - for example, up to \$4,450 in a motor vehicle (Figure 4.1). Each state enacts its own exemption statutes which differ considerably in regards to which assets are protected. The debtor may claim state or federal exemptions, unless their state specifically prohibits the use of federal exemptions, known as "opt-out"⁴. Which state exemption laws apply to a given case is determined by where the debtor lived in the 730 days before filing for bankruptcy⁵. Asset exemption is a combinatorial optimization problem, like a legal version of the well known knapsack problem [Cormen et al. 2022], where an asset can only be protected by certain exemptions and the goal is to minimize the dollar value of unprotected assets. This process involves many intermediate tasks and can be challenging even for legal professionals.

4.3.2 Structured Legal Knowledge

While the dynamic nature of the framework enables controlled variation in task structure and complexity, it also requires that the underlying legal process be represented in a precise and machine-computable form. At the core of OpenExempt are two expert annotated datasets:

⁴11 U.S.C. §522(b)(2)

⁵Id. §522(b)(3)(A)

- **Debtor Assets.** The framework contains over 500 assets (motor vehicles, real estate, household goods) each manually labeled with the complete set of applicable exemptions for every supported jurisdiction. During task generation, the framework samples from this asset collection, providing one of the many factors that enable OpenExempt to construct a vast combinatorial space of possible cases.
- **Exemption Statutes.** We encode federal and state exemption statutes into structured representations that capture the logical rules required for symbolic reasoning, including monetary caps and state opt-out provisions. We also capture a rich set of constraints and relationships that commonly arise in exemption statutes, as discussed in Section 4.3.2.1. OpenExempt currently supports federal exemption statutes and state exemptions for Arizona, Illinois, Oregon, Pennsylvania, and Wisconsin⁶. See Section B.8 for a list of statute sources.

Both datasets adopt a dual representation design, where each asset or statute exists as a pair: a natural language form used in constructing the task prompt, and a structured representation used in computing gold solutions grounded in legal knowledge. This approach is inspired by prior work on smart contracts for legal documents, most notably Accord [Roche et al. 2021] and Catala [Merigoux et al. 2021].

4.3.2.1 Exemption Constraints and Dependencies

To capture the structure and logic of legal exemptions, such as those found in the U.S. Bankruptcy Code, we introduce a formal representation of exemption constraints and dependencies. These refer to the various common conditions and relationships that govern how an exemption may be applied in practice. Exemption constraints include quantitative

⁶These states were selected for diversity in both opt-out status and generosity in exemption coverage and limits.

Table 4.1: Exemption constraints and dependencies represented in OpenExempt, with citations to exemption statutes that exhibit these properties.

Variable Name	Description	Example Exemption
single_limit, married_limit	Maximum aggregated dollar amount that may be claimed by a single debtor or a married couple filing jointly.	11 U.S.C. § 522(d)(1)
per_item_limit	Maximum claimable amount per item, distinct from the overall aggregate limit.	11 U.S.C. § 522(d)(3)
single_item_claim_count, married_item_claim_count	Restricts the use of an exemption to a single item per claim (e.g., one motor vehicle). Married couples filing jointly may each be entitled to a separate single-item claim (e.g., one motor vehicle each).	735 ILCS 5/12-1001(c)
fallback_exemption	Specifies a relationship with another exemption, whose unused aggregate limit may be reallocated to this exemption.	11 U.S.C. § 522(d)(5)
fallback_single_limit, fallback_married_limit	Maximum amount claimable under the fallback exemption, based on marital status.	11 U.S.C. § 522(d)(5)
mutual_exclusion	Defines a mutual exclusion relationship with another exemption, such that claiming either one prohibits the use of the other.	Wis. Stat. § 815.18(3)(b)

or structural limitations, such as caps on the allowable amount per item, differentiated limits for single versus married filers, or restrictions limiting a claim to a single asset. Exemption dependencies, in contrast, encode logical relationships between exemptions, such as mutual exclusions or fallback provisions. Together, these elements form a layer of semantic structure that is critical to accurately modeling exemption behavior and enabling reasoning over exemption applicability and interaction. See Table 4.1 for details on exemption constraints and dependencies.

4.3.3 Dynamic Task Generation

OpenExempt dynamically generates benchmark tasks based on a user-defined configuration file that specifies the structure, scope and complexity of the legal problems being created (see Table B.2 for complete list of parameters). This process is largely driven by two components: CaseGenerator and TaskGenerator (Figure 4.2). At runtime, CaseGenerator constructs symbolic bankruptcy cases by sampling case attributes within the bounds set by the configuration. The resulting case object captures relevant legal facts, including parties, marital status, petition date, domicile timeline, and the applicable exemption jurisdiction based on that timeline. TaskGenerator then renders these structured facts into a natural language prompt. When the user-defined task scope includes intermediate subtask solutions, TaskGenerator invokes the OpenExempt task solver to compute the required intermediate outputs and embeds them into the prompt.

To transform structured case data into natural language narratives, OpenExempt uses a template based approach rather than relying on direct end-to-end generation by a language model. While converting structured data into prose is common in other domains, we argue that the precision required for legal text makes unvalidated generation unsuitable, as models frequently introduce subtle ambiguities or inadvertently alter material facts [Dahl et al. 2024]. To address this, we use language models to generate a diverse set of candidate phrasings for narrative elements (e.g., asset ownership), which are then manually screened by a legal professional and converted into parameterized templates. This hybrid approach preserves linguistic variety while ensuring that the resulting fact patterns remain precise and aligned with the ground truth.

4.3.4 Computing Gold Solutions

While OpenExempt tasks are dynamically generated, all solutions are grounded in expert knowledge. The annotated assets enumerate the exemption claims permissible for each asset, while the machine-readable statutes encode the constraints that govern how those claims may be applied. Together, these resources enable the solver to validate candidate outputs and thus define the solution space. For asset-level tasks, like Task EC and EV (defined in Section 4.4.1), the ground truth is directly recovered from the asset annotations for the relevant jurisdiction. For estate-level tasks, like Task NA and OE, which require jointly allocating exemptions across all assets, the framework employs the symbolic solver to perform a branch and bound search over all legally valid exemption assignments. This brute force search is made tractable by pruning partial solutions that cannot surpass the best known allocation, based on remaining exemption capacity and unprocessed assets. Because the solver only explores legally valid allocations, and because all legal rules defining valid claims originate from expert curated encodings, the resulting optimal allocation is both computationally verified and expert grounded.

4.3.4.1 Objective Correctness

Legal reasoning benchmarks must navigate the inherent gray area of statutory interpretation. OpenExempt mitigates this challenge by tightly controlling the scope of legal content and assets included in the benchmark, enabling the construction of tasks where solutions are objectively correct to a high degree of confidence. This requires the deliberate exclusion or modification of statutory provisions that introduce subjectivity. The goal of OpenExempt is not to perfectly model the application of the Bankruptcy Code and state exemption laws, but rather to construct complex legal reasoning tasks with objectively correct answers, which closely resemble real-world legal problems. We prioritize objec-

tive correctness through the following design choices:

- **Controlled Asset and Statute Selection.** We curate the pool of assets and exemptions to exclude provisions that rely on subjective standards, such as those requiring an item to be "reasonably necessary". By focusing primarily on tangible assets with clear statutory definitions and avoiding exemptions that depend on complex debtor attributes (disability status, profession), we ensure that the applicability of an exemption is a binary and deterministic question. The description of each asset contains all necessary predicates for a model to determine its eligibility.
- **Normalized Statutory Text for Self-Contained Reasoning.** The exemption statutes in OpenExempt are normalized to eliminate external references and latent ambiguity. For example, an exemption can incorporate requirements defined outside the current title: "Uniforms and accoutrements as provided by 51 Pa.C.S. § 4103"⁷. In these situations, we omit the reference or inline relevant text if possible. This ensures the model is evaluated on its ability to reason over the task prompt, rather than its ability to recall external legal knowledge not present in that context.
- **Encodable Exemption Logic.** We restrict the benchmark to exemption provisions whose operative logic can be faithfully captured by our formal constraint and dependency representation (Section 4.3.2.1). While this representation covers many common statutory patterns, not all exemptions can be reduced to these encodings given the infinite variability in natural language. By excluding provisions that cannot be encoded, we ensure that our derived ground truth solutions remain computationally verifiable.

⁷42 Pa. Cons. Stat. §8124(a)(4)

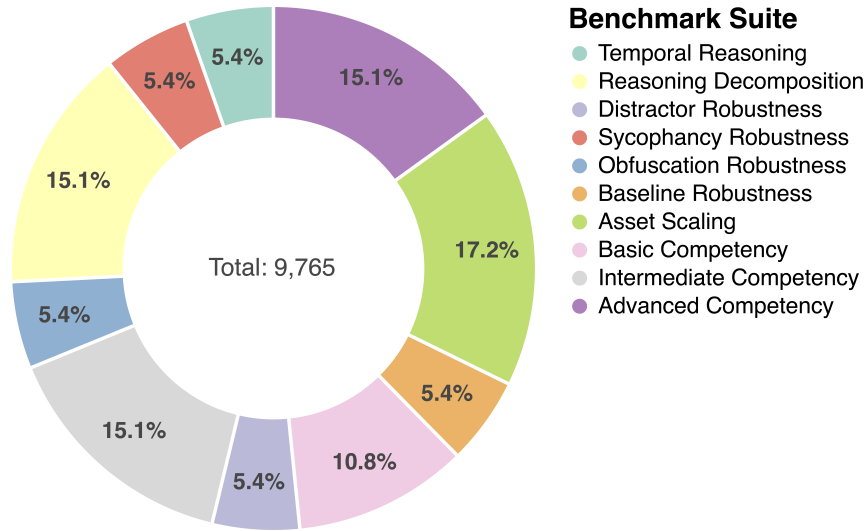


Figure 4.3: Sample distribution across benchmark suites.

4.4 The OpenExempt Benchmark

Using the above framework, we construct the OpenExempt benchmark consisting of 9,765 samples across nine evaluation suites (Figure 4.3 shows the sample distribution across suites).

4.4.1 Tasks

OpenExempt is composed of five tasks, with a total of 15 task variants, that mirror the sequence of legal reasoning steps a debtor’s attorney performs when protecting assets in bankruptcy. For each task, the model receives a fact pattern detailing the debtor’s situation, which may include asset disclosures and residential history, along with a corpus of relevant federal and state laws. We show an example Task EC prompt in Figure 4.5, provide additional prompt examples with solutions in Section B.7, and describe each task below:

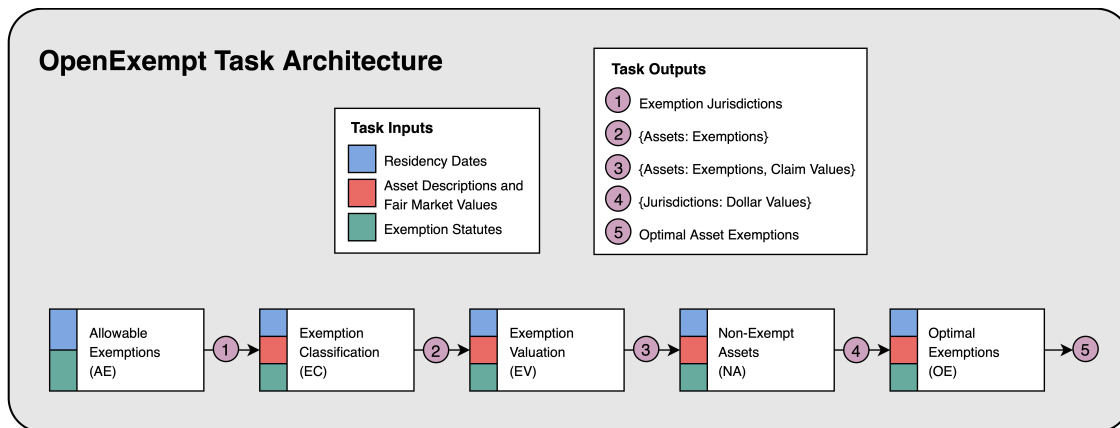


Figure 4.4: OpenExempt Task Pipeline. Each task depends on the successful completion of its predecessors, forming a composable sequence in which users can select any slice to isolate evaluation of specific reasoning types.

- **Task AE (Allowable Exemptions):** Before exemptions can be claimed, the Bankruptcy Code requires first understanding which state or federal exemptions are available to the Debtor. This task involves applying the multi-step "730-day Rule"⁸ to the Debtor's residency history to identify the applicable exemption jurisdictions, while accounting for state opt-out provisions⁹.
- **Task EC (Exemption Classification):** Once the allowed exemption jurisdictions have been identified, each asset must be matched to the categories of exempt property defined by statute. This task requires rule-based reasoning to determine if a given asset satisfies the exemption antecedent, the specific property category defined by the statute. This is a multi-label classification problem since multiple exemptions can apply to a single asset.
- **Task EV (Exemption Valuation):** Exemptions are typically limited to a fixed dollar amount defined by the statute. This task requires not only identifying applicable exemptions, but also applying these statutory caps to calculate the maximum pro-

⁸11 U.S.C. §522(b)(3)(A)

⁹Id. §522(b)(2)

tectable dollar value for each asset under each of its available exemptions. Tasks EC and EV require asset-level reasoning since each asset-exemption pair is considered independently, without factoring in aggregate limits.

- **Task NA (Non-exempt Assets):** Tasks NA and OE demand estate-level reasoning to strategically allocate exemptions across all the Debtor’s assets. This task requires solving this strategic allocation to determine the minimal total dollar value of non-exempt assets after applying all applicable exemptions, for each allowable exemption jurisdiction.
- **Task OE (Optimal Exemptions):** This task requires articulating the complete, optimal strategy to achieve the best outcome from Task NA. This requires selecting the allowable exemption jurisdiction that minimizes non-exempt asset value, and generating the explicit exemption schedule for that optimal jurisdiction. The schedule must produce a complete mapping of which exemptions, and what dollar amounts, are allocated to each specific asset.

Task Variants. Since OpenExempt tasks form a sequential pipeline (Figure 4.4), solving any given task depends on the successful completion of its predecessors. The OpenExempt framework allows users to configure which earlier steps are already solved and provided in the task prompt. This creates a family of task variants, where a task can be presented in its vanilla form (no prior steps solved) or with some or all preceding steps solved by the framework. Each variant corresponds to a contiguous interval of the pipeline. This design enables a fine-grained assessment of how cumulative complexity and error propagation impact model performance, a capability we later demonstrate with the Reasoning Decomposition evaluation suite.

Task EC Example: Distractor Robustness Suite

For each asset in the estate, identify all applicable exemptions under which that asset may be protected.

Your answer to this task must be based solely on applying the provided Federal and State statutes to the given facts. If the task involves a married couple, assume all assets mentioned are jointly owned, with each spouse holding an equal undivided interest, unless explicitly stated otherwise. Assume all assets are held for the personal use of the Debtor(s), unless explicitly stated otherwise.

Facts: Megha and Dalia Joshi (hereinafter the Debtors) filed for bankruptcy on March 7th 2024. The Debtors began living in Litchfield Park, Arizona on 19 March 2012, but relocated to Waunakee, Wisconsin on 9.10.2021. For the 28 days following that date, Megha stayed in Patagonia, Arizona to complete a boater safety course and obtain a state-issued boating certificate, a new requirement for their job as a marine biologist. While there, they signed a rental agreement (Contract #R-781) for a boat slip at a local marina to berth the assigned training vessel for the duration of the course. The Joshis chose to relocate their household to Weyauwega, Wisconsin on 12th of September, 2021. Records show that Megha and Dalia Joshi possess a compact Bluetooth speaker with splash-resistant casing with a value of \$400.00. Megha and Dalia own a calico cat worth \$145.00. The Debtor's name is listed on a UTMA savings account, and the corresponding Form 1099-INT is mailed to their address. The account holds a balance of \$8,150, which originated as an irrevocable gift from the Debtor's brother to the account's beneficiary, the Debtor's 14 year-old nephew. The known assets of Megha and Dalia Joshi include a woven tapestry wall hanging with bohemian motif worth \$1,305.00 and a Hi-Point C9 9mm pistol appraised at \$195.00.

Figure 4.5: Example Task EC (Exemption Classification) prompt from the Distractor Robustness evaluation suite; response format instructions and statutes abridged for brevity.

4.4.2 Benchmark Suites

The OpenExempt Benchmark organizes its evaluation into nine suites designed to capture both broad and fine-grained assessments of legal reasoning. The three competency suites evaluate a wide range of exemption scenarios to provide a holistic view of a model's reasoning capabilities, mirroring a traditional benchmark. In contrast, the six diagnostic suites isolate and vary one specific dimension of task complexity, such as the density of obfuscating statements. This approach enables targeted, causal analysis, allowing us to

isolate and understand precisely which specific reasoning challenges are contributing to performance degradation. See Table B.3 for a summary of configuration settings for each suite.

4.4.2.1 Competency Suites

Basic, Intermediate, and Advanced Competency. These three suites form a structured progression of difficulty where higher tiers contain more complex fact patterns, including larger asset pools, more extensive domicile histories, and exemption statutes drawn from a broader set of state jurisdictions. This tiered design ensures that the benchmark remains informative across a wide range of model capabilities: smaller models can be meaningfully evaluated on the lower tiers without collapsing to failure, while larger models can be challenged at higher tiers without saturating performance. In this way, the competency suites yield reliable, discriminative signals that align with the reasoning capacity of the model being assessed.

4.4.2.2 Diagnostic Suites

Temporal Reasoning. This suite isolates reasoning about temporal rules that govern exemption eligibility under the Bankruptcy Code. In these tasks, the Debtor’s prior residences are spread across multiple states and dates, and the model must determine which exemptions the Debtor is permitted to claim by correctly applying the 730-day Rule under 11 U.S.C. §522(b)(3)(A). This rule is a three part statutory test that requires reasoning about both the duration and location of the Debtor’s domicile. By increasing the complexity of the domicile history while holding all other settings constant, we can precisely measure how temporal complexity affects model performance.

Reasoning Decomposition. This diagnostic suite measures the effects of cumulative complexity and error propagation across the OpenExempt task pipeline (Figure 4.4). It evaluates Tasks EC, EV, NA and OE, by testing against all possible preceding task variants (Task AE has no preceding tasks). This configuration allows for a causal decomposition of total error into two components: stage error, which reflects the model’s inability to solve the target task itself (e.g., Task EV), and propagation error, which arises from the model’s reliance on its own incorrect conclusions from preceding steps (e.g., Tasks AE and EC).

Distractor, Sycophancy, and Obfuscation Robustness. The OpenExempt benchmark includes three diagnostic suites designed to evaluate how well models maintain legal accuracy when faced with extraneous, misleading, or irrelevant information within the fact pattern. OpenExempt supports two forms of obfuscation: irrelevant facts, which introduce legally immaterial details about assets or prior residences (e.g., property not owned by the Debtor or travel that does not alter domicile), and opinions, which present subjective statements about assets or exemption eligibility that carry no legal force. The Distractor Robustness suite introduces irrelevant facts, testing whether models can ignore seemingly pertinent but legally inconsequential information. The Sycophancy Robustness suite introduces opinion statements, measuring whether models are influenced by subjective assertions rather than statutory requirements. The Obfuscation Robustness suite combines both types, presenting the full range of distracting and misleading content. The effect of each perturbation is isolated by comparison to a baseline configuration with no obfuscating statements. Figure 4.5 shows an example task prompt with embedded distractor statements.

Asset Scaling. Asset pool size is a significant driver of task complexity. This suite evaluates how model performance changes as the number of assets in the Debtor’s es-

tate is incrementally scaled. The primary source of emerging difficulty is the strategic competition between assets for limited statutory exemption values. In cases with one or two assets, the optimal allocation of exemptions can be trivial. However, as the asset pool size grows and multiple assets become eligible for the same finite exemption values, the task transitions into a complex optimization problem. The model is forced to consider alternative exemptions and strategically allocate the available dollar value of each to maximize the total protected value of the estate, thereby demanding a much stricter and more comprehensive legal reasoning process.

4.5 Results and Discussion

We summarize our findings here, and provide complete experiment details in the [Appendix](#).

4.5.1 Experimental Setup

To support few-shot learning, we follow LEXam [Fan et al. 2025] and split 5 samples from each task dataset into a dev set, with the remaining 100 samples in the test set. Evaluation suites contain a collection of these 105-sample datasets, each with its own configuration file. Across all suites, this yields 9,765 samples in total, split into 9,300 test samples and 465 dev samples. Prior work has shown that language models can struggle with in-context demonstrations in the legal domain [Servantez, Barrow, et al. 2024]. Therefore, we focus this work on evaluating models in a zero-shot setting to establish baseline performance, but leave exploration of few-shot learning for future work.

Table 4.2: Model performance (sample-based $F1$) by task across Basic (bc), Intermediate (ic), and Advanced Competency (ac) suites.

Task	AE	EC	EV	NA	OE
Suite	bc / ic / ac	bc / ic / ac	bc / ic / ac	bc / ic / ac	bc / ic / ac
GPT-5	.884 /.743 /.612	.924 /.744 /.554	.893 /.635 /.496	.893 /.733 /.558	.933 /.744 /.404
o3	.917 /.747 /.604	.901 /.743 /.571	.912 /.651 /.500	.949 /.728 /.548	.944 /.724 /.408
o4-mini	.940 /.757 /.621	.711 /.575 /.418	.691 /.541 /.388	.744 /.539 /.326	.844 /.543 /.265
Claude-Sonnet-4	.940 /.743 /.625	.723 /.502 /.452	.697 /.478 /.353	.718 /.499 /.340	.805 /.476 /.255
Gemini-2.5-Pro	.943 /.753 /.605	.900 /.740 /.549	.877 /.623 /.502	.889 /.665 /.540	.938 /.714 /.518
DeepSeek-R1	.957 /.728 /.610	.809 /.612 /.457	.771 /.546 /.364	.860 /.649 /.476	.901 /.607 /.347
GPT-4.1	.955 /.714 /.588	.522 /.276 /.224	.453 /.207 /.163	.689 /.504 /.316	.777 /.538 /.240
Llama-4-Maverick	.865 /.659 /.586	.515 /.317 /.226	.496 /.320 /.193	.554 /.227 /.122	.703 /.260 /.095
DeepSeek-V3	.942 /.673 /.627	.598 /.365 /.291	.530 /.366 /.274	.594 /.393 /.196	.802 /.330 /.170
Claude-3.5-Haiku	.707 /.572 /.360	.529 /.411 /.317	.415 /.344 /.256	.502 /.204 /.069	.561 /.147 /.026
Gemma-3	.710 /.539 /.403	.503 /.447 /.331	.373 /.264 /.191	.404 /.224 /.137	.660 /.140 /.007
Gemini-2.5-Flash	.935 /.723 /.574	.835 /.671 /.474	.836 /.562 /.396	.870 /.586 /.441	.935 /.671 /.355
Llama-4-Scout	.596 /.480 /.386	.401 /.360 /.281	.350 /.294 /.172	.422 /.188 /.118	.569 /.136 /.033

4.5.2 Models

We evaluate 13 language models grouped into three categories: 1) *reasoning models*: GPT-5 [OpenAI 2025a], Claude-Sonnet-4 [Anthropic 2025], Gemini-2.5-Pro [Google 2025b], o3 [OpenAI 2025c], o4-mini [OpenAI 2025c], and Deepseek-R1 [DeepSeek-AI, Guo, et al. 2025]; 2) *large models*: GPT-4.1 [OpenAI 2025b], Llama-4-Maverick (17B-128E-Instruct) [Meta 2025], and Deepseek-V3 [DeepSeek-AI, Liu, et al. 2025]; 3) *efficient models*: Gemini-2.5-Flash [Google 2025a], Claude-3.5-Haiku [Anthropic 2024], Llama-4-Scout (17B-16E-Instruct) [Meta 2025], and Gemma-3-(27b-it) [Team et al. 2025]. We use a temperature of 0 for all models that support temperature, except for DeepSeek-R1 which we set to 0.6 based on developer recommended settings [DeepSeek 2025]. We set max token length to 16384, or the maximum token length supported by the model if it is less. We find these extended outputs are necessary to ensure complete answers.

4.5.3 Evaluation Protocol

Across all tasks, OpenExempt reports precision, recall and F1 scores computed at the sample level and then macro-averaged across samples. For asset-level tasks (EC, EV), the evaluator first computes per asset scores within a case, then averages across assets to determine the sample score, preventing assets with more applicable exemptions from dominating the aggregate. For tasks with multi-label predictions (AE, EC, EV), we evaluate using set overlap across discrete labels (jurisdictions or exemption citations). For tasks involving dollar valued predictions (EV, NA, OE), we additionally compute mean absolute relative error (MARE) between predicted and gold amounts. A numeric prediction is treated as correct if it falls within a 5% absolute relative error tolerance of the corresponding gold value. Because relative error can become unstable for gold amounts near zero, we add a small stabilizing constant ϵ . Formally, for a predicted amount $\hat{y}$ and gold amount y , we define the within-tolerance indicator:

$$\mathbb{I}_\tau(\hat{y}, y) = \mathbf{1} \left[\left| \frac{\hat{y}}{y + \epsilon} - 1 \right| < \tau \right]_{\epsilon=1, \tau=0.05}.$$

For tasks that require structured outputs (EC, EV, NA, OE), predictions that fail to parse are marked as *invalid format* and are scored as incorrect, making format compliance a measured component of the benchmark. We observe a low rate of malformed responses across all evaluated models, typically below 1% (Table B.1). We discuss LM response validation in detail in Section B.3.

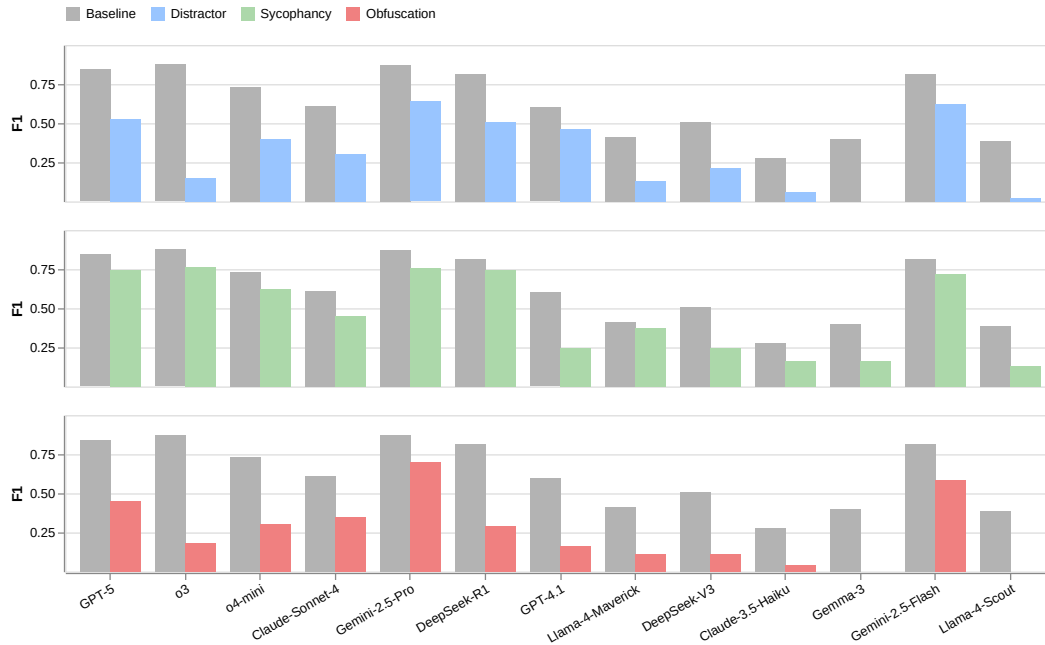


Figure 4.6: Model performance (F1) on Task OE under Distractor, Sycophancy, and Obfuscation perturbations. Colored bars show performance under each robustness suite, highlighting the degree to which model accuracy degrades relative to baseline.

4.5.4 Competency Evaluation

Basic, Intermediate, and Advanced Competency. **Competency suites produce stable, monotonic performance degradation across increasing difficulty levels, yielding clear and reliable distinctions between model reasoning capacities** (Table 4.2). Reasoning LMs predominantly outperform efficient and large non-reasoning models, with performance gaps particularly pronounced in more difficult settings. Gemini-2.5-Flash is a clear exception to this trend: despite being an efficient model, it consistently performs closer to reasoning models across our evaluations. GPT-5 and Gemini-2.5-Pro often rank among the best performing models. Yet, model recency alone does not explain performance, as o3 frequently outperforms newer models, even in the advanced tier. Notably, no model achieves an F1 score above 0.63 on any task in the advanced suite, indicating room for improvement in advanced legal reasoning where multi-step inference and complex rule

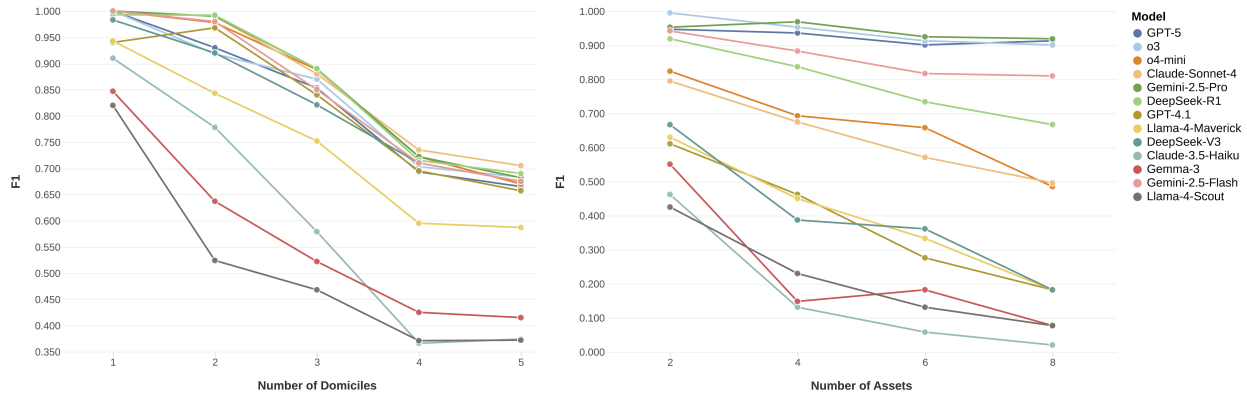


Figure 4.7: Model performance (F1) on Temporal Reasoning (left) and Asset Scaling (right) suites.

application are prevalent. The Basic Competency suite reveals clear distinctions between efficient models on Task OE that vanish under advanced settings. For example, Gemma-3 outperforms Llama-4-Scout in the basic tier ($F1 = 0.66$ vs. 0.569), but both models collapse to near-zero performance in the advanced tier. Conversely, reasoning models approach saturation on simpler tasks ($F1 > 0.9$ on AE-bc), masking capabilities that only diverge under higher complexity settings. Our tiered evaluation approach mitigates these floor and ceiling effects that would otherwise obscure distinctions between models. To disentangle the failures observed in the competency evaluations, we next turn to the diagnostic suites.

4.5.5 Diagnostic Evaluation

Temporal Reasoning. **Temporal reasoning fails at a predictable threshold for reasoning models: performance declines modestly, then drops sharply, before leveling off** (Figure 4.7, Table B.4). As temporal complexity increases while holding all other settings constant, reasoning models achieve perfect or near perfect performance with one domicile, and degrade only slightly at two (typically less than 0.03 $F1$). Performance drops become noticeably sharper as complexity reaches three and four domiciles, with four marking

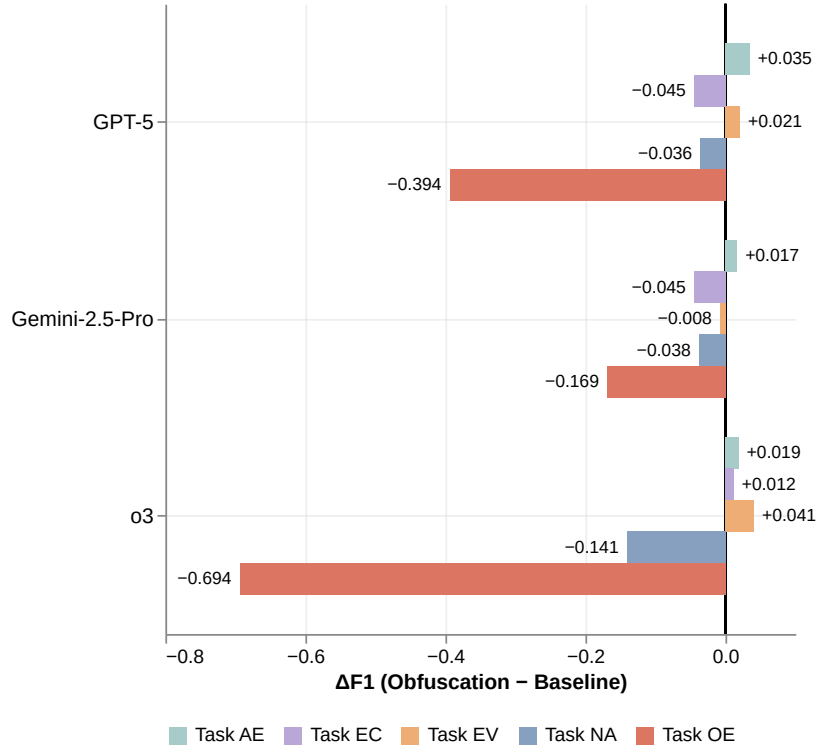


Figure 4.8: Obfuscation Robustness by task for three top performing reasoning models. Each bar shows the absolute change in F1 (ΔF_1) under obfuscation perturbations, computed as obfuscation minus baseline. Obfuscating statements are identical across tasks. Positive values indicate performance gains, negative values indicate degradation.

the clearest breaking point (at least a 0.145 F_1 decrease for all reasoning models). This trend does not continue at five domiciles, where performance decreases typically return to only a few F_1 points. We observe a similar pattern across all reasoning models (Figure 4.7). In contrast, efficient models exhibit greater variability, and tend to degrade earlier and more smoothly as domicile complexity increases.

Asset Scaling. Asset scaling sharply separates model capacity: a select few frontier reasoning models remain robust under multi-asset exemption optimization, while many other models collapse (Figure 4.7, Tables B.11 to B.13). Performance declines are comparatively modest on asset-level tasks (EC, EV), even for efficient models, but become

markedly sharper on estate-level tasks (NA, OE). For example, when scaling assets from 2 to 8, the $F1$ score for Llama-4-Scout decreases by 0.057 (EC) and 0.053 (EV), compared to 0.177 (NA) and 0.348 (OE). This gap reflects the shift from local exemption decisions to globally constrained allocation across competing assets. However, this degradation pattern is not universal, as three reasoning models (GPT-5, o3, Gemini-2.5-Pro) prove significantly more robust to increases in asset count, even where optimization demands are most acute. For example, GPT-5 declines by only 0.034 $F1$ on Task OE under full asset scaling (2 to 8 assets). These results indicate that the performance drops observed in the Advanced Competency suite cannot be attributed to asset complexity alone, but rather to the interaction of multiple difficulty dimensions.

Distractor, Sycophancy, and Obfuscation Robustness. **Model vulnerability to obfuscation is not uniform: identical statements can produce disparate levels of harm, which compound as legal reasoning becomes more complex** (Figures 4.6 and 4.8, Tables B.8 to B.10). Model performance typically declines in the presence of obfuscating statements (distractors, opinions), but several models exhibit slight performance increases on simpler tasks (AE, EC), particularly under distractors alone. This pattern suggests extraneous information can encourage more deliberate analysis in less complex settings. Across Tasks AE through NA, the strongest reasoning models (GPT-5, o3, Gemini-2.5-Pro) exhibit little to no degradation under any obfuscation setting (Figure 4.8). Yet for Task OE, all reasoning models show substantial declines, with distractor and obfuscation perturbations producing the sharpest drops (Figure 4.6 and 4.8). This contrast is revealing given that all tasks use identical obfuscation statements. Models demonstrate a clear ability to discount irrelevant facts and opinions in simpler tasks, but stumble when the same statements appear in a more complex setting.

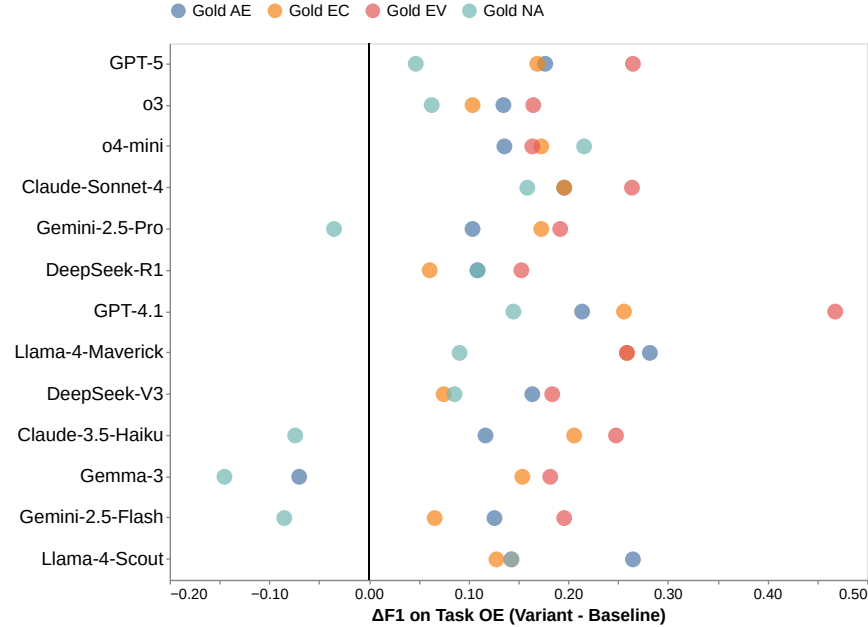


Figure 4.9: Reasoning Decomposition on Task OE. Each point shows the absolute change in F1 (ΔF_1) when a model is provided with gold solutions to a specific intermediate subtask, computed as variant minus baseline (no solved steps). Positive values indicate performance gains, negative values indicate degradation.

Reasoning Decomposition. **Correct intermediate solutions do not guarantee downstream performance gains and can even degrade it, revealing that reasoning through intermediate steps can be more beneficial than conditioning on partial solutions** (Figure 4.9, Tables B.5 to B.7). Providing gold intermediate solutions typically boosts downstream performance, indicating that error propagation is a dominant factor in multi-step reasoning failures. Yet exceptions to this trend suggest that partial solutions can disrupt the model’s reasoning trajectory for estate-level tasks that involve long reasoning paths. For example, efficient models often perform worse on Task OE when provided with gold NA solutions, despite NA supplying the exact non-exempt dollar amount that OE seeks to minimize. Notably, four of six reasoning models perform worse on Task NA when provided with gold EC or EV solutions, despite these steps specifying the exemptions and valuation limits needed to compute the total non-exempt amount. This behavior is par-

ticularly surprising for two reasons: (1) it is most pronounced in the reasoning models that otherwise exhibit the strongest reasoning capabilities across our experiments (GPT-5, o3, Gemini-2.5-Pro), and (2) it is largely absent in efficient and large non-reasoning models. We hypothesize that reasoning oriented post-training reinforces end-to-end reasoning trajectories, rather than conditional reasoning from partially solved states. As a result, providing intermediate conclusions can sometimes reduce performance by disrupting these reasoning trajectories.

4.6 Limitations

We note several limitations rooted in our design choices. OpenExempt currently: (i) focuses on bankruptcy and state exemption law; (ii) evaluates only U.S. federal law and a small number of selected state jurisdictions; (iii) does not support multilingual tasks; and (iv) focuses on objectively correct tasks, which does not reflect the ambiguity common in legal practice. Given these limitations, OpenExempt should be treated as a complement to current evaluation methods, not a replacement. OpenExempt was designed to be easily extended by either legal or technical skill sets. We believe there is significant potential to build on OpenExempt and view these limitations as natural starting points for future work, including developing and evaluating new approaches to instruction tuning for step-wise legal reasoning.

4.7 Chapter Summary

We introduce OpenExempt, a framework for dynamically generating complex legal tasks grounded in structured legal knowledge, and a diagnostic benchmark for evaluating legal reasoning capabilities in language models. We release OpenExempt to the public to

support further research and encourage collaboration between the legal and NLP communities.

Chapter 5

A Unified Perspective on Language Models as Legal Reasoners

5.1 The Unified Perspective

This dissertation has shown that legal reasoning with language models is advanced by symbolic structure grounded in legal practice. Each chapter of this dissertation is an instantiation of that principle, applied at a different point in the legal reasoning pipeline. Chapter 2 imposed structure at the level of legal language itself, extracting the semantic skeleton of contractual obligations into a symbolic graph. Chapter 3 imposed structure at the level of reasoning procedure, guiding models through the decomposition and recombination of compositional rules. Chapter 4 imposed structure at the level of evaluation, generating controlled tasks that expose exactly where reasoning fails and why. Taken together, these chapters reflect a broader view that language models can be made both more capable and more legible as legal reasoners across three dimensions: extraction, generation, and evaluation, with each deployed according to the needs of the task.

5.2 Core Insights

Considering the dissertation as a whole yields several key insights about language models as legal reasoners:

Legal reasoning capability in language models is real but fragile. Across all experiments, models demonstrate a clear capacity to reason over legal text, including recovering semantic structure, performing rule application, and solving nontrivial legal tasks. However, that capability can degrade rapidly as reasoning demands grow. It collapses under conditions that legal practice considers routine: longer reasoning chains, more complex dependencies, or the introduction of irrelevant information. That brittleness suggests that what models possess is not durable legal reasoning competence, but a limited capacity that must be carefully elicited and supported. The broader insight is that legal reasoning in language models is neither absent nor robust.

Legal structure elicits legal reasoning in language models. The dissertation shows this in multiple forms. Legal text becomes more tractable when its operative components are represented explicitly. Rule application improves when the reasoning process is decomposed into intermediate rule elements rather than relying on the model to implicitly track complex logical states. Representing contractual obligations as typed entities and directed relations rather than unstructured text allows downstream systems to verify, execute, and audit those obligations. Evaluation becomes more informative when task complexity is controlled rather than collapsed into a static benchmark score. In each case, making legal structure explicit does not merely make model behavior easier to interpret, it creates the conditions under which stronger legal reasoning becomes possible.

Robust legal reasoning requires more than correct conclusions. A correct answer does not necessarily indicate a sound reasoning process, and an aggregate benchmark score can obscure the specific demands that caused failure. The dissertation repeatedly shows the importance of examining intermediate structure, whether in extracted legal representations, decomposed reasoning steps, or controlled benchmark variations. This makes it possible to distinguish different kinds of failure that would otherwise be conflated: failures of interpretation, failures of rule application, or failures induced by error propagation over long reasoning trajectories. For the legal domain, where justification and traceability matter alongside outcome, this distinction is paramount.

These observations support a broader conclusion about language models as legal reasoners. Their limitations do not arise solely because law is complex, but because legal tasks require a form of disciplined reasoning that probabilistic next-token prediction does not reliably produce on its own. At the same time, the dissertation shows that these limitations are not fixed. When legal tasks are structured in ways that reflect the dependencies, procedures, and constraints that govern legal analysis, language models become more dependable and more transparent. The central conclusion is therefore not that language models cannot reason about law, but that effective computational legal reasoning emerges most clearly when statistical models are guided by structures that make legal reasoning explicit.

5.3 Future Directions

These insights point to a clear direction for future research. The first is the advancement of training and inference methods that better support stepwise legal reasoning. The results of this dissertation suggest that models have latent capacity for rule-based reasoning

that can be elicited through structured guidance, yet they remain vulnerable to long reasoning paths, error propagation, and complex interactions between intermediate steps. Future work should therefore explore whether similar demonstrations used in instruction tuning, or decomposed task formulations used in reinforcement learning, can produce more reliable structured reasoning. The OpenExempt framework is well positioned to support this effort. Its ability to generate large quantities of dynamic legal reasoning tasks with verified gold solutions creates a scalable training environment for stepwise legal reasoning, something that has historically been unavailable due to the cost of expert annotation. A pressing research question is whether models post-trained on similar data become more robust on held-out legal reasoning tasks, and whether those gains generalize across legal domains.

Legal evaluation must also evolve alongside the capabilities of language models. As models improve, the field will require increasingly complex legal tasks to meaningfully differentiate performance and expose remaining weaknesses. Yet this creates a growing tension for traditional benchmark design. If legal evaluation continues to rely primarily on direct expert annotation of static question-answer pairs, then the time, cost, and expertise required to build sufficiently challenging datasets will rise accordingly. At the same time, static benchmarks remain limited in what they reveal. Even when they contain difficult problems, they often reduce performance to a binary signal of success or failure, offering little insight into which reasoning demands caused the error or how failure emerged over the course of a multi-step legal task. Future work should therefore move toward evaluation methods that are not only more scalable, but also better able to probe legal reasoning more deeply by varying task complexity in controlled ways, isolating distinct reasoning demands, and examining intermediate reasoning paths rather than only final outputs. While OpenExempt provides an initial blueprint for using structured le-

gal knowledge to generate scalable, diagnostic evaluations, it remains bound to a single area of law, and its construction required substantial legal interpretation, formalization, and engineering effort. Extending this kind of methodology more broadly will therefore require new ways of reducing the legal and engineering overhead involved in creating structured, computable benchmarks. The challenge ahead is not only to build dynamic evaluations for one area of law, but to develop generalizable methods for constructing richer and more diagnostic benchmarks across the entire legal domain.

5.4 Concluding Remarks

The stakes of this work extend well beyond benchmark scores. Language models capable of robust legal reasoning have the potential to transform the legal industry by reducing human error, improving the efficiency of legal analysis, and increasing the capacity of legal professionals to navigate growing volumes of complex legal text. Advancing computational legal reasoning is not only a technical challenge, it also bears directly on the accessibility of legal services, and therefore on the global crisis in equal access to justice.

This dissertation has demonstrated a path toward more reliable computational legal reasoning by respecting its underlying logical structure. By translating legal knowledge and procedures into machine-computable representations, structured reasoning methods, and diagnostic evaluation frameworks, this dissertation narrows the gap between probabilistic text generation and strict rule application. Ultimately, advancing language models as legal reasoners is not about replacing human judgment, but about building systems whose logic can be interpreted, whose failures can be diagnosed, and whose conclusions can be verified.

References

- Vinay Aggarwal, Aparna Garimella, Balaji Vasan Srinivasan, Anandhavelu N, and Rajiv Jain. Nov. 2021. “ClauseRec: A Clause Recommendation Framework for AI-aided Contract Authoring.” In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, (Nov. 2021), 8770–8776. DOI: [10.18653/v1/2021.emnlp-main.691](https://doi.org/10.18653/v1/2021.emnlp-main.691).
- Norah Alzahrani et al.. 2024. *When Benchmarks are Targets: Revealing the Sensitivity of Large Language Model Leaderboards*. (2024). <https://arxiv.org/abs/2402.01781> arXiv: 2402.01781 [cs.CL].
- Anthropic. May 2025. *Claude Sonnet 4: System Card*. Accessed: 2025-12-25. (May 2025). <https://www-cdn.anthropic.com/6be99a52cb68eb70eb9572b4cafad13df32ed995.pdf>.
- Anthropic. 2024. *The Claude 3 Model Family: Opus, Sonnet, Haiku*. Accessed: 2025-12-25. (2024). <https://assets.anthropic.com/m/61e7d27f8c8f5919/original/Claude-3-Model-Card.pdf>.
- Michael Bommarito II au2 and Daniel Martin Katz. 2022. *GPT Takes the Bar Exam*. (2022). arXiv: 2212.14402 [cs.CL].
- Andrew Blair-Stanek, Nils Holzenberger, and Benjamin Van Durme. 2023. *BLT: Can Large Language Models Handle Basic Legal Text?* (2023). arXiv: 2311.09693 [cs.CL].
- Andrew Blair-Stanek, Nils Holzenberger, and Benjamin Van Durme. 2023. “Can GPT-3 Perform Statutory Reasoning?” In: *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law (ICAAIL ’23)*. Association for Computing Machinery, Braga, Portugal, 22–31. ISBN: 9798400701979. DOI: [10.1145/3594536.3595163](https://doi.org/10.1145/3594536.3595163).

- Tom B. Brown et al.. 2020. *Language Models are Few-Shot Learners*. (2020). arXiv: [2005.14165 \[cs.CL\]](#).
- Ilias Chalkidis, Ion Androutsopoulos, and Nikolaos Aletras. July 2019. “Neural Legal Judgment Prediction in English.” In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Korhonen, David Traum, and Lluís Màrquez. Association for Computational Linguistics, Florence, Italy, (July 2019), 4317–4323. DOI: [10.18653/v1/P19-1424](#).
- Ilias Chalkidis, Ion Androutsopoulos, and Achilleas Michos. 2017. “Extracting Contract Elements.” In: *Proceedings of the 16th Edition of the International Conference on Artificial Intelligence and Law (ICAAIL ’17)*. Association for Computing Machinery, London, United Kingdom, 19–28. ISBN: 9781450348911. DOI: [10.1145/3086512.3086515](#).
- Ilias Chalkidis, Ion Androutsopoulos, and Achilleas Michos. July 2018. “Obligation and Prohibition Extraction Using Hierarchical RNNs.” In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Melbourne, Australia, (July 2018), 254–259. DOI: [10.18653/v1/P18-2041](#).
- Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. *LEGAL-BERT: The Muppets straight out of Law School*. (2020). DOI: [10.48550/ARXIV.2010.02559](#).
- Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Martin Katz, and Nikolaos Aletras. 2022. *LexGLUE: A Benchmark Dataset for Legal Language Understanding in English*. (2022). <https://arxiv.org/abs/2110.00976> arXiv: [2110.00976 \[cs.CL\]](#).
- Mark Chen et al.. 2021. *Evaluating Large Language Models Trained on Code*. (2021). DOI: [10.48550/ARXIV.2107.03374](#).
- Odysseas S. Chlapanis, Dimitrios Galanis, Nikolaos Aletras, and Ion Androutsopoulos. 2025. *GreekBarBench: A Challenging Benchmark for Free-Text Legal Reasoning and Citations*. (2025). <https://arxiv.org/abs/2505.17267> arXiv: [2505.17267 \[cs.CL\]](#).
- Christopher D Clack. 2021. “Languages for smart and computable contracts.” *arXiv preprint arXiv:2104.03764*.
- Christopher D. Clack, Vikram A. Bakshi, and Lee Braine. 2017. *Smart Contract Templates: foundations, design landscape and research directions*. (2017). <https://arxiv.org/abs/1608.00771> arXiv: [1608.00771 \[cs.CY\]](#).

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2022. *Introduction to Algorithms*. (4th ed.). MIT Press, Cambridge, MA.
- Silvia Crafa, Cosimo Laneve, and Giovanni Sartor. 2022. “Stipula: a domain specific language for legal contracts.” In: *Int. Workshop Programming Languages and the Law*.
- Matthew Dahl, Varun Magesh, Mirac Suzgun, and Daniel E Ho. Jan. 2024. “Large Legal Fictions: Profiling Legal Hallucinations in Large Language Models.” *Journal of Legal Analysis*, 16, 1, (Jan. 2024), 64–93. DOI: [10.1093/jla/laae003](https://doi.org/10.1093/jla/laae003).
- DeepSeek. 2025. *DeepSeek R1 Model Card*. Accessed: 2025-12-27. (2025). <https://huggingface.co/deepseek-ai/DeepSeek-R1>.
- DeepSeek-AI, Daya Guo, et al.. 2025. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. (2025). <https://arxiv.org/abs/2501.12948> arXiv: 2501.12948 [cs.CL].
- DeepSeek-AI, Aixin Liu, et al.. 2025. *DeepSeek-V3 Technical Report*. (2025). <https://arxiv.org/abs/2412.19437> arXiv: 2412.19437 [cs.CL].
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. “Bert: Pre-training of deep bidirectional transformers for language understanding.” *arXiv preprint arXiv:1810.04805*.
- Xiao Ding, Zhongyang Li, Ting Liu, and Kuo Liao. 2019. *ELG: An Event Logic Graph*. (2019). DOI: [10.48550/ARXIV.1907.08015](https://doi.org/10.48550/ARXIV.1907.08015).
- Markus Eberts and Adrian Ulges. 2019. “Span-based Joint Entity and Relation Extraction with Transformer Pre-training.” *ArXiv*, abs/1909.07755.
- Ed. by K. J. Holyoak and R. G. Morrison Jr. “Legal Reasoning.” *The Cambridge Handbook of Thinking and Reasoning*. Cambridge University Press, 685–704.
- Yu Fan et al.. 2025. *LEXam: Benchmarking Legal Reasoning on 340 Law Exams*. (2025). <https://arxiv.org/abs/2505.12864> arXiv: 2505.12864 [cs.CL].
- Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, Fengzhe Zhou, Zhuo Han, Songyang Zhang, Kai Chen, Zongwen Shen, and Jidong Ge. 2023. *LawBench: Benchmarking Legal Knowledge of Large Language Models*. (2023). <https://arxiv.org/abs/2309.16289> arXiv: 2309.16289 [cs.CL].

- Ruka Funaki, Yusuke Nagata, Kohei Suenaga, and Shinsuke Mori. May 2020. “A Contract Corpus for Recognizing Rights and Obligations.” English. In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. European Language Resources Association, Marseille, France, (May 2020), 2045–2053. ISBN: 979-10-95546-34-4. <https://aclanthology.org/2020.lrec-1.251>.
- Ingo Glaser, Elena Scepankova, and Florian Matthes. 2018. “Classifying semantic types of legal sentences: Portability of machine learning models.” In: *Legal Knowledge and Information Systems*. IOS Press, 61–70.
- Google. Dec. 2025a. *Gemini 2.5 Flash Model Card*. Accessed: 2025-12-25. (Dec. 2025). <https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-2-5-Flash-Model-Card.pdf>.
- Google. June 2025b. *Gemini 2.5 Pro Model Card*. Accessed: 2025-12-25. (June 2025). <https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-2-5-Pro-Model-Card.pdf>.
- Guido Governatori and Meng Weng Wong. 2023. “Defeasible Semantics for L4.” *Workshop on Programming Languages and the Law*.
- Alex Graves. 2017. *Adaptive Computation Time for Recurrent Neural Networks*. (2017). arXiv: [1603.08983](https://arxiv.org/abs/1603.08983) [cs.NE].
- James Grimmelmann, Benjamin Sobel, and David Stein. 2025. “Generative Misinterpretation.” *Harvard Journal on Legislation*, 63.1.
- Ralph Grishman and Beth M Sundheim. 1996. “Message understanding conference-6: A brief history.” In: *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.
- Neel Guha, Julian Nyarko, Daniel E. Ho, and Christopher Ré. 2025. *Building GenAI Benchmarks: A Case Study in Legal Applications*. (2025). eprint: https://academic.oup.com/book/0/chapter/523978823/chapter-ag-pdf/66125429/book_59908_section_523978823.ag.pdf. DOI: 10.1093/oxfordhb/9780198940272.013.0007.
- Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, et al.. 2023. *LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models*. (2023). <https://arxiv.org/abs/2308.11462> arXiv: 2308.11462 [cs.CL].
- Peter Henderson, Mark S. Krass, Lucia Zheng, Neel Guha, Christopher D. Manning, Dan Jurafsky, and Daniel E. Ho. 2022. *Pile of Law: Learning Responsible Data Filtering from*

- the Law and a 256GB Open-Source Legal Dataset*. (2022). <https://arxiv.org/abs/2207.00220> arXiv: 2207.00220 [cs.CL].
- Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. *CUAD: An Expert-Annotated NLP Dataset for Legal Contract Review*. (2021). <https://arxiv.org/abs/2103.06268> arXiv: 2103.06268 [cs.CL].
- Valentin Hofmann et al.. 2025. *Fluid Language Model Benchmarking*. (2025). <https://arxiv.org/abs/2509.11106> arXiv: 2509.11106 [cs.CL].
- Nils Holzenberger, Andrew Blair-Stanek, and Benjamin Van Durme. 2020. *A Dataset for Statutory Reasoning in Tax Law Entailment and Question Answering*. (2020). <https://arxiv.org/abs/2005.05257> arXiv: 2005.05257 [cs.CL].
- Nils Holzenberger and Benjamin Van Durme. Aug. 2021. “Factoring Statutory Reasoning as Language Understanding Challenges.” In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli. Association for Computational Linguistics, Online, (Aug. 2021), 2742–2758. DOI: [10.18653/v1/2021.acl-long.213](https://doi.org/10.18653/v1/2021.acl-long.213).
- Yiran Hu, Huanghai Liu, Qingjing Chen, Ning Zheng, Chong Wang, Yun Liu, Charles L. A. Clarke, and Weixing Shen. 2025. *J&H: Evaluating the Robustness of Large Language Models Under Knowledge-Injection Attacks in Legal Domain*. (2025). <https://arxiv.org/abs/2503.18360> arXiv: 2503.18360 [cs.CL].
- Liane Huttner and Denis Merigoux. 2020. “Catala: Moving towards the future of legal expert systems.” *Artificial Intelligence and Law*. <https://api.semanticscholar.org/CorpusID:225231457>.
- Albert Q. Jiang et al.. 2023. *Mistral 7B*. (2023). arXiv: 2310.06825 [cs.CL].
- Cong Jiang and Xiaolei Yang. 2023. *Legal Syllogism Prompting: Teaching Large Language Models for Legal Judgment Prediction*. (2023). arXiv: 2307.08321 [cs.CL].
- Zhanming Jie, Jierui Li, and Wei Lu. May 2022. “Learning to Reason Deductively: Math Word Problem Solving as Complex Relation Extraction.” In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, (May 2022), 5944–5955. DOI: [10.18653/v1/2022.acl-long.410](https://doi.org/10.18653/v1/2022.acl-long.410).

- Abhinav Joshi, Akshat Sharma, Sai Kiran Tanikella, and Ashutosh Modi. 2023. *U-CREAT: Unsupervised Case Retrieval using Events extrAcTion*. (2023). <https://arxiv.org/abs/2307.05260> arXiv: 2307.05260 [cs.IR].
- Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2021. *Text Modular Networks: Learning to Decompose Tasks in the Language of Existing Models*. (2021). arXiv: 2009.00751 [cs.CL].
- Douwe Kiela et al.. 2021. *Dynabench: Rethinking Benchmarking in NLP*. (2021). <https://arxiv.org/abs/2104.14337> arXiv: 2104.14337 [cs.CL].
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. *Large Language Models are Zero-Shot Reasoners*. (2023). arXiv: 2205.11916 [cs.CL].
- Sarah Lawsky. Dec. 2017. “A Logic for Statutes.” *Florida Tax Review*, 21, (Dec. 2017), 60–80. DOI: [10.5744/ftr.2017.0002](https://doi.org/10.5744/ftr.2017.0002).
- Sarah Lawsky. Jan. 2022. “Coding the Code: Catala and Computationally Accessible Tax Law.” *SMU Law Review*, 75, (Jan. 2022), 535. DOI: [10.25172/smulr.75.3.4](https://doi.org/10.25172/smulr.75.3.4).
- Sha Li, Heng Ji, and Jiawei Han. 2021. “Document-level event argument extraction by conditional generation.” *arXiv preprint arXiv:2104.05919*.
- Chin-Yew Lin. July 2004. “ROUGE: A Package for Automatic Evaluation of Summaries.” In: *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, (July 2004), 74–81. <https://aclanthology.org/W04-1013/>.
- Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. July 2020. “A Joint Neural Model for Information Extraction with Global Features.” In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, (July 2020), 7999–8009. DOI: [10.18653/v1/2020.acl-main.713](https://doi.org/10.18653/v1/2020.acl-main.713).
- Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. *Program Induction by Rationale Generation : Learning to Solve and Explain Algebraic Word Problems*. (2017). arXiv: 1705.04146 [cs.AI].
- Ed. by Robert J. Sternberg and Scott Barry Editors Kaufman. “Intelligence and Reasoning.” *The Cambridge Handbook of Intelligence*. Cambridge Handbooks in Psychology. Cambridge University Press, 419–441.

- Varun Magesh, Faiz Surani, Matthew Dahl, Mirac Suzgun, Christopher D. Manning, and Daniel E. Ho. 2024. *Hallucination-Free? Assessing the Reliability of Leading AI Legal Research Tools*. (2024). <https://arxiv.org/abs/2405.20362> arXiv: 2405.20362 [cs.CL].
- Laura Manor and Junyi Jessy Li. June 2019. “Plain English Summarization of Contracts.” In: *Proceedings of the Natural Legal Language Processing Workshop 2019*. Ed. by Nikolaos Aletras, Elliott Ash, Leslie Barrett, Daniel Chen, Adam Meyers, Daniel Preotiuc-Pietro, David Rosenberg, and Amanda Stent. Association for Computational Linguistics, Minneapolis, Minnesota, (June 2019), 1–11. DOI: [10.18653/v1/W19-2201](https://doi.org/10.18653/v1/W19-2201).
- Puneet Mathur, Vlad Morariu, Verena Kaynig-Fittkau, Jiuxiang Gu, Franck Dernoncourt, Quan Hung Tran, Ani Nenkova, Dinesh Manocha, and Rajiv Jain. 2022. “DocTime: A Document-level Temporal Dependency Graph Parser.” In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 993–1009.
- Denis Merigoux, Nicolas Chataing, and Jonathan Protzenko. Aug. 2021. “Catala: a programming language for the law.” *Proc. ACM Program. Lang.*, 5, ICFP, Article 77, (Aug. 2021), 29 pages. DOI: [10.1145/3473582](https://doi.org/10.1145/3473582).
- Meta. 2025. *Llama 4 Model Cards and Prompt formats*. Accessed: 2025-12-25. (2025). <https://www.llama.com/docs/model-cards-and-prompt-formats/llama4/>.
- Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. *Multi-hop Reading Comprehension through Question Decomposition and Rescoring*. (2019). arXiv: [1906.02916](https://arxiv.org/abs/1906.02916) [cs.CL].
- Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2025. *GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models*. (2025). <https://arxiv.org/abs/2410.05229> arXiv: 2410.05229 [cs.LG].
- Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. 2022. *Reframing Instructional Prompts to GPTk’s Language*. (2022). arXiv: [2109.07830](https://arxiv.org/abs/2109.07830) [cs.CL].
- Venkatesh Mishra, Bimsara Pathiraja, Mihir Parmar, Sat Chidananda, Jayanth Srinivasa, Gaowen Liu, Ali Payani, and Chitta Baral. 2025. *Investigating the Shortcomings of LLMs in Step-by-Step Legal Reasoning*. (2025). <https://arxiv.org/abs/2502.05675> arXiv: 2502.05675 [cs.CL].

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. “Orca: Progressive learning from complex explanation traces of gpt-4.” *arXiv preprint arXiv:2306.02707*.

Joel Niklaus, Veton Matoshi, Pooja Rani, Andrea Galassi, Matthias Stürmer, and Ilias Chalkidis. 2023. “LEXTREME: A Multi-Lingual and Multi-Task Benchmark for the Legal Domain.” In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, 3016–3054. DOI: [10.18653/v1/2023.findings-emnlp.200](https://doi.org/10.18653/v1/2023.findings-emnlp.200).

Joel Niklaus, Veton Matoshi, Matthias Stürmer, Ilias Chalkidis, and Daniel E. Ho. 2024. *MultiLegalPile: A 689GB Multilingual Legal Corpus*. (2024). <https://arxiv.org/abs/2306.02069> arXiv: 2306.02069 [cs.CL].

Joel Niklaus, Lucia Zheng, et al.. 2025. *LawInstruct: A Resource for Studying Language Model Adaptation to the Legal Domain*. (2025). <https://arxiv.org/abs/2404.02127> arXiv: 2404.02127 [cs.CL].

Maxwell Nye et al.. 2021. *Show Your Work: Scratchpads for Intermediate Computation with Language Models*. (2021). arXiv: [2112.00114](https://arxiv.org/abs/2112.00114) [cs.LG].

OpenAI. 2023. “Gpt-4 technical report.” *View in Article*, 2, 13.

OpenAI. Aug. 2025a. *GPT-5 System Card*. Accessed: 2025-12-25. (Aug. 2025). <https://cdn.openai.com/gpt-5-system-card.pdf>.

OpenAI. Apr. 2025b. *Introducing GPT-4.1 in the API*. Accessed: 2025-12-25. (Apr. 2025). <https://openai.com/index/gpt-4-1/>.

OpenAI. Apr. 2025c. *OpenAI o3 and o4-mini System Card*. Accessed: 2025-12-25. (Apr. 2025). <https://openai.com/index/o3-o4-mini-system-card/>.

Monica Palmirani, Guido Governatori, Antonino Rotolo, Said Tabet, Harold Boley, and Adrian Paschke. 2011. “LegalRuleML: XML-Based Rules and Norms.” *RuleML America*, 7018, 298–312.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. *Measuring and Narrowing the Compositionality Gap in Language Models*. (2023). arXiv: [2210.03350](https://arxiv.org/abs/2210.03350) [cs.CL].

Pydantic. 2025. *Pydantic Validation*. Accessed: 2025-12-28. (2025). <https://docs.pydantic.dev>.

- Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D. Manning. 2019. *Answering Complex Open-domain Questions Through Iterative Query Generation*. (2019). arXiv: [1910.07000](https://arxiv.org/abs/1910.07000) [cs.CL].
- RapidFuzz. 2025. *RapidFuzz Documentation*. Accessed: 2025-12-28. (2025). <https://rapidfuzz.github.io/RapidFuzz/>.
- Mitchel Resnick et al.. 2009. “Scratch: programming for all.” *Communications of the ACM*, 52, 11, 60–67.
- Niall Roche, Walter Hernandez, Eason Chen, Jérôme Siméon, and Dan Selman. 2021. “Ergo—a programming language for Smart Legal Contracts.” *arXiv preprint arXiv:2112.07064*.
- Martha T Roth. 1995. *Law collections from Mesopotamia and Asia minor*. Vol. 6. Scholars Press.
- Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. *Interpretation of Natural Language Rules in Conversational Machine Reading*. (2018). arXiv: [1809.01494](https://arxiv.org/abs/1809.01494) [cs.CL].
- Abhilasha Sancheti, Aparna Garimella, Balaji Vasanth Srinivasan, and Rachel Rudinger. 2022. *Agent-Specific Deontic Modality Detection in Legal Language*. (2022). DOI: [10.48550/ARXIV.2211.12752](https://doi.org/10.48550/ARXIV.2211.12752).
- Sergio Servantez, Joe Barrow, Kristian Hammond, and Rajiv Jain. 2024. *Chain of Logic: Rule-Based Reasoning with Large Language Models*. (2024). <https://arxiv.org/abs/2402.10400> arXiv: [2402.10400](https://arxiv.org/abs/2402.10400) [cs.CL].
- Sergio Servantez, Nedim Lipka, Alexa Siu, Milan Aggarwal, Balaji Krishnamurthy, Aparna Garimella, Kristian Hammond, and Rajiv Jain. 2023. “Computable Contracts by Extracting Obligation Logic Graphs.” In: *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law (ICAAIL ’23)*. Association for Computing Machinery, Braga, Portugal, 267–276. ISBN: 9798400701979. DOI: [10.1145/3594536.3595162](https://doi.org/10.1145/3594536.3595162).
- Yuzhen Shi et al.. 2026. *PLawBench: A Rubric-Based Benchmark for Evaluating LLMs in Real-World Legal Practice*. (2026). <https://arxiv.org/abs/2601.16669> arXiv: [2601.16669](https://arxiv.org/abs/2601.16669) [cs.CL].
- Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025. *The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity*. (2025). <https://arxiv.org/abs/2506.06941> arXiv: [2506.06941](https://arxiv.org/abs/2506.06941) [cs.AI].

Harry Surden. 2012. “Computable Contracts.” *UC Davis Law Review*, 46.

Alon Talmor and Jonathan Berant. 2018. *The Web as a Knowledge-base for Answering Complex Questions*. (2018). arXiv: 1803.06643 [cs.CL].

Gemma Team et al.. 2025. *Gemma 3 Technical Report*. (2025). <https://arxiv.org/abs/2503.19786> arXiv: 2503.19786 [cs.CL].

Hugo Touvron et al.. 2023. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. (2023). arXiv: 2307.09288 [cs.CL].

Arianna Trozze, Toby Davies, and Bennett Kleinberg. 2024. *Large Language Models in Cryptocurrency Securities Cases: Can a GPT Model Meaningfully Assist Lawyers?* (2024). <https://arxiv.org/abs/2308.06032> arXiv: 2308.06032 [cs.AI].

Don Tuggener, Pius von Däniken, Thomas Peetz, and Mark Cieliebak. May 2020. “LEDGAR: A Large-Scale Multi-label Corpus for Text Classification of Legal Provisions in Contracts.” English. In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. European Language Resources Association, Marseille, France, (May 2020), 1235–1241. ISBN: 979-10-95546-34-4. <https://aclanthology.org/2020.lrec-1.155>.

Minh Van Nguyen, Viet Dac Lai, and Thien Huu Nguyen. 2021. *Cross-Task Instance Representation Interactions and Label Dependencies for Joint Information Extraction with Graph Convolutional Networks*. (2021). DOI: 10.48550/ARXIV.2103.09330.

Ngoc Phuoc An Vo, Irene Manotas, Octavian Popescu, Algimantas Černiauskas, and Vadim Sheinin. Sept. 2021. “Recognizing and Splitting Conditional Sentences for Automation of Business Processes Management.” In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*. INCOMA Ltd., Held Online, (Sept. 2021), 1490–1497. <https://aclanthology.org/2021.ranlp-1.167>.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. Nov. 2019. “Entity, Relation, and Event Extraction with Contextualized Span Representations.” 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)*. Association for Computational Linguistics, Hong Kong, China, (Nov. 2019), 5784–5789. DOI: 10.18653/v1/D19-1585.

Chenguang Wang, Xiao Liu, Zui Chen, Haoyun Hong, Jie Tang, and Dawn Song. 2022. “DeepStruct: Pretraining of language models for structure prediction.” *arXiv preprint arXiv:2205.10475*.

- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. (2023). arXiv: 2203.11171 [cs.CL].
- Yubo Wang et al.. 2024. *MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark*. (2024). <https://arxiv.org/abs/2406.01574> arXiv: 2406.01574 [cs.CL].
- Zihan Wang, Hongye Song, Zhaochun Ren, Pengjie Ren, Zhumin Chen, Xiaozhong Liu, Hongsong Li, and Maarten de Rijke. July 2021. “Cross-Domain Contract Element Extraction with a Bi-directional Feedback Clause-Element Relation Network.” In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, (July 2021). DOI: 10.1145/3404835.3462873.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. (2023). arXiv: 2201.11903 [cs.CL].
- Reinhardt Wenzina and Katharina Kaiser. 2013. “Identifying Condition-Action Sentences Using a Heuristic-Based Information Extraction Method.” In: *KR4HC/ProHealth*.
- Jason Weston and Sainbayar Sukhbaatar. 2023. *System 2 Attention (is something you might need too)*. (2023). arXiv: 2311.11829 [cs.CL].
- Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. *Break It Down: A Question Understanding Benchmark*. (2020). arXiv: 2001.11770 [cs.CL].
- Yuxin Xiao, Zecheng Zhang, Yuning Mao, Carl Yang, and Jiawei Han. 2021. *SAIS: Supervising and Augmenting Intermediate Steps for Document-Level Relation Extraction*. (2021). DOI: 10.48550/ARXIV.2109.12093.
- Yuan Yao et al.. 2019. “DocRED: A large-scale document-level relation extraction dataset.” *arXiv preprint arXiv:1906.06127*.
- Fangyi Yu, Lee Quartey, and Frank Schilder. 2022. *Legal Prompting: Teaching a Language Model to Think Like a Lawyer*. (2022). arXiv: 2212.01326 [cs.CL].
- Kepu Zhang, Weijie Yu, Zhongxiang Sun, and Jun Xu. 2025. *An Explicit Syllogistic Legal Reasoning Framework for Large Language Models*. (2025). <https://arxiv.org/abs/2504.04042> arXiv: 2504.04042 [cs.CL].

- Li Zhang, Matthias Grabmair, Morgan Gray, and Kevin Ashley. 2025. *Thinking Longer, Not Always Smarter: Evaluating LLM Capabilities in Hierarchical Legal Reasoning*. (2025). <https://arxiv.org/abs/2510.08710> arXiv: 2510.08710 [cs.CL].
- “Do LLMs Truly “Understand” when a Precedent Is Overruled?” *Legal Knowledge and Information Systems*. IOS Press, (Dec. 2025). ISBN: 9781643686387. DOI: [10.3233/faia251592](https://doi.org/10.3233/faia251592).
- Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. 2021. *When Does Pretraining Help? Assessing Self-Supervised Learning for Law and the CaseHOLD Dataset*. (2021). <https://arxiv.org/abs/2104.08671> arXiv: 2104.08671 [cs.CL].
- Lucia Zheng, Neel Guha, Javokhir Arifov, Sarah Zhang, Michal Skreta, Christopher D. Manning, Peter Henderson, and Daniel E. Ho. Mar. 2025. “A Reasoning-Focused Legal Retrieval Benchmark.” In: *Proceedings of the Symposium on Computer Science and Law on ZZZ* (CSLAW ’25). ACM, (Mar. 2025), 169–193. DOI: [10.1145/3709025.3712219](https://doi.org/10.1145/3709025.3712219).
- Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. 2019. *JEC-QA: A Legal-Domain Question Answering Dataset*. (2019). arXiv: [1911.12011](https://arxiv.org/abs/1911.12011) [cs.CL].
- Denny Zhou et al.. 2023. *Least-to-Most Prompting Enables Complex Reasoning in Large Language Models*. (2023). arXiv: [2205.10625](https://arxiv.org/abs/2205.10625) [cs.AI].

Appendix A

Chain of Logic Appendix

A.1 LegalBench Task Examples

Personal Jurisdiction (Figure 3.3)

- Rule: There is personal jurisdiction over a defendant in the state where the defendant is domiciled, or when (1) the defendant has sufficient contacts with the state, such that they have availed itself of the privileges of the state and (2) the claim arises out of the nexus of the defendant's contacts with the state.
- Fact Pattern: Dustin is a repairman who lives in Arizona and repairs computers in California, Oregon, and Washington. While travelling to repair a computer in Washington, Dustin is involved in a car crash in Oregon with Laura, a citizen of Texas. After the accident, Dustin returns to Arizona. Laura sues him in Oregon.
- Issue: Is there personal jurisdiction?

Diversity Jurisdiction 1

- Rule: Diversity jurisdiction exists when there is (1) complete diversity between plain-

tiffs and defendants, and (2) the amount-in-controversy (AiC) is greater than \$75k.

- Fact Pattern: James is from Arizona. Lucas is from Arizona. James sues Lucas for negligence for \$64,000.
- Issue: Is there diversity jurisdiction?

Diversity Jurisdiction 2

- Rule: Diversity jurisdiction exists when there is (1) complete diversity between plaintiffs and defendants, and (2) the amount-in-controversy (AiC) is greater than \$75k.
- Fact Pattern: Sophia is from Arkansas. Benjamin is from Hawaii. Noah is from Arkansas. Sophia sues Benjamin and Noah each for defamation for \$24,000.
- Issue: Is there diversity jurisdiction?

Diversity Jurisdiction 3

- Rule: Diversity jurisdiction exists when there is (1) complete diversity between plaintiffs and defendants, and (2) the amount-in-controversy (AiC) is greater than \$75k.
- Fact Pattern: William is from Montana. Theodore is from Connecticut. William sues Theodore for medical malpractice for \$9,000 and negligence for \$35,000.
- Issue: Is there diversity jurisdiction?

Diversity Jurisdiction 4

- Rule: Diversity jurisdiction exists when there is (1) complete diversity between plaintiffs and defendants, and (2) the amount-in-controversy (AiC) is greater than \$75k.

- Fact Pattern: Emma is from New Hampshire. Mia is from Wisconsin. Evelyn is from California. Emma and Mia both sue Evelyn for copyright infringement for \$5,400,000.
- Issue: Is there diversity jurisdiction?

Diversity Jurisdiction 5

- Rule: Diversity jurisdiction exists when there is (1) complete diversity between plaintiffs and defendants, and (2) the amount-in-controversy (AiC) is greater than \$75k.
- Fact Pattern: Elijah is from Hawaii. Ava is from Oklahoma. Amelia is from Minnesota. Elijah and Ava both sue Amelia for defamation for \$3,000 and copyright infringement for \$80,000.
- Issue: Is there diversity jurisdiction?

Diversity Jurisdiction 6

- Rule: Diversity jurisdiction exists when there is (1) complete diversity between plaintiffs and defendants, and (2) the amount-in-controversy (AiC) is greater than \$75k.
- Fact Pattern: Theodore is from North Dakota. Amelia is from Georgia. Benjamin is from Delaware. Mia is from Illinois. Theodore and Amelia both sue Benjamin for trademark infringement for \$42,000 and copyright infringement for \$71,000. Theodore and Amelia both sue Mia for securities fraud for \$45,000 and medical malpractice for \$57,000.
- Issue: Is there diversity jurisdiction?

J.Crew Blocker

- Rule: The JCrew Blocker is a provision that typically includes (1) a prohibition on the borrower from transferring IP to an unrestricted subsidiary, and (2) a requirement that the borrower obtains the consent of its agent/lenders before transferring IP to any subsidiary.
- Fact Pattern: Notwithstanding anything to foregoing, no Intellectual Property that is material to the Borrower and its Restricted Subsidiaries, taken as a whole (as reasonably determined by the Borrower), shall be owned by or licensed, contributed or otherwise transferred to any Unrestricted Subsidiary.
- Issue: Do the following provisions contain JCrew Blockers?

A.2 Model Performance by Rule

Model performance (accuracy) by rule for all reasoning tasks. Diversity Jurisdiction results are aggregated across all six datasets.

	Personal Jurisdiction	Diversity Jurisdiction	J.Crew Blocker
Zero-Shot	68.0	73.9	87.0
Zero-Shot-LR	60.0	69.2	42.6
Zero-Shot-LS	70.0	69.5	87.0
Standard Prompting	72.0	73.5	72.2
Chain of Thought	74.0	69.7	66.7
Self-Ask	60.0	70.7	74.0
Chain of Logic (ours)	78.0	88.6	94.4

Table A.1: GPT-3.5 Performance (Accuracy) by LegalBench Rule

	Personal Jurisdiction	Diversity Jurisdiction	J.Crew Blocker
Zero-Shot	84.0	87.1	100
Zero-Shot-LR	86.0	87.8	98.1
Zero-Shot-LS	78.0	94.2	98.1
Standard Prompting	76.0	93.6	96.3
Chain of Thought	82.0	89.2	96.3
Self-Ask	84.0	78.5	96.3
Chain of Logic (ours)	90.0	94.3	92.6

Table A.2: GPT-4 Performance (Accuracy) by LegalBench Rule

	Personal Jurisdiction	Diversity Jurisdiction	J.Crew Blocker
Zero-Shot	66.0	65.9	90.7
Zero-Shot-LR	52.0	66.0	77.8
Zero-Shot-LS	42.0	50.0	83.3
Standard Prompting	56.0	57.7	81.5
Chain of Thought	62.0	60.7	92.6
Self-Ask	70.0	52.8	79.6
Chain of Logic (ours)	72.0	66.6	85.2

Table A.3: Llama-2-70b Performance (Accuracy) by LegalBench Rule

	Personal Jurisdiction	Diversity Jurisdiction	J.Crew Blocker
Zero-Shot	56.0	65.4	61.1
Zero-Shot-LR	50.0	66.1	72.2
Zero-Shot-LS	48.0	62.4	46.3
Standard Prompting	44.0	46.3	46.3
Chain of Thought	36.0	66.2	37.0
Self-Ask	52.0	70.8	16.7
Chain of Logic (ours)	54.0	72.4	63.0

Table A.4: MistralOrca-7b Performance (Accuracy) by LegalBench Rule

Appendix B

OpenExempt Appendix

B.1 Modular Task Prompts

We manually write instructions for each task, which are combined with the generated fact patterns and selected exemption statutes to form the task prompts. When the user specifies a task variant (Section 4.4.1), we also include solved intermediate reasoning steps. All task prompt components (instructions, fact patterns, statutes) are stored separately in the benchmark, allowing us to collapse repeated elements across prompts to substantially reduce storage size. This modular design also aligns with our diagnostic evaluation goals by allowing the community to explore changes to question format and phrasing, which prior work has shown can have unpredictable effects on model performance [Alzahrani et al. 2024; Y. Wang et al. 2024]. OpenExempt provides a `TaskDataset` class to handle loading and iterating over examples.

B.2 Response Format Compliance

Table B.1: Frequency and Percentage of Model Responses with Malformed JSON

Model	Frequency	Percent
GPT-5	2	0.03
o3	4	0.05
o4-mini	34	0.44
Claude-Sonnet-4	1	0.01
Gemini-2.5-Pro	3	0.04
DeepSeek-R1	34	0.44
GPT-4.1	12	0.15
Llama-4-Maverick	173	2.22
DeepSeek-V3	3	0.04
Claude-3.5-Haiku	22	0.28
Gemma-3	198	2.54
Gemini-2.5-Flash	153	1.96
Llama-4-Scout	117	1.50
Total	756	0.75

B.3 Response Validation

OpenExempt provides an Evaluator class to handle task-specific evaluation logic, including response format compliance and validation of predicted claims. For each sample, the evaluator: (i) isolates the final solution by extracting the suffix after the “FINAL AN-

SWER:” marker; (ii) parses the response with a task-specific Pydantic parser [Pydantic 2025]; and (iii) normalizes exemption citations (case folding, trimming) and aligns asset descriptions using fuzzy string matching (RapidFuzz [RapidFuzz 2025] partial ratio with a threshold of 90) to ensure stable mapping between model predictions and gold labels. For all tasks except OE, evaluation compares predictions directly against the provided gold targets. Since optimal exemption schedules may not be unique, we evaluate Task OE by first validating predicted claims (e.g., ensuring claims obey exemption caps), before comparing against the known optimal outcome. This validation process is grounded in the same symbolic case objects and machine-readable statutes used during task generation. The predicted solution need not match the gold target to be correct, as long as its legally valid and achieves the same degree of protection.

B.4 Configuration Parameters

Table B.2: OpenExempt configuration parameters. Each parameter is specified within the configuration file to control task scope and complexity, dataset size, and degree of obfuscation.

Parameter(s)	Description
<code>start_task_id, terminal_task_id</code>	The process of exempting assets under the Bankruptcy Code proceeds through a fixed sequence of intermediate tasks (see Figure 4.4). These configuration parameters specify which portion of that sequence the model is responsible for solving. <code>start_task_id</code> marks the first task to be solved, and <code>terminal_task_id</code> marks the last. When both are set to the same value (e.g., 3-3), the configuration isolates a single reasoning task; when set to the broadest range (e.g., 1-5), it evaluates the entire exemption process.
<code>dataset_size</code>	Specifies the number of unique tasks, and their corresponding ground-truth solutions, to generate under the given configuration. Each task is independently sampled using the specified asset ranges, jurisdictions, obfuscation settings, and all other configuration parameters.
<code>asset_count_min, asset_count_max</code>	Defines the minimum and maximum number of assets to include in each generated task. The actual asset count is sampled uniformly across this range, ensuring an equal distribution of tasks at each asset count. This allows controlled variation in task complexity across the dataset.
<code>married_ratio</code>	Specifies the proportion of generated tasks that involve married debtors. Marital status affects applicable exemption limits in many jurisdictions.
<code>domicile_count_min,</code> <code>domicile_count_max</code>	The minimum and maximum number of prior domiciles to include in each fact pattern, sampled uniformly across the specified range. Domicile history determines which federal and state exemption laws a Debtor is eligible to claim.
<code>state_jurisdictions</code>	Specifies the set of U.S. state jurisdictions used for task generation. For each task, one jurisdiction is sampled uniformly from this list to serve as the Debtor’s allowable exemption jurisdiction. The exemption statutes for all listed jurisdictions are included in the prompt, requiring the model to identify the correct jurisdiction and apply its exemption laws to the facts.
<code>irrelevant_asset_facts,</code> <code>irrelevant_domicile_facts,</code> <code>asset_opinions, domicile_opinions</code>	Boolean parameters that control the inclusion of obfuscating information in the fact pattern. When enabled, the benchmark injects legally immaterial details or subjective statements related to assets or domicile history. These parameters are used to evaluate the model’s robustness to distraction, misdirection, and sycophancy by testing its ability to disregard extraneous details while applying the correct legal reasoning.
<code>data_directory,</code> <code>asset_directory,</code> <code>statute_directory, template_directory,</code> <code>output_directory</code>	File path parameters that specify where the framework loads input resources and saves generated outputs. The input directories point to data dependencies required for task generation (annotated assets, exemption statutes, natural-language templates), while <code>output_directory</code> designates where generated tasks and solutions are written.

B.5 Benchmark Suite Composition

Table B.3: Summary of configuration settings for each evaluation suite. Each dataset in the benchmark contains a configuration file with the exact construction specification.

Suite	Tasks	Solved Steps	Asset Count	Married Ratio	DomicileObfuscation Count	States
Temporal Reasoning	AE	No	N/A	0.5	1-5	No All
Reasoning Decomposition	EC-OE	Yes	6	1.0	4	No WI, IL, OR
Distractor Robustness	All	No	4	0.5	3	Yes AZ, PA, WI
Sycophancy Robustness	All	No	4	0.5	3	Yes AZ, PA, WI
Obfuscation Robustness	All	No	4	0.5	3	Yes AZ, PA, WI
Asset Scaling	EC-OE	No	2-8	0.0	2	No IL, OR, PA
Basic Competency	All	No	2	0.0	2-3	No WI, IL
Intermediate Competency	All	No	3-5	0.5	4	Yes AZ, PA, OR
Advanced Competency	All	No	6-8	1.0	5	Yes All

B.6 Diagnostic Suite Results

Table B.4: Model Performance ($F1$) on Temporal Reasoning Suite.

Number of Domiciles	One	Two	Three	Four	Five
GPT-5	1.00	.930	.853	.693	.665
o3	1.00	.918	.870	.703	.683
o4-mini	1.00	.978	.888	.722	.670
Claude-Sonnet-4	1.00	.990	.880	.735	.705
Gemini-2.5-Pro	1.00	.990	.890	.722	.682
DeepSeek-R1	.993	.992	.890	.715	.690
GPT-4.1	.940	.968	.840	.695	.657
Llama-4-Maverick	.943	.843	.752	.595	.587
DeepSeek-V3	.983	.920	.821	.710	.675
Claude-3.5-Haiku	.910	.778	.579	.366	.374
Gemma-3	.847	.637	.522	.425	.415
Gemini-2.5-Flash	1.00	.980	.850	.710	.675
Llama-4-Scout	.820	.524	.468	.371	.372

Table B.5: Efficient Model Performance ($F1$) on Reasoning Decomposition Suite.

Task	Solved Steps	Claude-3.5 Haiku	Gemma-3	Gemini-2.5 Flash	Llama-4 Scout
EC	<i>None</i>	.402	.437	.595	.267
	<i>AE</i>	.706	.444	.854	.565
EV	<i>None</i>	.326	.262	.544	.223
	<i>AE</i>	.544	.280	.712	.242
	<i>EC</i>	.685	.492	.827	.347
NA	<i>None</i>	.137	.248	.513	.167
	<i>AE</i>	.190	.187	.759	.309
	<i>EC</i>	.322	.289	.513	.297
	<i>EV</i>	.372	.224	.462	.290
OE	<i>None</i>	.113	.165	.592	.148
	<i>AE</i>	.230	.095	.718	.413
	<i>EC</i>	.319	.319	.658	.276
	<i>EV</i>	.361	.347	.788	.291
	<i>NA</i>	.039	.020	.507	.291

Table B.6: Large Model Performance ($F1$) on Reasoning Decomposition Suite.

Task	Solved Steps	GPT-4.1	Llama-4 Maverick	DeepSeek V3
EC	<i>None</i>	.265	.258	.367
	<i>AE</i>	.629	.533	.724
EV	<i>None</i>	.271	.219	.396
	<i>AE</i>	.593	.512	.559
	<i>EC</i>	.785	.798	.750
NA	<i>None</i>	.481	.197	.288
	<i>AE</i>	.511	.445	.473
	<i>EC</i>	.361	.420	.297
	<i>EV</i>	.441	.503	.397
OE	<i>None</i>	.305	.214	.387
	<i>AE</i>	.519	.496	.551
	<i>EC</i>	.561	.473	.462
	<i>EV</i>	.773	.473	.571
	<i>NA</i>	.450	.305	.473

Table B.7: Reasoning Model Performance ($F1$) on Reasoning Decomposition Suite.

Task	Solved Steps	GPT-5	o3	o4-mini	Sonnet-4	Gemini Pro	DeepSeek R1
EC	<i>None</i>	.714	.742	.499	.534	.714	.575
	<i>AE</i>	.983	.983	.789	.924	.958	.803
EV	<i>None</i>	.671	.700	.549	.402	.668	.552
	<i>AE</i>	.800	.814	.695	.716	.811	.610
	<i>EC</i>	.836	.875	.882	.869	.851	.847
NA	<i>None</i>	.650	.681	.427	.414	.662	.582
	<i>AE</i>	.917	.907	.649	.735	.840	.709
	<i>EC</i>	.588	.529	.524	.519	.567	.499
	<i>EV</i>	.557	.539	.407	.531	.563	.458
OE	<i>None</i>	.611	.630	.485	.462	.684	.540
	<i>AE</i>	.788	.765	.621	.658	.788	.649
	<i>EC</i>	.780	.734	.658	.658	.857	.601
	<i>EV</i>	.876	.795	.649	.726	.876	.693
	<i>NA</i>	.658	.693	.701	.621	.649	.649

Table B.8: Efficient Model Performance ($F1$) on Baseline, Distractor, Sycophancy, and Obfuscation Robustness Suites.

Task	Suite	Claude-3.5 Haiku	Gemma-3	Gemini-2.5 Flash	Llama-4 Scout
AE	<i>Baseline</i>	.618	.528	.745	.577
	<i>Distractor</i>	.645	.575	.831	.510
	<i>Sycophancy</i>	.507	.377	.831	.442
	<i>Obfuscation</i>	.509	.390	.796	.457
EC	<i>Baseline</i>	.478	.466	.772	.394
	<i>Distractor</i>	.453	.477	.739	.459
	<i>Sycophancy</i>	.312	.385	.749	.314
	<i>Obfuscation</i>	.319	.359	.719	.390
EV	<i>Baseline</i>	.382	.264	.715	.333
	<i>Distractor</i>	.378	.308	.677	.298
	<i>Sycophancy</i>	.236	.199	.701	.271
	<i>Obfuscation</i>	.262	.201	.652	.237
NA	<i>Baseline</i>	.297	.356	.763	.316
	<i>Distractor</i>	.154	.189	.753	.194
	<i>Sycophancy</i>	.159	.198	.776	.152
	<i>Obfuscation</i>	.112	.115	.787	.130
OE	<i>Baseline</i>	.276	.400	.817	.387
	<i>Distractor</i>	.058	.000	.621	.020
	<i>Sycophancy</i>	.165	.165	.718	.131
	<i>Obfuscation</i>	.039	.000	.582	.000

Table B.9: Large Model Performance ($F1$) on Baseline, Distractor, Sycophancy, and Obfuscation Robustness Suites.

Task	Suite	GPT-4.1	Llama-4 Maverick	DeepSeek V3
AE	<i>Baseline</i>	.837	.820	.813
	<i>Distractor</i>	.835	.720	.802
	<i>Sycophancy</i>	.837	.748	.764
	<i>Obfuscation</i>	.788	.755	.710
EC	<i>Baseline</i>	.313	.341	.478
	<i>Distractor</i>	.335	.408	.472
	<i>Sycophancy</i>	.143	.303	.219
	<i>Obfuscation</i>	.095	.277	.239
EV	<i>Baseline</i>	.285	.369	.407
	<i>Distractor</i>	.273	.355	.401
	<i>Sycophancy</i>	.118	.232	.208
	<i>Obfuscation</i>	.105	.192	.195
NA	<i>Baseline</i>	.555	.416	.461
	<i>Distractor</i>	.467	.271	.328
	<i>Sycophancy</i>	.517	.321	.198
	<i>Obfuscation</i>	.450	.189	.186
OE	<i>Baseline</i>	.601	.413	.507
	<i>Distractor</i>	.462	.131	.214
	<i>Sycophancy</i>	.246	.374	.246
	<i>Obfuscation</i>	.165	.113	.113

Table B.10: Reasoning Model Performance ($F1$) on Baseline, Distractor, Sycophancy, and Obfuscation Robustness Suites.

Task	Suite	GPT-5	o3	o4-mini	Sonnet-4	Gemini Pro	DeepSeek R1
AE	<i>Baseline</i>	.785	.808	.825	.810	.835	.835
	<i>Distractor</i>	.843	.845	.855	.865	.895	.887
	<i>Sycophancy</i>	.792	.805	.823	.830	.843	.827
	<i>Obfuscation</i>	.820	.827	.842	.820	.852	.830
EC	<i>Baseline</i>	.824	.816	.725	.597	.840	.755
	<i>Distractor</i>	.837	.832	.636	.535	.839	.738
	<i>Sycophancy</i>	.858	.871	.528	.486	.833	.533
	<i>Obfuscation</i>	.779	.828	.469	.419	.795	.435
EV	<i>Baseline</i>	.771	.737	.602	.532	.767	.637
	<i>Distractor</i>	.755	.734	.590	.487	.784	.656
	<i>Sycophancy</i>	.804	.787	.456	.411	.760	.510
	<i>Obfuscation</i>	.792	.778	.468	.336	.759	.432
NA	<i>Baseline</i>	.862	.868	.644	.511	.865	.789
	<i>Distractor</i>	.850	.764	.573	.557	.837	.705
	<i>Sycophancy</i>	.867	.835	.560	.539	.832	.818
	<i>Obfuscation</i>	.826	.727	.467	.474	.827	.674
OE	<i>Baseline</i>	.844	.876	.734	.611	.870	.817
	<i>Distractor</i>	.529	.148	.400	.305	.639	.507
	<i>Sycophancy</i>	.742	.765	.621	.450	.758	.742
	<i>Obfuscation</i>	.450	.182	.305	.347	.701	.291

Table B.11: Efficient Model Performance ($F1$) on Asset Scaling Suite.

Task	Asset Count	Claude-3.5 Haiku	Gemma-3	Gemini-2.5 Flash	Llama-4 Scout
EC	2	.484	.445	.927	.426
	4	.452	.420	.902	.382
	6	.432	.404	.898	.367
	8	.412	.434	.888	.369
EV	2	.338	.362	.910	.367
	4	.343	.319	.899	.334
	6	.318	.235	.892	.270
	8	.328	.254	.866	.314
NA	2	.411	.370	.868	.416
	4	.314	.276	.848	.354
	6	.191	.242	.832	.290
	8	.140	.266	.845	.239
OE	2	.462	.551	.942	.425
	4	.131	.148	.883	.230
	6	.058	.182	.817	.131
	8	.020	.077	.810	.077

Table B.12: Large Model Performance ($F1$) on Asset Scaling Suite.

Task	Asset Count	GPT-4.1	Llama-4 Maverick	DeepSeek V3
EC	2	.441	.514	.473
	4	.345	.465	.521
	6	.363	.479	.520
	8	.327	.470	.439
EV	2	.437	.447	.426
	4	.416	.438	.477
	6	.349	.437	.457
	8	.364	.447	.429
NA	2	.588	.528	.562
	4	.545	.480	.444
	6	.546	.472	.483
	8	.531	.433	.411
OE	2	.611	.630	.667
	4	.462	.450	.387
	6	.276	.333	.361
	8	.182	.182	.182

Table B.13: Reasoning Model Performance ($F1$) on Asset Scaling Suite.

Task	Asset Count	GPT-5	o3	o4-mini	Sonnet-4	Gemini Pro	DeepSeek R1
EC	2	.964	.968	.794	.807	.951	.809
	4	.981	.961	.720	.770	.950	.801
	6	.941	.965	.754	.712	.958	.802
	8	.941	.945	.701	.706	.935	.804
EV	2	.949	.956	.763	.736	.934	.774
	4	.968	.945	.737	.721	.946	.752
	6	.957	.918	.720	.658	.946	.742
	8	.955	.950	.683	.666	.935	.729
NA	2	.927	.955	.737	.676	.883	.870
	4	.898	.962	.734	.633	.898	.850
	6	.918	.942	.687	.605	.889	.870
	8	.938	.949	.717	.643	.897	.867
OE	2	.947	.995	.824	.795	.953	.919
	4	.936	.953	.693	.675	.969	.837
	6	.901	.913	.658	.571	.925	.734
	8	.913	.901	.485	.496	.919	.667

B.7 Task Prompt Examples

Temporal Reasoning Suite: Task AE

Determine which state or federal exemptions may be claimed by the Debtor(s) under the provided statutes.

Your answer to this task must be based solely on applying the provided Federal and State statutes to the given facts.

Response Format: Your response must end with your final answer in the following template: FINAL ANSWER: [YOUR FINAL ANSWER]. Your final answer must consist of only a comma-separated list of jurisdictions, without any additional text. States should be identified by name. If federal exemptions are allowed, include 'Federal' in the list. Example response format: FINAL ANSWER: Alaska, Federal

Facts: Luis Gonzalez (hereinafter the Debtor) filed for bankruptcy on 14 July 2024. After moving their household to Delta, Pennsylvania on Saturday, March 21st, 2020, Luis Gonzalez eventually relocated to Marana, Arizona on 29th of February 2024.

Statutes:

Solution

Federal, Pennsylvania

Figure B.1: Example Task AE (Allowable Exemptions) prompt from the Distractor Robustness evaluation suite; statutes abridged for brevity.

Distractor Robustness Suite: Task EC

For each asset in the estate, identify all applicable exemptions under which that asset may be protected.

Your answer to this task must be based solely on applying the provided Federal and State statutes to the given facts. If the task involves a married couple, assume all assets mentioned are jointly owned, with each spouse holding an equal undivided interest, unless explicitly stated otherwise. Assume all assets are held for the personal use of the Debtor(s), unless explicitly stated otherwise.

Response Format: Your response must end with your final answer in the following template: FINAL ANSWER: [YOUR FINAL ANSWER]. Your final answer must consist of only valid JSON in the exact format specified below. Provide your final answer as a JSON object where: each key is the exact asset description provided in the fact pattern, and each value is an array of applicable exemption citations. Example response format: FINAL ANSWER: "1981 DeLorean DMC-12": ["11 U.S.C. § 522(d)(2)", "11 U.S.C. § 522(d)(5)"]

Facts: Megha and Dalia Joshi (hereinafter the Debtors) filed for bankruptcy on March 7th 2024. The Debtors began living in Litchfield Park, Arizona on 19 March 2012, but relocated to Waunakee, Wisconsin on 9.10.2021. For the 28 days following that date, Megha stayed in Patagonia, Arizona to complete a boater safety course and obtain a state-issued boating certificate, a new requirement for their job as a marine biologist. While there, they signed a rental agreement (Contract #R-781) for a boat slip at a local marina to berth the assigned training vessel for the duration of the course. The Joshis chose to relocate their household to Weyauwega, Wisconsin on 12th of September, 2021. Records show that Megha and Dalia Joshi possess a compact Bluetooth speaker with splash-resistant casing with a value of \$400.00. Megha and Dalia own a calico cat worth \$145.00. The Debtor's name is listed on a UTMA savings account, and the corresponding Form 1099-INT is mailed to their address. The account holds a balance of \$8,150, which originated as an irrevocable gift from the Debtor's brother to the account's beneficiary, the Debtor's 14 year-old nephew. The known assets of Megha and Dalia Joshi include a woven tapestry wall hanging with bohemian motif worth \$1,305.00 and a Hi-Point C9 9mm pistol appraised at \$195.00.

Statutes:

Solution

```
{"compact Bluetooth speaker with splash-resistant casing": ["11 U.S.C. § 522(d)(3)", "11 U.S.C. § 522(d)(5)", "Wis. Stat. § 815.18(3)(d)"], "calico cat": ["11 U.S.C. § 522(d)(3)", "11 U.S.C. § 522(d)(5)", "Wis. Stat. § 815.18(3)(d)"], "woven tapestry wall hanging with bohemian motif": ["11 U.S.C. § 522(d)(3)", "11 U.S.C. § 522(d)(5)", "Wis. Stat. § 815.18(3)(d)"], "Hi-Point C9 9mm pistol": ["11 U.S.C. § 522(d)(5)", "Wis. Stat. § 815.18(3)(d)]}
```

Figure B.2: Example Task EC (Exemption Classification) prompt from the Distractor Robustness evaluation suite; statutes abridged for brevity.

Reasoning Decomposition Suite: Task OE

Determine the optimal set of exemptions to best protect the assets in the estate. The goal is to minimize the total dollar value of non-exempt assets. If property may be exempted under multiple jurisdictions, you must select the jurisdiction that would result in the best solution.

Your answer to this task must be based solely on applying the provided Federal and State statutes to the given facts. If the task involves a married couple, assume all assets mentioned are jointly owned, with each spouse holding an equal undivided interest, unless explicitly stated otherwise. Assume all assets are held for the personal use of the Debtor(s), unless explicitly stated otherwise.

Facts: Tobias and Leon Fischer (hereinafter the Debtors) filed for bankruptcy on 10th day of January, 2024. Ownership of the small mountain cabin used year-round as the principal residence, priced at \$49,500.00, is claimed by Tobias and Leon Fischer. The pair of suede ankle boots with zipper closure currently owned by the Fischers carries a value of \$225.00. The Debtors assert ownership of an audiologist prescribed custom-fit hearing aids with behind-the-ear receiver and noise filtering with a market value of \$1,425.00. A 14-karat gold engagement band with engraving on the inner surface valued at \$770.00 is under the ownership of Tobias and Leon. An oxygen concentrator with portable carry cart and backup battery (physician authorized) appraised at \$3,250.00 is the property of Tobias and Leon Fischer. A disclosure of assets by the Fischers reports a floor-length curtains with floral embroidery with a current value of \$280.00.

Solved Reasoning Steps: The following reasoning steps have already been solved. Use this information to aid you in completing the remainder of the task. All applicable exemptions have been identified below for each asset in the estate. The small mountain cabin used year-round as the principal residence may be exempted under 11 U.S.C. § 522(d)(1), 11 U.S.C. § 522(d)(5), and Or. Rev. Stat. § 18.395(1). The pair of suede ankle boots with zipper closure may be exempted under 11 U.S.C. § 522(d)(3), 11 U.S.C. § 522(d)(5), Or. Rev. Stat. § 18.345(1)(b), and Or. Rev. Stat. § 18.345(1)(p). The audiologist prescribed custom-fit hearing aids with behind-the-ear receiver and noise filtering may be exempted under 11 U.S.C. § 522(d)(5), 11 U.S.C. § 522(d)(9), Or. Rev. Stat. § 18.345(1)(h), and Or. Rev. Stat. § 18.345(1)(p). The 14-karat gold engagement band with engraving on the inner surface may be exempted under 11 U.S.C. § 522(d)(4), 11 U.S.C. § 522(d)(5), Or. Rev. Stat. § 18.345(1)(b), and Or. Rev. Stat. § 18.345(1)(p). The oxygen concentrator with portable carry cart and backup battery (physician authorized) may be exempted under 11 U.S.C. § 522(d)(5), 11 U.S.C. § 522(d)(9), Or. Rev. Stat. § 18.345(1)(h), and Or. Rev. Stat. § 18.345(1)(p). The floor-length curtains with floral embroidery may be exempted under 11 U.S.C. § 522(d)(3), 11 U.S.C. § 522(d)(5), Or. Rev. Stat. § 18.345(1)(f), and Or. Rev. Stat. § 18.345(1)(p).

Statutes:

Solution

```
{
  "small mountain cabin used year-round as the principal residence": [
    {
      "citation": "11 U.S.C. § 522(d)(5)",
      "claim_value": 30850
    },
    {
      "citation": "11 U.S.C. § 522(d)(1)",
      "claim_value": 18650
    }
  ],
  "pair of suede ankle boots with zipper closure": [
    {
      "citation": "11 U.S.C. § 522(d)(3)",
      "claim_value": 225
    }
  ],
  "audiologist prescribed custom-fit hearing aids with behind-the-ear receiver and noise filtering": [
    {
      "citation": "11 U.S.C. § 522(d)(9)",
      "claim_value": 1425
    }
  ],
  "14-karat gold engagement band with engraving on the inner surface": [
    {
      "citation": "11 U.S.C. § 522(d)(4)",
      "claim_value": 770
    }
  ],
  "oxygen concentrator with portable carry cart and backup battery (physician authorized)": [
    {
      "citation": "11 U.S.C. § 522(d)(9)",
      "claim_value": 3250
    }
  ],
  "floor-length curtains with floral embroidery": [
    {
      "citation": "11 U.S.C. § 522(d)(3)",
      "claim_value": 280
    }
  ]
}
```

Figure B.3: Example Task OE (Optimal Exemptions) prompt from the Distractor Robustness evaluation suite; response format instructions and statutes abridged for brevity.

B.8 Statute Source by Jurisdiction

Table B.14: Sources of Statutes by Jurisdiction.

Jurisdiction	Source
Federal	https://uscode.house.gov
Arizona	https://www.azleg.gov/arstitle/
Illinois	https://www.ilga.gov/legislation/ilcs/ilcs.asp
Oregon	https://www.oregonlegislature.gov/bills_laws/pages/ors.aspx
Pennsylvania	https://www.palegis.us/statutes/consolidated
Wisconsin	https://docs.legis.wisconsin.gov/statutes/statutes/815