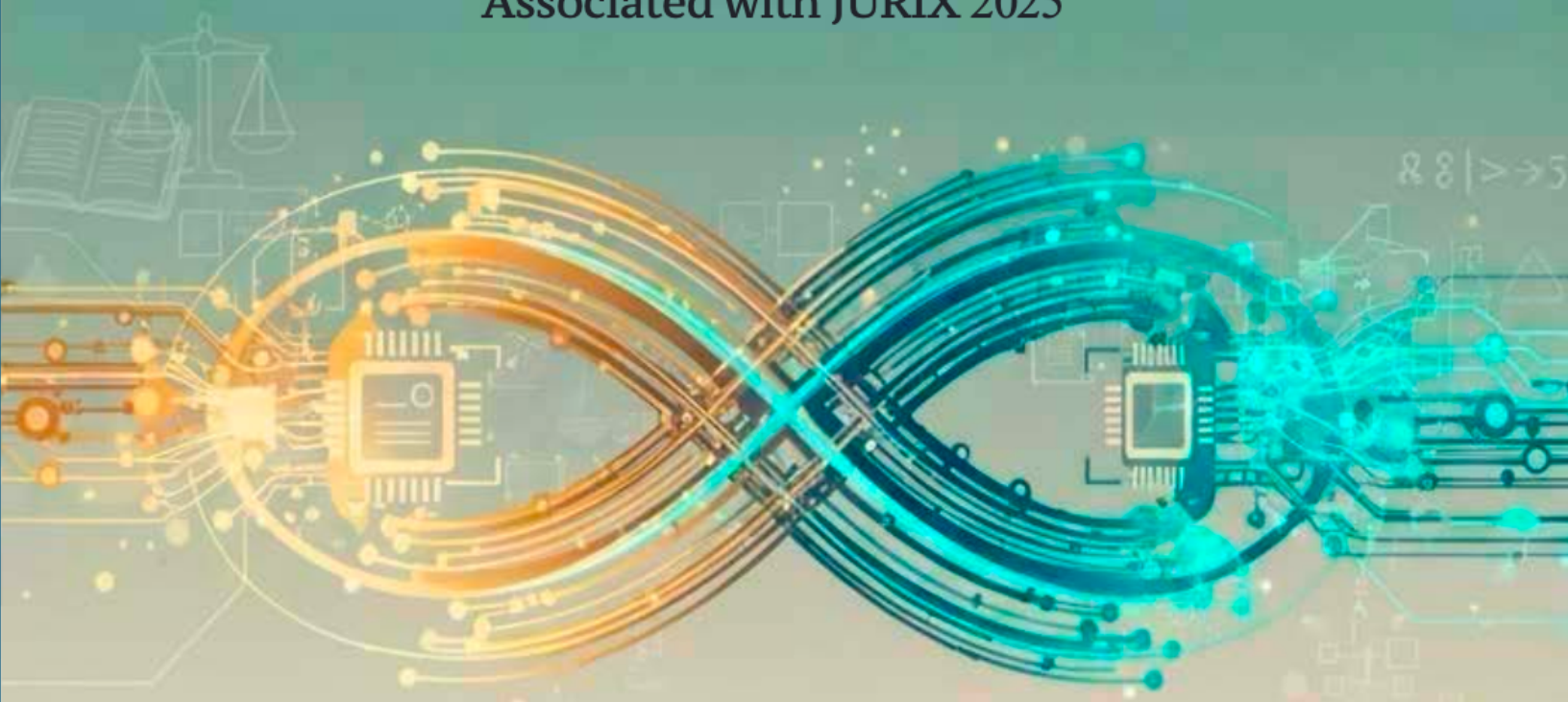
PROCEEDINGS OF THE INTERNATIONAL WORKSHOP ON TRANSLATING NATURAL LEGAL LANGUAGE INTO FORMAL REPRESENTATIONS

NLL2FR 2025

Associated with JURIX 2025



December 9, 2025 | Torino, Italy

Edited by: Ken Satoh, Georg Borges, Hannes Westermann, and May Myo Zin

Proceedings of the International Workshop on Translating Natural Legal Language into Formal Representations

(NLL2FR 2025)

in association with

the 38th International Conference on Legal Knowledge and Information Systems (JURIX 2025)

NLL2FR 2025 Co-Chairs

Ken Satoh, Center for Juris-Informatics, Japan
Georg Borges, Saarland University, Germany
Hannes Westermann, Maastricht University, The Netherlands
May Myo Zin, Center for Juris-Informatics, Japan

December 9, 2025

Preface

This volume contains the papers presented at NLL2FR 2025: The International Workshop on Translating Natural Legal Language into Formal Representation, held on December 9, 2025, in association with the 38th International Conference on Legal Knowledge and Information Systems (JURIX 2025).

As automation and AI systems become increasingly integrated into legal practice, the need to bridge human-readable and machine-executable legal information grows ever more urgent. Laws, regulations, contracts, and case descriptions are typically written for human readers, who rely on shared background knowledge, professional experience, and contextual understanding. Unlike humans, automated systems such as compliance-checking tools and legal decision-support applications require these legal texts to be rewritten in a clear, structured, and formally precise way so they can be interpreted and processed automatically. The NLL2FR workshop is dedicated to developing and evaluating methods for translating natural legal language into such representations, helping to bridge the gap between human-centred legal texts and machine-executable legal knowledge. It encourages contributions that explore methodologies, tools, theoretical frameworks, and practical pipelines for analyzing, structuring, and formalizing legal texts. The goal is to foster an exchange of ideas among communities that address different aspects of the translation problem, from linguistic analysis to formal modeling and executable representations.

For the 2025 edition, we received 16 submissions, each reviewed by three reviewers, and accepted 14 papers (12 long and 2 short) for presentation and discussion. These contributions collectively reflect the breadth and growing maturity of the field. They include work on automated legal text annotation and information extraction, argument modeling and factor identification, the formalization and structuring of legal texts, legal NER, the development of legal ontologies, and integrated frameworks that link natural-language norms to machine-executable representations. The approaches range from carefully engineered rule-based systems and logic-based formalisms to methods leveraging large language models (LLMs). Many papers combine both paradigms—often in complementary or comparative ways—to convert natural-language legal sources into precise, operational representations. Together, they highlight the progress made and the challenges that remain in understanding and structuring the complex language of the law for computational use and provide a valuable foundation for future research.

We extend our sincere thanks to all authors for their submissions and to the members of the Program Committee for their thorough evaluations, insightful comments, and unwavering support of the review process. We also thank the JURIX 2025 organizers for their support in hosting the workshop.

December 9, 2025

Ken Satoh
Georg Borges
Hannes Westermann
May Myo Zin

Table of Contents

Program Committee

Akira Shimazu	Japan Advanced Institute of Science and Technology
Adrian Paschke	Freie Universität Berlin
Michał Araszkiewicz	Jagiellonian University
Adam Wyner	Swansea University
Satoshi Tojo	Asia University
Le-Minh Nguyen	Japan Advanced Institute of Science and Technology
Anelia Kurteva	University of Birmingham
Vu Tran	Japan Advanced Institute of Science and Technology
Wachara Fungwacharakorn	Cener for Juris-Informatics
Guido Governatori	Central Queensland University
Katsumi Nitta	Center for Juris-Informatics
Yuntao Kong	Center for Juris-Informatics
Davide Liga	University of Luxembourg
Aye Aye Mar	Japan Advanced Institute of Science and Technology
Su Myat Noe	National Institue of Informatics
Makoto Nakamura	Niigata Institute of Technology
María Navas-Loro	Universidad Politécnica de Madrid
Ha-Thanh Nguyen	National Institute of Informatics
Minh-Phuong Nguyen	Japan Advanced Institute of Science and Technology
Livio Robaldo	Swansea University
Diogo Sasdelli	University for Continuing Education Krems
Sabine Wehnert	Otto von Guericke University Magdeburg
Xue Jieying	Center for Juris-Informatics

Error Analysis in LLM-based Factor Identification and Discovery

Wachara Fungwacharakorn[1[0000−0001−9294−3118]], May Myo Zin[1[0000−0003−1315−7704]], and Ken Satoh[1[0000−0002−9309−4602]]

Center of Juris-Informatics, ROIS-DS, Tokyo, Japan
{wacharaf, maymyozin, ksatoh}@nii.ac.jp

Abstract. Factors are fundamental features for representing legal cases computationally, and AI and Law researchers have long explored the use of machine learning and large language models (LLMs) for factor extraction. This paper investigates errors in two key LLM-based factor extraction tasks: *factor identification* and *factor discovery*. We report experimental results across two legal domains – trade secrets and credit card applications – using two LLMs, GPT-4o and Gemini 2.0 Flash Lite. The results show that context-dependent factors and short case descriptions often lead to errors in factor identification, while overlapping factors and long case descriptions tend to cause errors in factor discovery. These findings suggest strategies such as adapting extraction methods to factor characteristics and applying legal case summarization techniques in factor extraction to mitigate errors.

Keywords: Factor · Legal Reasoning · Large Language Models

1 Introduction

Factors are fundamental components in legal analysis and computational legal reasoning [10]. Each factor captures factual patterns that tend to strengthen or weaken a position in a legal dispute [2]. For example, in the domain of trade secret law, factual patterns such as *whether the trade secret has sufficient security measures* and *whether the trade secret is easy to reverse engineer* are considered factors that influence the decision [11]. In AI and Law, factors are used to model case-based legal reasoning, such as HYPO [11], CATO [1], and models of precedential constraint [9]. In those models, reasoning proceeds by identifying similarities and differences between factor sets as representations to those factual situations of the cases.

As legal AI systems advance, machine learning and large language models (LLMs) have emerged as promising tools for automating factor identification [4, 7, 12] as well as factor discovery [8]. Traditionally, factor identification and discovery tasks are performed by trained legal professionals to identify and discover which factual elements are relevant to a particular outcome, as discussed in early case-based legal reasoning systems [1]. However, the reliability of LLM-based factor identification and discovery remains understudied, particularly regarding

how characteristics of the factors themselves influence model performance. Understanding these error patterns is critical for deploying LLMs responsibly in legal contexts, where mistakes can have serious consequences for case outcomes and access to justice.

This paper systematically investigates errors in LLM-based factor identification and discovery, which are two fundamental tasks in factor extraction. We focus our study on two legal domains. The first one is the trade secret domain, with factors that have been long studied in AI and Law research. The second one is the credit card application domain, with factors that were newly constructed. We selected four pro-plaintiff factors and four pro-defendant factors from each domain and generated a case description for each possible set of selected factors. We explore automating those tasks based on two LLMs: GPT-4o and Gemini-2.0-Flash-Lite. Our analysis shows that high-performing factors tend to be concrete and established by contracts while low-performing factors tend to be complex and context-dependent. Furthermore, we found that LLMs tend to better identify factors from longer descriptions but better discover factors from shorter ones.

The paper is structured as follows. Section 2 provides background and related work on factor identification and factor discovery tasks. Section 3 provides an experiment design. Section 4 reports the result of the experiment. Section 5 discusses the result. Finally, Section 6 concludes this paper.

2 Background and Related Work

In this section, we provide background on factor-based legal reasoning and related tasks. In factor-based legal reasoning, each factor is assumed to *favor* one position on a legal dispute: either the position of plaintiff (π), or the position of defendant (δ). Formally, let $\mathcal{F}$ be the set of all factors in a particular legal domain, where each factor $f \in \mathcal{F}$ is either a *pro-plaintiff* factor (noted by a superscript f^{π}) or a *pro-defendant* factor (noted by a superscript f^{δ}). Given any plain text T describing a factual situation of a case, called a *case description*, it assumes that there is a set of factors $X_T \subseteq \mathcal{F}$ that represents the situation described by T.

Next, we define the two fundamental tasks in factor extraction that form the focus of this paper, namely *factor identification* and *factor discovery*. These two tasks are closely relevant to membership and equivalence queries [3] in machine learning. We provide definitions based on those queries and related work based on each task as follows.

2.1 Factor Identification

Factor Identification, which is also known as *factor classification*, is the task of determining whether or not a given factor is required to represent the factual situation of the present case. We formally define the task, based on a membership query [3], as follows.

Definition 1 (factor identification: *ide*). *Given a case description T and a factor $x \in \mathcal{F}$, the* factor identification function*, denoted as $ide(T, x)$, returns 'y' if $x \in X_T$, or 'n' otherwise.*

Previous research [4, 7, 12] in AI and Law has explored various machine learning techniques to automate this task. They found that most legal decisions typically depend on a relatively small subset of commonly used factors. However, it requires more efforts in order to provide more transparent explanations that legal professionals could understand and verify. Their works mainly established that automating factor identification is computationally feasible but highlighted the ongoing challenge of explainability requirements essential for legal applications.

2.2 Factor Discovery

Factor discovery is the task to define a new factor that is not included in the given set of factors, but required to represent the factual situation of the present case. We formally define the task, based on an equivalence query [3], as follows.

Definition 2 (factor discovery: *dis*). *Given a case description T and a finite set of factors $Y \subseteq \mathcal{F}$, the* factor discovery function*, denoted as $dis(T, Y)$, returns one factor $x \in X_T \backslash Y$ if it exists (i.e., $X_T \backslash Y$ is not empty), or 'n' otherwise.*

Recent work [8] has investigated how well LLMs could automatically extract domain representations as factors from raw legal opinions. They found that while fully automated factor discovery achieved only weak to moderate performance, human-in-the-loop approaches showed feasibility. A critical limitation they identified was that all models, including humans, occasionally returned legally irrelevant factors, highlighting the need to incorporate fundamental legal knowledge into model prompting. This work suggests that LLM-based factor discovery holds promise for assisting legal researchers and practitioners in organizing case law and developing predictive models, but requires careful human oversight to ensure legal relevance and accuracy.

3 Experiment Design

In this section, we present the workflow of our experiments on factor identification and discovery using LLMs, as depicted in Fig. 1. Since it is infeasible to assess all factors, we define a *factor list* $\mathbb{F} \subseteq \mathcal{F}$ consisting of selected factors in a certain domain. Then, for each subset X of $\mathbb{F}$, we instruct LLMs to generate a case description T with the expectation that the set of factors X_T representing T is identical to X. We denote this generation as $gen(X)$. To ensure each subset can be a representation of a situation, $\mathbb{F}$ is checked to be conflict-free; that is, two factors can occur at the same time. We also instruct LLMs to simulate $ide(T, p)$ and $dis(T, P)$ as defined in the previous section. We detail each component in the pipeline as follows.

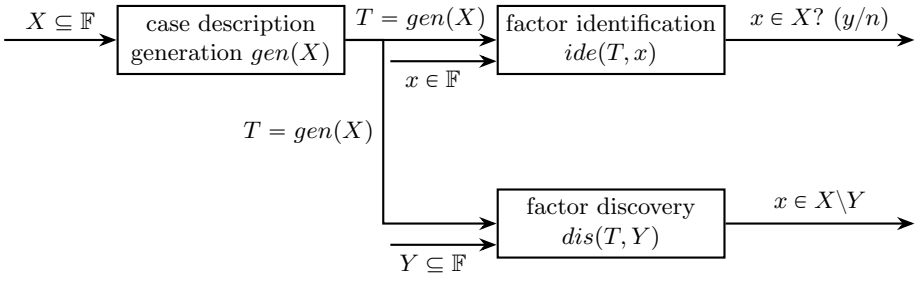


Fig. 1: Experiment Workflow

3.1 Factor Lists

In this paper, we consider two legal domains. The first domain is the domain of trade secret law (π: the defendant misappropriated the trade secret; δ: the defendant did not misappropriate the trade secret), as used in CATO [1]. We selected four pro-plaintiff factors and four pro-defendant factors that were used to represent publicly available case documents and discussed in several prior studies (e.g., [5, 12, 13]). The CATO factor list, denoted as $\mathbb{F}_{cato}$, consists of the following factors:

1. **mea**$^{\pi}$: The plaintiff implemented sufficient security measures to maintain the secrecy of the trade secret information.
2. **hir**$^{\pi}$: The defendant hired the plaintiff's key personnel with trade secret information.
3. **ide**$^{\pi}$: The defendant's product is identical or nearly identical.
4. **mat**$^{\pi}$: The defendant brought or reused materials from the plaintiff's project.
5. **dis**$^{\delta}$: The plaintiff disclosed the trade secret information during business negotiations.
6. **out**$^{\delta}$: The plaintiff shared the trade secret information to people outside the organization.
7. **gen**$^{\delta}$: The defendant's product could have been developed without using the plaintiff's trade secrets.
8. **pub**$^{\delta}$: The plaintiff disclosed information in public forum.

The second factor domain is the domain of credit card application (π: the credit card application was rejected; δ the credit card application was accepted). Althrough these factors have been used in our recent work [6], they have not been publicly available and discussed at the time of the experiment. To make it comparable with the CATO factor list, we selected only the first four pro-plaintiff factors and the first four pro-defendant factors from the original list in [6]. The CREDIT factor list, denoted as $\mathbb{F}_{credit}$, consists of the following factors:

1. **cre**$^{\pi}$: The applicant has high number of recent credit inquiries.
2. **mis**$^{\pi}$: The applicant has a history of missed or late payments.

3. $\mathbf{loi}^{\pi}$: The applicant has insufficient income.
4. $\mathbf{lch}^{\pi}$: The applicant has limited credit history.
5. $\mathbf{dir}^{\delta}$: The applicant has a low debt-to-income ratio.
6. $\mathbf{emp}^{\delta}$: The applicant has a long and stable employment history.
7. $\mathbf{pay}^{\delta}$: The applicant has a consistent payment history on existing loans.
8. $\mathbf{dec}^{\delta}$: The applicant has significant assets declared.

3.2 Case Description Generation

To generate case descriptions, we instruct LLMs using the following prompt template:

Prompt 1.1: Case Description Generation

```
TASK: To generate an example of case description in {domain} domain that
requires all factors in an included factor set to represent the siuation
of the case, and prevents using any factors in an excluded factor set to
represent the situation of the case.
INPUT: You will be provided with an included factor set and an excluded
    factor set.
Included Factor Set:
   {included_factor_set}
Excluded Factor Set:
   {excluded_factor_set}
OUTPUT: Generate a concise case description in plain text.
Do NOT explicitly use the same words as those in factors.
Do NOT include an outcome of the case.
```

The prompt is to generate a case description from a subset of factor lists $X \subseteq \mathbb{F}$, called an *included factor set*, and $\mathbb{F} \backslash X$, called an *excluded factor set*. This aims to generate an example of case description that uses all factors in X and does not use any factor in $\mathbb{F} \backslash X$ to represent the corresponding situation. We generate a case description for every non-empty proper subset of factor list. Since our factor lists each have eight factors, $2^8 - 2 = 254$ case descriptions are generated (the two excluded sets are the empty set and the whole factor set). We used two large language models in the experiments: GPT-4o and GEMINI-2.0-FLASH-LITE so there are two generated descriptions for each subset. Here is one example of case description generated by GPT-4o, for a subset $\{\mathbf{mea}^{\pi}, \mathbf{pub}^{\delta}\}$ of $\mathbb{F}_{cato}$:

> In a recent case involving alleged misappropriation of proprietary information, the plaintiff, a technology company, maintained strict protocols to secure their proprietary algorithms, ensuring that employees could only access this information under stringent clearance levels and control measures. Despite these efforts, the algorithms were inadvertently made available in a public online repository due to a misconfiguration by a third-party vendor responsible for their IT services. [...]

Here is one example of case description generated by GEMINI-2.0-FLASH-LITE, for a subset $\{\mathbf{mea}^{\pi}, \mathbf{pub}^{\delta}\}$ of $\mathbb{F}_{cato}$:

A software company, Alpha Corp, stored its proprietary source code on a password-protected server with limited employee access. Despite these measures to protect the confidentiality of its development process, Alpha Corp presented its new program at a large industry conference, allowing attendees to freely examine and test it. This public demonstration provided everyone the opportunity to inspect the inner workings of the software.

3.3 Factor Identification

To identify whether a given factor is used to represent the situation of the case, we instruct LLM using the following prompt template:

Prompt 1.2: Factor Identification

```
TASK: To identify whether a given factor is used to represent the
    situation of the case.
INPUT: You will be provided with a case description and a factor.
Case Description:
   {description}
Factor:
   {factor}
OUTPUT:
'YES' if the factor is used to represent the situation of the case.
'NO' if the factor is NOT used to represent the situation of the case.
```

The experiment on this task is as follows. For each case description generated from a subset of the factor list, we tried to identify every factor in the full factor list. If the factor is in the original subset and is correctly identified by the LLM, it is counted as '*true positive*' (TP). If the factor is in the original subset but is not identified by the LLM, it is counted '$false\ negative$' (FN). If the factor is not in the original subset but it is identified by the LLM, it is counted as '$false\ positive$' (FP). After considering all generated case descriptions, we measure the performance of identifying each factor using precision, recall, and the F1-score, as in general classification performance metrics, defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

3.4 Factor Discovery

To discover a new factor to represent the situation of the case, we instruct LLMs using the following prompt template:

Prompt 1.3: Factor Discovery

```
TASK: To define a new factor that should be introduced beyond the factor
    given in the factor list to represent the situation of the case.
INPUT: You will be provided with a factor list and a case description.
Case Description:
   {description}
Factor List:
   {factor_list}
OUTPUT:
Define a new factor that should be introduced in a concise plain text.
Do NOT provide a reason or an explanation.
```

The experiment on this task is as follows. For each case description generated from a subset of the factor list, we took one factor out from the subset and provided the remaining set as the input factor list to the LLM. Since the descriptions of the discovered factor provided by the LLM would not be identical to those descriptions that we have, we instruct LLMs using the following prompt template to check whether two factor descriptions describe the same factor.

Prompt 1.4: Factor Checking

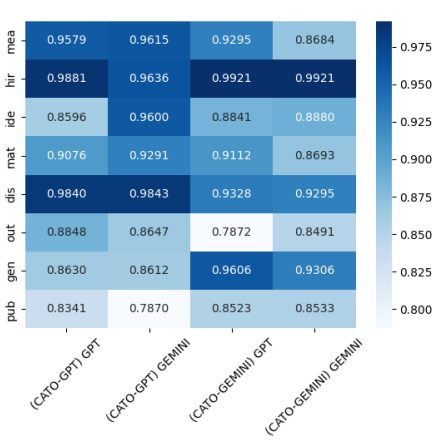
```
TASK: To check whether two factor descriptions describe the same factor.
INPUT: You will be provided with two factor descriptions.
Factor Description 1:
   {description1}
Factor Description 2:
   {description2}
OUTPUT:
'YES' if two factor descriptions describe the same factor.
'NO' if two factor descriptions do NOT describe the same factor.
```

If the description of the discovered factor and the description of the taken out factor describe the same thing, it is counted as a '*hit*'. After considering all generated case descriptions, we measure the performance of discovering each factor using a hit rate, defined as follows:

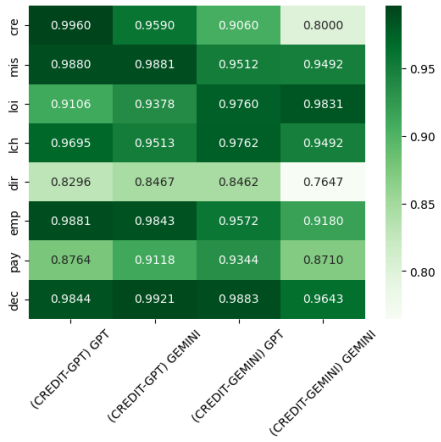
$$\text{Hit Rate} = \frac{\text{Number of Times the Factor Taken Out and Discovered (Hit)}}{\text{Number of Times the Factor Taken Out}}$$

4 Result

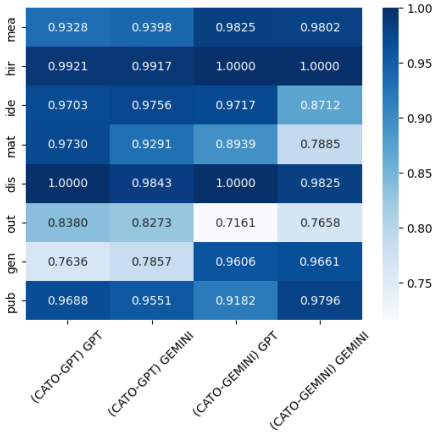
In this section, we report the result from our experiments. In our experiments, we used two factor lists: Cato and Credit and two large language models: GPT-4o and Gemini-2.0-Flash-Lite (denoted as GPT and Gemini, respectively). Hence, there are four generated case description sets, namely Cato-GPT, CATO-Gemini, Credit-GPT, and Credit-Gemini. We report the results as follows.



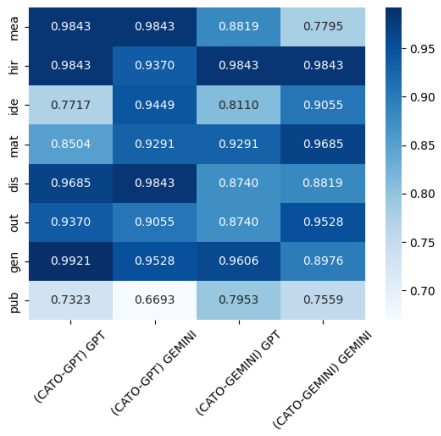
(a) (CATO) F1-score



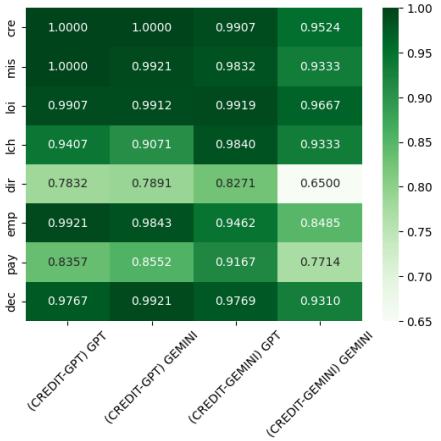
(b) (CREDIT) F1-score



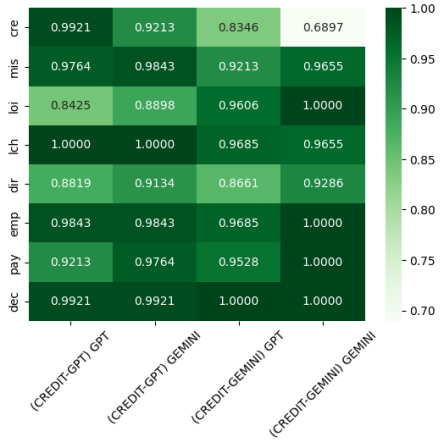
(c) (CATO) Precision



(d) (CATO) Recall



(e) (CREDIT) Precision



(f) (CREDIT) Recall

Fig. 2: Comparative performance on factor identification

4.1 Factor Identification

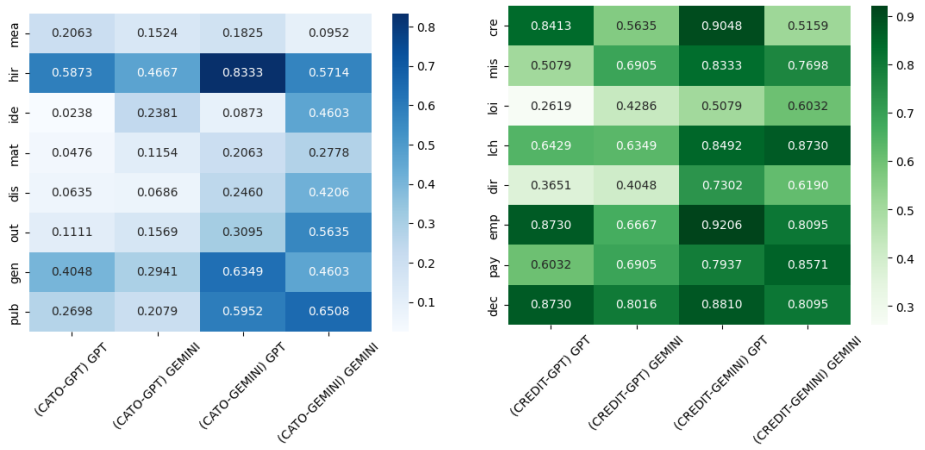
Fig. 2 shows comparative performance (precision, recall, and the F1-score) on the factor identification task using heatmaps. First, we consider which CATO factors contribute the highest and the lowest F1-scores and how those factors contribute precision and recall. Fig. 2a shows that $\mathbf{hir}^{\pi}$ contributes the highest F1-score in the experiments based on CATO-GPT and also the experiments based on CATO-GEMINI and using GEMINI as the base LLM. $\mathbf{dis}^{\delta}$ contributes the highest F1-score in the experiment based on CATO-GEMINI and using GPT as the base LLM. Fig. 2c and Fig. 2d show that both factors contribute higher precision and recall compared to others. Meanwhile, Fig. 2a shows that $\mathbf{pub}^{\delta}$ contributes the lowest F1-score in the experiments based on CATO-GPT and $\mathbf{out}^{\delta}$ contributes the lowest F1-score in the experiments based on CATO-GEMINI. Fig. 2c and Fig. 2d show $\mathbf{pub}^{\delta}$ contributes relatively low recall while $\mathbf{out}^{\delta}$ contributes relatively low precision.

Next, we analyze which CREDIT factors that contribute the highest and the lowest F1-score and how those factors contribute precision and recall. Fig. 2b shows that $\mathbf{emp}^{\delta}$ has the highest F1-score in the experiment based on CREDIT-GPT and using GPT as the base LLM; $\mathbf{dec}^{\delta}$ has the highest F1-score in the experiment based on CREDIT-GPT and using GEMINI as the base LLM and the experiment based on CREDIT-GEMINI and using GPT as the base LLM; and $\mathbf{loi}^{\pi}$ has the highest F1-score in the experiment based on CREDIT-GEMINI and using GEMINI as the base LLM. Fig. 2e and Fig. 2f show that those factors tend to have high recall. Meanwhile, $\mathbf{dir}^{\delta}$ contributes the lowest F1-score in the experiments based on CREDIT factors across every setting. Fig. 2e shows that $\mathbf{dir}^{\delta}$ contributes relatively low precision.

When comparing across the domains, Fig. 2c and Fig. 2e show that pro-plaintiff factors tend to contribute higher precision than pro-defendant factors. Fig. 2f shows that pro-defendant CREDIT factors tend to contribute higher recall than pro-plaintiff CREDIT factors. However, Fig. 2d does not clearly show that pro-defendant CATO factors tend to contribute higher recall than pro-plaintiff CATO factors. Additionally, when comparing between the case description sets generated by both LLMs, the case description sets generated by GPT tend to contribute higher and more consistent performance (precision, recall, and the F1-score) than the description sets generated by GEMINI.

4.2 Factor Discovery

Fig. 3 shows comparative performance (hit rate) on the factor discovery task using heatmaps. Fig. 3a shows that $\mathbf{hir}^{\pi}$ contributes the highest hit rate. It also shows that $\mathbf{ide}^{\pi}$ contributes the lowest hit rate in the experiments based on CATO factors using GPT as the base LLM; $\mathbf{dis}^{\delta}$ contributes the lowest hit rate in the experiment based on CATO-GPT and using GEMINI as the base LLM; and $\mathbf{mea}^{\pi}$ contributes the lowest hit rate in the experiment based on CATO-GEMINI and using GEMINI as the base LLM. Fig. 3b shows that $\mathbf{dec}^{\delta}$ contributes the highest hit rate in the experiment based on CREDIT-GPT and



(a) (CATO) Hit Rate (b) (CREDIT) Hit Rate

Fig. 3: Comparative performance on factor discovery

emp$^{\delta}$ contributes the highest hit rate in the experiment using GPT as the base LLM. It also shows that **loi**$^{\pi}$ contributes the lowest hit rate in the experiments based on CREDIT factors and using GPT as the base LLM; **dir**$^{\delta}$ contributes the lowest hit rate in the experiment based on CREDIT-GPT and using GEMINI as the base LLM; and **cre**$^{\pi}$ contributes the lowest hit rate in the experiment based on CREDIT-GEMINI and using GEMINI as the base LLM.

When comparing across the domains, Fig. 3 shows that pro-defendant factors tend to contribute higher hit rate than pro-plaintiff factors. Additionally, when comparing between the case description sets generated by both LLMs, the case description sets generated by GEMINI tend to contribute higher and consistent hit rate than the description sets generated by GPT.

5 Discussion

This paper examined how the inherent characteristics of legal factors systematically influence the performance of LLMs in two fundamental tasks of factor extraction: *factor identification* and *factor discovery*.

Our analysis of factor identification (Section 4.1) revealed that high-performing factors tend to be concrete or contract-based, such as those involving employment or explicit contractual terms. Their clarity and explicitness make them easier for LLMs to identify. In $\mathbb{F}_{cato}$, for instance, factors such as **hir**$^{\pi}$ (the defendant hired the plaintiff's key personnel with trade secret information) and **dis**$^{\delta}$ (the plaintiff disclosed the trade secret information during business negotiations) performed well. Similarly, in $\mathbb{F}_{credit}$, **emp**$^{\delta}$ (the applicant has a long and stable employment) and **dec**$^{\delta}$ (the applicant has significant assets declared) were high-performing.

In contrast, low-performing factors tend to be complex or context-dependent, often involving boundaries or degrees that require nuanced reasoning. These factors typically demand contextual interpretation or inferential judgment beyond surface-level cues. For example, in $\mathbb{F}_{cato}$, $\mathbf{pub}^{\delta}$ (the plaintiff disclosed information in public forum) and $\mathbf{out}^{\delta}$ (the plaintiff shared the trade secret information to people outside the organization) performed poorly. Likewise, in $\mathbb{F}_{credit}$, $\mathbf{dir}^{\delta}$ (the applicant has a low debt-to-income ratio) consistently exhibited the lowest performance across all settings.

These tendencies align with factor polarity. Because plaintiffs are typically the initiating party in litigation, pro-plaintiff factors tend to be better supported by concrete evidence, such as contracts or documents, leading to higher precision. By contrast, pro-defendant factors show more variation across domains. In $\mathbb{F}_{cato}$, pro-defendant factors are generally context-dependent, while in $\mathbb{F}_{credit}$, they tend to be more concrete and context-independent.

Our analysis of factor discovery (Section 4.2) highlighted the challenges LLMs face in defining new factors from case descriptions. In comparison to identification, discovery performance was relatively lower. Difficult-to-discover factors were either too subtle or highly overlapping with other factors in the set, making it challenging for the LLM to isolate and reconstruct the missing element. In $\mathbb{F}_{cato}$, examples include $\mathbf{ide}^{\pi}$ (the defendant's product is identical or nearly identical), $\mathbf{dis}^{\delta}$ (the plaintiff disclosed the trade secret information during business negotiations), and $\mathbf{mea}^{\pi}$ (the plaintiff implemented sufficient security measures to maintain the secrecy of the trade secret information) are hard to discover. In $\mathbb{F}_{credit}$, $\mathbf{loi}^{\pi}$ (the applicant has insufficient income), $\mathbf{dir}^{\delta}$ (applicant has a low debt-to-income ratio), and $\mathbf{cre}^{\pi}$ (the applicant has high number of recent credit inquiries) were particularly challenging to discover.

We also found that task performance varied with the length of the case description. As shown in Section 3.2, case descriptions generated by GPT tended to be longer than those produced by Gemini. Consequently, LLMs achieved better performance in factor identification using descriptions generated by GPT, likely because longer descriptions emphasize relevant details. Conversely, factor discovery performed better with shorter descriptions generated by Gemini, which may contain less textual noise and redundancy.

Interestingly, we observed that the well-established CATO factors often resulted in lower performance than the newly constructed Credit factors. One possible explanation is that LLMs already possess background knowledge about trade secret law. When generating case descriptions, they may implicitly add unlisted but related factors, which in turn affects both the identification and discovery tasks.

These findings reveal a limitation in our experimental setup: some errors may originate from the case description generation phase itself. Future research should therefore focus on extracting factors directly from real precedent cases to better capture authentic legal reasoning. Nonetheless, our results highlight important technical challenges for legal AI development. In particular, they suggest the

value of adapting extraction strategies to factor characteristics and applying legal case summarization techniques in factor extraction to mitigate errors.

6 Conclusion

This paper investigated errors arising from the use of LLMs to automate two fundamental tasks in factor extraction: *factor identification* and *factor discovery*. We conducted experiments across two legal domains – trade secret law and credit card applications – using two LLMs, GPT-4o and Gemini 2.0 Flash Lite, to generate case descriptions and perform these tasks. Our analysis reveals that high-performing factors tend to be concrete and established by contracts, whereas low-performing factors are typically complex and context-dependent. Additionally, we found that LLMs identify factors more accurately from longer case descriptions but discover new factors more effectively from shorter ones. Future research could extend this work by focusing on factor extraction from real precedent cases, adapting extraction methods to specific factor characteristics, and applying legal case summarization techniques in factor extraction.

Acknowledgements

This work was supported by the "R&D Hub Aimed at Ensuring Transparency and Reliability of Generative AI Models" project of the Ministry of Education, Culture, Sports, Science and Technology, the "Strategic Research Projects" grant from ROIS (Research Organization of Information and Systems), and JSPS KAKENHI Grant Numbers, JP22H00543, JP25H00522, JP25H01112, and JP25H01152. We appreciate all comments from the reviewers.

References

1. Aleven, V.: Teaching case-based argumentation through a model and examples. Ph.D. thesis, University of Pittsburgh (1997)
2. Aleven, V., Ashley, K.D.: Doing things with factors. In: Proceedings of the 5th international conference on artificial intelligence and law. pp. 31–41 (1995)
3. Angluin, D.: Queries and concept learning. Machine learning **2**(4), 319–342 (1988)
4. Ashley, K.D., Brüninghaus, S.: Automatically classifying case texts and predicting outcomes. Artificial Intelligence and Law **17**(2), 125–165 (2009)
5. Brüninghaus, S., Ashley, K.D.: Reasoning with textual cases. In: International Conference on Case-Based Reasoning. pp. 137–151. Springer (2005)
6. Fungwacharakorn, W., Zin, M.M., Nguyen, H.T., Kong, Y., Satoh, K.: Argumentative reasoning with language models on non-factorized case bases. In: Proceedings of the Second International Workshop on Next-Generation Language Models for Knowledge Representation and Reasoning (NeLaMKRR 2025) (2025)
7. Gray, M., Savelka, J., Oliver, W., Ashley, K.: Automatic identification and empirical analysis of legally relevant factors. In: Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law. pp. 101–110 (2023)

8. Gray, M., Savelka, J., Oliver, W., Ashley, K.: Using LLMs to discover legal factors. In: Legal Knowledge and Information Systems, pp. 60–71. IOS Press (2024)
9. Horty, J.F., Bench-Capon, T.J.: A factor-based definition of precedential constraint. Artificial intelligence and Law **20**(2), 181–214 (2012)
10. Rempell, S.: Factors. Buff. L. Rev. **70**, 1755 (2022)
11. Rissland, E.L., Ashley, K.D.: A case-based system for trade secrets law. In: Proceedings of the 1st international conference on Artificial intelligence and law. pp. 60–66. Association for Computing Machinery, New York, NY, USA (1987)
12. Wyner, A., Peters, W.: Towards annotating and extracting textual legal case factors. In: Proceedings of the 3rd Workshop on Semantic Processing of Legal Texts (SPLeT 2010). pp. 36–45 (2010)
13. Zheng, H., Grossi, D., Verheij, B.: Logical comparison of cases. In: International Workshop on AI Approaches to the Complexity of Legal Systems. pp. 125–140. Springer (2018)

From LegalRuleML to Defeasible Deontic Logic

Guido Governatori[1], Monica Palmirani[2], and Muhammad Asif[2]

[1] School of Engineering and Technology, Central Queensland University, Australia
[2] ALMA-AI, University of Bologna

Abstract. LegalRuleML is an application-agnostic standard for the representation of legal knowledge. Accordingly, once a set of legal provisions has been encoded in LegalRuleML, they must be translated into a target language before the rules can be used in practical applications. In this paper, we present a novel parser that enables the translation from LegalRuleML to a recent implementation of Defeasible Deontic Logic. The transformation process is illustrated with provisions from the Australian Spent Conviction scheme (Part VIIC of the Australian Crimes Act 1914).

Keywords: LegalRuleML, Defeasible Deontic Logic, Answer Set Programming

1 Introduction

A recent OECD white paper [13] reports that the adoption of Rules as Code (RaC) in government services would provide several benefits. RaC is the practice of expressing laws and regulations in a machine-readable format, enabling automated decision-making and service delivery. The benefits of RaC include improved efficiency, accuracy, and transparency in government services. However, the adoption of RaC also presents several challenges, including the need for standardisation, the complexity of legal language, and the challenges of legal automated decision-making. A further issue that hampers the adoption of RaC is that many languages and frameworks have been proposed for the execution of rules (see [2, 14, 17, 6] for an overview of existing RaC approaches). Often these languages are domain specific and adopt different semantics. Accordingly, there is no common understanding of how the consequences of rules should be computed. This means that the same provision can be encoded in different ways; consequently, the representations are not interoperable.

To address the issue of standardisation, the OASIS LegalRuleML Technical Committee has developed LegalRuleML [15, 1], an XML-based standard for the representation of legal norms. LegalRuleML provides a rich set of constructs to represent legal norms, including deontic concepts such as obligation, permission, and prohibition. However, LegalRuleML is application-agnostic, meaning that it does not provide a specific language for the execution of rules. Therefore, once a set of legal provisions has been encoded in LegalRuleML, they must be translated into a target language before the rules can be used in practical applications. Accordingly, translators/parsers are needed to transform the rules into target languages for execution.

The contribution of this paper is twofold. First, we illustrate how to represent legal norms in LegalRuleML using examples from the Australian Spent Conviction scheme (Part VIIC of the Australian Crimes Act 1914). Second, we present a novel parser that enables the translation from LegalRuleML to a recent implementation of Defeasible Deontic Logic (DDL) [6], a logic specifically designed for reasoning with legal norms. The DDL implementation is based on Answer Set Programming (ASP) [5, 4] and is available as an open-source tool[3].

The rest of the paper is structured as follows. In Section 2, we provide a brief overview of LegalRuleML. In Section 3, we provide a brief overview of Defeasible Deontic Logic. In Section 4, we present our parser from LegalRuleML to DDL. In Section 5, we illustrate the use of LegalRuleML and the parser with examples from the Australian Spent Conviction scheme. Finally, in Section 6, we conclude the paper and discuss future work.

2 LegalRuleML

LegalRuleML [15, 1] (LRML) is an OASIS standard for the representation of legal provisions. The standard is based on XML and provides a rich set of constructs to represent legal norms. In LegalRuleML, a legal provision is represented as a rule with an IF ... THEN structure, where rules can be classified as constitutive or prescriptive. Also, it is possible to specify the strength of the rule (strict, defeasible, and defeater). A strict rule is a rule that is always applicable when the conditions in the IF part are satisfied. A defeasible rule is a rule that can be defeated by other rules, while a defeater is a rule that can be used to prevent the application of other rules, but it does not support the derivation of conclusions. Accordingly, a (defeasible) prescriptive rule is represented as:

```
<lrml:PrescriptiveStatement key="ps_r">
  <ruleml:Rule key="r">
    <lrml:hasStrength>
      <lrml:DefeasibleStrength iri="lov:defeasible"/>
    </lrml:hasStrength>
    <ruleml:if>
      <!-- logical formula -->
    </ruleml:if>
    <ruleml:then>
      <!-- deontic formula(s) in the scope of obligation -->
    </ruleml:then>
  </ruleml:Rule>
</lrml:PrescriptiveStatement>
```

The structure of a constitutive rules is similar. The main differences are that the rule is enclosed in a `<lrml:ConstitutiveStatement>` element, and the deontic formula in the `<ruleml:then>` element is replaced by a formula without deontic operators.

Permissive rules are represented similarly, but the difference from prescriptive rules is that the deontic formulas in the `<ruleml:then>` element are in the scope of the permission deontic operator.

[3] `https://github.com/gvdgdo/Defeasible-Deontic-Logic`

Formulas in LegalRuleML are built using the constructs provided by RuleML [3], extended with deontic operators. The basic building block is the atomic formula, represented as:

```
<ruleml:Atom>
  <ruleml:Rel iri="voc:predicate"/>
  <ruleml:Ind iri=":constant"/>
  <ruleml:Var keyref="#variable"/>
</ruleml:Atom>
```

where `voc:predicate` is the predicate of the atom (the definition and meaning of the predicate is given in an external vocabulary file referred to by `voc`).[4] The predicate can have an arity (including 0, in which case it is a proposition), and the arguments can be constants or variables. The element `<ruleml:Ind>` is used to represent a constant, while the element `<ruleml:Var>` is used to represent a variable. More complex formulas can be built using the standard logical connectives. Also, LegalRuleML admits the use of functions. We use `iri` to bind variables and constants. More specifically, when `iri` starts with `:` we introduce a new variable/constant, while when it starts with `#` we refer to an already defined variable/constant.

A *deontic formula* is a formula in the scope of a deontic operator. The deontic operators we consider are obligation, permission, and prohibition (obligation of the negation). Thus, for example, the deontic formula "it is obligatory to disclose the conviction" can be represented as:

```
<lrml:Obligation>
  <ruleml:Atom>
    <ruleml:Rel iri="voc:disclose"/>
    <ruleml:Var iri=":person"/>
    <ruleml:Var iri=":conviction"/>
  </ruleml:Atom>
</lrml:Obligation>
```

A special type of deontic formula is the *SuborderList* which is used to represent a list of alternative obligations, where the elements in the list are ordered. For example, the suborder list "it is obligatory to pay a fine; if it is not paid, to do community service; and if that is not done, to undergo imprisonment" can be represented as:

```
<lrml:SuborderList>
  <lrml:Obligation>
    <ruleml:Atom>
      <ruleml:Rel iri="voc:pay_fine"/>
    </ruleml:Atom>
  </lrml:Obligation>
  <lrml:Obligation>
    <ruleml:Atom>
      <ruleml:Rel iri="voc:do_community_service"/>
    </ruleml:Atom>
  </lrml:Obligation>
```

[4] In this paper we assume the existence of two vocabularies, `voc` defining the terms specific to the domain application, and `loc` for the definition of logical terms and functions.

```
  <lrml:Obligation>
    <ruleml:Atom>
      <ruleml:Rel iri="voc:do_imprisonment"/>
    </ruleml:Atom>
  </lrml:Obligation>
</lrml:SuborderList>
```

The final type of statement we consider is the specification of the relative strength of two rules. This is captured by `lrml:Override` element. For example, the statement "rule r_1 is stronger than rule r_2", meaning that in case the rules are both applicable, rule r_1 takes precedence over rule r_2, is represented as

```
<lrml:OverrideStatement>
  <lrml:Override over="#r_1" under="#r_2"/>
</lrml:OverrideStatement>
```

Furthermore, LegalRuleML offers facilities to connect the rules with their sources:

```
<lrml:Association>
  <lrml:appliesSource keyref="ex:Article 4"/>
  <lrml:toTarget keyref="#r4"/>
</lrml:Association>
```

This statement specifies that rule r is the encoding of Article 4 of the document specified by the reference "ex". However, we do not use this feature in this paper.

The last element is that ontologies or vocabularies can be integrated with the specification of the rules. In the example above, the vocabulary is used to specify the relation between the rules and the concepts in the domain. Thus, `voc:publish` and `voc:commission` are terms in the vocabulary. For our purpose, the elements in the vocabulary have the following form:

```
<Term key="evaluate">
  <Atom>evaluate<Atom>
  <Description>evaluate the product</Description>
</Term>
```

LegalRuleML offers many more features for representing the meta-data and for linking the rules with their legal source. However, since these are not relevant for the purpose of the current paper, they will be ignored. For a detailed presentation of LegalRuleML and its features see [1].

3 Defeasible Deontic Logic (DDL)

Defeasible Deontic Logic (DDL) [9, 11] is an efficient rule-based logic specifically designed for reasoning with legal norms. The presentation of DDL in this paper is based on its recent implementation in Answer Set Programming (ASP) [6]. While ASP and DDL are two computational approaches to non-monotonic reasoning, they have different semantics and thus they are not directly comparable. In addition, some recent works [7, 10] point out that Stable Semantics (the semantics for ASP) is not suitable for normative reasoning, while the semantics of DDL

does not suffer from the problems affecting ASP. However, ASP can be used as a meta-program to correctly encode the reasoning mechanism of DDL. In addition, the ASP implementation allows representing rules with variables and constants, and the computation is supported by the efficient ASP grounding mechanism.

The language of DDL is defined on a set of atoms, where an atom is declared with the expressions

```
atom(atomName).
atom(atomName(<Variables>)) :-
  <typeVariable1>(Variable1), ..., <typeVariableN>(VariableN).
```

depending on whether we define a proposition or a literal based on a predicate (of any arity). In the case of a predicate, the variables occurring in the predicate must be grounded (and we introduce a type or domain for each variable).

Rules are built from literals, where a literal is either a plain literal (an atom or a negated atom—an atom in the scope of a negation, i.e., `non`) or a deontic literal, where a deontic literal is a literal in the scope of a deontic operator, specifically, obligation (`obl`) and permission (`perm`). A prohibition is represented as the obligation of the negation of the literal.

DDL offers three types of rules: constitutive rules, prescriptive rules and permissive rules. A constitutive rule is used to define new concepts in terms of existing ones. A constitutive rule has the form

```
constitutiveRule(<label>, <head>).
body(<label>, (<body1>;...;<bodyN>)).
```

where `<label>` is a unique identifier for the rule, `<head>` is a literal (atom or negated atom), and `<body1>;...;<bodyN>` is a conjunction of literals. The intuitive meaning of the rule is that if one of the elements in the body holds, then the head holds. In case the head or any element of the body contains a variable, the rules must be grounded, and the variables must be declared as explained above for atoms. Thus we have the following pseudo-code:

```
constitutiveRule(<label>(<Variables>), <head>(<Variables>)) :-
  <typeVariable1>(<Variable1>), ..., <typeVariableN>(<VariableN>).
body(<label>(<Variables>),
     (<body1>(<Variables>);...;<bodyN>(<Variables>))) :-
  <typeVariable1>(<Variable1>), ..., <typeVariableN>(<VariableN>).
```

A prescriptive rule is used to represent an obligation, and has the form

```
prescriptiveRule(<label>, <head>).
body(<label>, (<body1>;...;<bodyN>)).
```

The intuitive meaning of the rule is that if the elements of the body hold, then it is obligatory that the head holds. The head and the body can contain variables, and the rules must be grounded as explained for constitutive rules.

Moreover, it is possible to specify compensatory obligations, i.e., obligations that come into force when another obligation is violated. This is done by means of the `compensate` rule, which has the form

```
compensate(<label>, <head1>, <head2>, <level>).
```

where <label> is the label of a prescriptive rule, <head1> and <head2> are two literals, and <level> is a positive integer. The intuitive meaning of the rule is that if the obligation <head1> is violated, then it is obligatory to fulfill the obligation <head2>. The level is used to specify a hierarchy of compensatory obligations.

Finally, a permissive rule is used to represent a permission, and has the form

```
permissiveRule(<label>, <head>).
body(<label>, (<body1>;...;<bodyN>)).
```

The intuitive meaning of the rule is that if the elements of the body hold, then it is permitted that the head holds. The head and the body can contain variables, and the rules must be grounded as explained for constitutive rules.

4 An LRML to DDL Parser

For the purpose of this paper, and the parser from LegalRuleML to DDL, we restrict the parser to a subset of LegalRuleML. Specifically, we restrict the formulas in the <ruleml:if> element to be conjunctions of (deontic) literals (atoms and negated atoms) or conjunctions of (deontic) literals and disjunctions (where the disjuncts are (deontic) literals). Also, we restrict the formulas in the <ruleml:then> element to be a single deontic literal or a conjunction of deontic literals or a suborder list for prescriptive rules; a single deontic literal or a conjunction of deontic literals for permissive formulas; and a single literal or a conjunction of literals for constitutive rules. While more complex formulas can be represented in LegalRuleML, the restriction is necessary to guarantee that the output of the parser is a well-defined DDL theory, and from practical experience it is sufficient to represent most legal norms. More complex formulas can be represented in LegalRuleML, but they would require a more complex translation to DDL, which is left for future work.

Unfortunately, space reasons prevent us from providing a formal definition of the parser. However, we give the main ideas and outline the translation, and we provide examples of the translations in Section 5. The parser checks the type of the rule and analyses each statement. If the statement contains a constitutive rule, the parser creates a constitutive rule in DDL. If the statement contains a prescriptive rule, the parser checks the then part to see if it contains obligations, permissions, or suborder lists. If the then part is a single element, then it creates a single rule (a prescriptive rule in the case of an obligation or suborder list), and a permissive rule for permissions.

Moreover, the parser checks if ruleml:if and ruleml:then parts contain variables. If they do, the parser collects them and adds them as arguments of the predicates in the label and the head of the rule. In ASP variables must be grounded, so for each variable, the parser creates a domain predicate. The convention we use is to use the suffix "ID" for variables representing objects, and "Dates" for variables representing dates. For example, if the variable is #person, the parser creates the domain predicate personID/1.

The output of this part is

```
<type>Rule(<label>(<variables>), <head>(<variables>)) :-
  <typeVariable1>(<variable1>), ... <typeVariableN>(<variableN>).
```

If the `ruleml:then` part contains a conjunction of elements, the parser creates a single rule if the elements are all obligations (prescriptive rule), or all permissions (permissive rule), or all literals (constitutive rule). If the elements are mixed, the parser creates multiple rules, one for each element (as illustrated above). If the conjuncts are of the same type, the parser creates a rule where the conjunctions are pooled in the second element of `<type>Rule`

```
<type>Rule(<label>(<variables>),
           (<head1>(<variables>); ... <headN>(<variables>))) :-
  <typeVariable1>(<variable1>), ... <typeVariableN>(<variableN>).
```

This is equivalent to the creation of multiple rules.

If the `ruleml:then` part contains a suborder list, the parser creates a prescriptive rule where the head of the rule is the first element of the list, and then creates n `compensate` rules, where n is the number of elements in the list minus one. The compensate rules have the following form:

```
compensate(<label>(<variables>),
           <head1>(<variables>), <head2>(<variables), 1) :-
  <typeVariable1>(<variable1>), ... , <typeVariableN>(<variableN>).
compensate(<label>(<variables>),
           <head2>(<variables>), <head3>(<variables), 2) :-
  <typeVariable1>(<variable1>), ... , <typeVariableN>(<variableN>).
...
compensate(<label>(<variables>),
           <headN-1>(<variables>), <headN>(<variables), N-1) :-
  <typeVariable1>(<variable1>), ... , <typeVariableN>(<variableN>).
```

If the `ruleml:if` part only contains a single literal or a conjunction of literals the parser creates a single body element (with the same convention for grounding as above):

```
body(<label>(<variables>),
    (<body1>(<variables>); ... ; <bodyN>(<variables>))) :-
  <typeVariable1>(<variable1>), ... , <typeVariableN>(<variableN>).
```

When it contains a conjunction of literals and disjunctions, for each disjunction we introduce a new literal for each literal in the disjunction, and we create a new constitutive rule for the new literal. The body of the original rule contains the conjunction of the conjuncts in the rule and the new literals.

```
<type>Rule(<label>(<variables>), <head>(<variables>)) :-
  <typeVariable1>(<variable1>), ... , <typeVariableN>(<variableN>).
body(<label>(<variables>),
    (<body1>(<variables>); ... ; <bodyM>(<variables>);
     <disjunction1>(<variables>); ... ; <disjunctionK>(<variables>))
         ) :-
  <typeVariable1>(<variable1>), ... , <typeVariableN>(<variableN>).

constitutiveRule(<newLabel1>(<variables>),
                 <disjunction1>(<variables>)) :-
  <typeVariable1>(<variable1>), ... , <typeVariableN>(<variableN>).
```

```
body(<newLabel1>(<variables>), <disjunct1_1>(<variables>)) :-
  <typeVariable1>(<variable1>), ... , <typeVariableN>(<variableN>).
...
constitutiveRule(<newLabelK>(<variables>),
                 <disjunction1>(<variables>)) :-
  <typeVariable1>(<variable1>), ... , <typeVariableN>(<variableN>).
body(<newLabelK>(<variables>), <disjunct1_K>(<variables>)) :-
  <typeVariable1>(<variable1>), ... , <typeVariableN>(<variableN>).
```

In the next section we illustrate the use of the parser.

5 A Case Study: Spent Conviction

A "spent conviction" is a conviction that becomes hidden from public view after a set period of time but, depending on certain factors, it still remains accessible for specific (public) purposes by specific interested parties. These schemes are mainly focused on convictions for less serious crimes and generally do not extend to convictions for violent crimes and sexual offences. The set period of time is also extended when the person has re-offended during the set period. In this paper we concentrate on examples from the Australian Spent Conviction scheme. In Australia, the Spent Conviction scheme is regulated by Part VIIC of the Crimes Act 1914 (Cth). We will use the example to illustrate how to model them in LegalRuleML, and then how to translate them into DDL using our parser.

The encoding in the current paper is inspired by [8], which presents an encoding of the Australian Spent Conviction scheme in DDL. However, the current encoding is different since it does not depend on specific characteristics of DDL, and it is not propositional.

We start our analysis with the definition of *waiting period* as given in the Crimes Act 1914 (Cth):

waiting period, in relation to an offence, means:

(a) if the person convicted of the offence was dealt with as a minor in relation to the conviction-the period of 5 years beginning on the day on which the person was convicted of the offence; or

(b) in any other case the period of 10 years beginning on the day on which the person was convicted of the offence.

The above provision can be represented in LegalRuleML by two constitutive rules (for space reasons, we present only the first one, the second one is similar).

```
<lrml:ConstitutiveStatement key="s_waiting_period_1">
  <ruleml:Rule key="wp_1">
    <lrml:hasStrength>
      <lrml:DefeasibleStrength iri="lov:defeasible"/>
    </lrml:hasStrength>
    <ruleml:if>
    <ruleml:And>
      <ruleml:Atom>
        <ruleml:Rel iri="voc:minorAtConviction"/>
        <ruleml:Var iri=":person"/>
        <ruleml:Var iri=":conviction"/>
        <ruleml:Var iri=":convictionDate"/>
```

```
      </ruleml:Atom>
      <ruleml:Atom>
        <ruleml:Rel iri="voc:fiveYearsElapsed"/>
        <ruleml:Var keyref="#person"/>
        <ruleml:Var keyref="#conviction"/>
        <ruleml:Var keyref="#convictionDate"/>
      </ruleml:Atom>
    </ruleml:And>
    </ruleml:if>
    <ruleml:then>
      <ruleml:Atom>
        <ruleml:Rel iri="voc:waitingPeriodEnded"/>
        <ruleml:Var keyref="#conviction"/>
        <ruleml:Var keyref="#person"/>
      </ruleml:Atom>
    </ruleml:then>
  </ruleml:Rule>
</lrml:ConstitutiveStatement>
```

The rule is translated into the following DDL rule:

```
constitutiveRule(wp_1(Conviction, Person),
                 waitingPeriodEnded(Conviction, Person)) :-
  convictionID(Conviction), personID(Person).
body(wp_1(Conviction, Person),
     (minor(Person, Conviction, ConvictionDate);
      fiveYearElapsed(Person, Conviction, ConvictionDate)) :-
  personID(Person), convictionID(Conviction),
  convictionDates(ConvictionDate).
```

Here we have a constitutive statement where the if part contains a simple conjunction of literals, and the then part a single literal. Accordingly, we create a single constitutive rule in DDL. In the declaration of the rule we have the head of the rule, and the label of the rule. For both elements, we identified the variables appearing in them, and we added them as arguments of the predicates. In the body of the rule, we have the conjunction of literals, and we can use a single body element. Again, we have to ground the variables appearing in the body, so we first collect them. We use the convention to use the word "Date" as suffix for variables representing dates (and the parser creates the suitable domain predicate, i.e., `convictionDates/1`).

For another example of a constitutive rule (where we have a disjunction) we can consider the rules encoding Section 85ZM9(2)(b):

(2) For the purposes of this Part, a person's conviction of an offence is spent if:

(b) the person was not sentenced to imprisonment for the offence, or was not sentenced to imprisonment for the offence for more than 30 months, and the waiting period for the offence has ended.

```
<lrml:ConstitutiveStatement key="s_85ZM_2b">
  <ruleml:Rule key="r_85ZM_2b">
    <lrml:hasStrength>
      <lrml:DefeasibleStrength iri="lov:defeasible"/>
    </lrml:hasStrength>
    <ruleml:if>
      <ruleml:And>
```

```
        <ruleml:Or>
          <ruleml:Neg>
            <ruleml:Atom>
              <ruleml:Rel iri="voc:SentencedToImprisonment"/>
              <ruleml:Var iri=":person"/>
              <ruleml:Var iri=":conviction"/>
            </ruleml:Atom>
          </ruleml:Neg>
        <ruleml:Atom>
          <ruleml:Rel iri="vocImprisonmentLess30Months"/>
          <ruleml:Var keyref="#person"/>
          <ruleml:Var keyref="#conviction"/>
        </ruleml:Or>
        <ruleml:Atom>
          <ruleml:Rel iri="voc:waitingPeriodEnded"/>
          <ruleml:Var keyref="#conviction"/>
          <ruleml:Var keyref="#person"/>
        </ruleml:Atom>
      </ruleml:And>
    </ruleml:if>
    <ruleml:then>
      <ruleml:Atom>
        <ruleml:Rel iri="voc:convictionIsSpent"/>
        <ruleml:Var keyref="#conviction"/>
        <ruleml:Var keyref="#person"/>
      </ruleml:Atom>
    </ruleml:then>
  </ruleml:Rule>
```

The rule is translated into the following DDL code:

```
constitutiveRule(r_85ZM_2b(Conviction, Person),
                 convictionIsSpent(Conviction, Person)) :-
  convictionID(Conviction), personID(Person).
body(r_85ZM_2b(Conviction, Person), (
  (disjunction_1(Person, Conviction);
   waitingPeriodEnded(Conviction, Person))) :-
  personID(Person), convictionID(Conviction).

constitutiveRule(r_85ZM_aux1(Person, Conviction),
                 disjunction_1(Person, Conviction)) :-
  convictionID(Conviction), personID(Person).
body(r_85ZM_aux1(Person, Conviction),
  non(sentencedToImprisonment(Person, Conviction))) :-
  personID(Person), convictionID(Conviction).

constitutiveRule(r_85ZM_aux2(Person, Conviction),
                 disjunction_1(Person, Conviction)) :-
  convictionID(Conviction), personID(Person).
body(r_85ZM_aux1(Person, Conviction),
  imprisonmentLess30Months(Person, Conviction) :-
  personID(Person), convictionID(Conviction)).
```

The first step is to create a constitutive rule for the provision. Since the body of the rule contains a disjunction, we create two auxiliary rules to capture the disjuncts.

The next example is a prescriptive rule, for which we consider Section 85ZP(1)(a):

(1) Subject to Division 6, but despite any other Commonwealth law or any State law or Territory law, where, under section 85ZR, a person is, in particular circumstances or for a particular purpose, to be taken never to have been convicted of an offence:

(a) the person is not required, in those circumstances or for that purpose, to disclose the fact that the person was charged with, or convicted of, the offence;

The application of Section 85ZP depends on the notion of pardon or wrongful conviction, which are defined in Section 85ZR. The meaning of Section 85ZP(1) is that if a person has been pardoned for a conviction, or has been wrongly convicted, then the person is permitted to not disclose the conviction or the charge. The provision can be represented in LegalRuleML as follows:

```
<lrml:PrescriptiveStatement key="ps_85ZP_1a">
  <ruleml:Rule key="r_85ZP_1a">
    <lrml:hasStrength>
      <lrml:DefeasibleStrength iri="lov:defeasible"/>
    </lrml:hasStrength>
    <ruleml:if>
    <ruleml:Or>
      <ruleml:Atom>
        <ruleml:Rel iri="voc:pardon"/>
        <ruleml:Var iri=":conviction"/>
        <ruleml:Var iri=":person"/>
      </ruleml:Atom>
      <ruleml:Atom>
        <ruleml:Rel iri="voc:wronglyConvicted"/>
        <ruleml:Var keyref="#conviction"/>
        <ruleml:Var keyref="#person"/>
    </ruleml:Or>
    </ruleml:if>
    <ruleml:then>
    <ruleml:And>
      <lrml:Permission>
        <ruleml:Neg>
          <ruleml:Atom>
            <ruleml:Rel iri="voc:discloseConviction"/>
            <ruleml:Var keyref="#person"/>
            <ruleml:Var keyref="#conviction"/>
          </ruleml:Atom>
        </ruleml:Neg>
      </lrml:Permission>
      <lrml:Permission>
        <ruleml:Neg>
          <ruleml:Atom>
            <ruleml:Rel iri="voc:discloseCharge"/>
            <ruleml:Var keyref="#person"/>
            <ruleml:Var keyref="#conviction"/>
          </ruleml:Atom>
        </ruleml:Neg>
      </lrml:Permission>
    </ruleml:And>
    </ruleml:then>
  </ruleml:Rule>
```

The rule is translated into the following DDL code:

```
permissiveRule(r_85ZP_1a(Conviction, Person),
                 (non(discloseConviction(Person, Conviction));
```

```
                  non(discloseCharge(Person, Conviction))) :-
  convictionID(Conviction), personID(Person).
body(r_85ZP_1a(Conviction, Person),
  pardonOrWronglyConvicted(Person, Conviction)) :-
  convictionID(Conviction), personID(Person).

constitutiveRule(r_85ZP_aux1(Person, Conviction),
                 pardonOrWronglyConvicted(Person, Conviction)) :-
  convictionID(Conviction), personID(Person).
body(r_85ZP_aux1(Person, Conviction),
  pardon(Conviction, Person)) :-
  convictionID(Conviction), personID(Person).

constitutiveRule(r_85ZP_aux2(Person, Conviction),
                 pardonOrWronglyConvicted(Person, Conviction)) :-
  convictionID(Conviction), personID(Person).
body(r_85ZP_aux2(Person, Conviction),
  wronglyConvicted(Conviction, Person)) :-
  convictionID(Conviction), personID(Person).
```

Here we have a prescriptive rule where the if part contains a disjunction of literals, and the then part a conjunction of permissions. Accordingly, we create a single permissive rule in DDL, where the head of the rule contains the conjunction of the permissions. Since the body contains a disjunction, we create two auxiliary constitutive rules to capture the disjuncts.

Once a set of rules has been translated into DDL, we can use the DDL implementation to reason with the rules. For example, we can check if a certain obligation is in force, or if a certain action is permitted. To this end we need to provide the facts describing a specific case. For example, we can consider the following facts:

```
fact(dateOfBirth(john,2000-01-01)).
fact(dateOfConviction(shoplifting, john, 2017-10-10)).
fact(non(SentencedToImprisonment(john, shoplifting)))
fact(currentDate(2024-11-18)).
```

The above facts state that John was born on January 1, 2000, he was convicted for shoplifting on October 10, 2017, he was a minor at the time of the conviction, and was not sentenced to imprisonment. The date of the case is November 18, 2024. Based on the above facts, we can check if John is obliged to disclose the conviction. The output of executing the DDL program with the above facts is:

```
defeasible(spentConviction(shoplifting, john))
defeasible(minorAtConviction(john, shoplifting, 2017-10-10))
defeasible(fiveYearElapsed(john, shoplifting, 2017-10-10))
defeasible(waitingPeriodEnded(shoplifting, john))
permitted(non(discloseConviction(john, shoplifting)))
permitted(non(discloseCharge(john, shoplifting)))
```

This means that the conviction is spent, and John is permitted to not disclose the conviction or the charge. The conviction is spent since the waiting period has ended (5 years from the date of conviction since John was a minor at the time of conviction).

6 Related Work and Summary

While the LegalRuleML standard has been proposed some years ago, there are only a few works on the implementation of parsers from LegalRuleML to specific logics. The lack of parsers limits the use of LegalRuleML as an interchange format for legal norms. Accordingly, LegalRuleML encodings must be manually translated into the target logic, or the encoding is created for a specific logic (see, for example, [16]), preventing interoperability. A few parsers have been proposed. [12] presented a parser from LegalRuleML to DDL. However, the parser is propositional, while our parser supports variables and constants. [18] shows how to translate (a fragment) of LegalRuleML to TPTP formalism for automated theorem proving.

In this paper we presented a parser from LegalRuleML to Defeasible Deontic Logic, specifically to an ASP implementation of the logic. We have illustrated its use with a case study based on the Australian Spent Conviction scheme. The methodology presented in the paper can be used to represent and reason with legal norms encoded in LegalRuleML. The current version of the parser has some limitations on the format (conjunctions or conjunctions of disjunctions). However, there are no technical difficulties in extending to full expressivity, though from a practical point of view the current version is sufficient to represent most legal norms (while possible, multiple nested conjunctions and disjunctions are not particularly common in legal texts). Future work includes the extension of the parser to cover a larger fragment of LegalRuleML, and a proper empirical evaluation of the efficiency of the translations.

Acknowledgements

This project is conducted with the support of the European Commission funds within ERC HyperModeLex. Grant agreement ID: 101055185.

References

1. Athan, T., Governatori, G., Paschke, A., Palmirani, M., Wyner, A.: LegalRuleML: Design principles and foundations. In: Faber, W., Paschke, A. (eds.) Reasoning Web. Web Logic Rules, pp. 151–188. No. 9203 in LNCS, Springer (2015). https://doi.org/10.1007/978-3-319-21768-0˙6
2. Batsakis, S., Baryannis, G., Governatori, G., Ilias, T., Antoniou, G.: Legal representation and reasoning in practice: A critical comparison. In: Palmirani, M. (ed.) Legal Knowledge and Information Systems. pp. 31–40. IOS Press (2018). https://doi.org/10.3233/978-1-61499-935-5-31
3. Boley, H., Tabet, S., Wagner, G.: Design rationale of ruleml: A markup language for semantic web rules. In: Proceedings of the First International Semantic Web Conference (ISWC 2002). Sardinia, Italy (2002)
4. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Communications of the ACM **54**(12), 92–103 (2011). https://doi.org/10.1145/2043174.2043195

5. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2012). https://doi.org/10.1007/978-3-031-01561-8
6. Governatori, G.: An ASP implementation of defeasible deontic logic. Künstliche Intelligenz **38**, 79–88 (2024). https://doi.org/10.1007/s13218-024-00854-9
7. Governatori, G.: Weak permission is not well-founded, grounded and stable (2024), https://arxiv.org/abs/2411.10624
8. Governatori, G., Casanovas, P., de Koker, L.: On the formal representation of the australian spent conviction scheme. In: Gutiérrez Basulto, V., Kliegr, T., Soylu, A., Giese, M., Roman, D. (eds.) Rules and Reasoning. LNCS, vol. 12173, pp. 177–185. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-57977-7˙14
9. Governatori, G., Olivieri, F., Rotolo, A., Scannapieco, S.: Computing strong and weak permissions in defeasible logic. Journal of Philosophical Logic **42**(6), 799–829 (2013). https://doi.org/10.1007/s10992-013-9295-1
10. Governatori, G., Rotolo, A.: Deontic argumentation. In: Proceedings of DEON 2025 (2025). https://doi.org/10.2139/ssrn.5779102
11. Governatori, G., Rotolo, A., Sartor, G.: Logic and the law: Philosophical foundations, deontics, and defeasible reasoning. In: Gabbay, D.M., Horty, J., Parent, X., van der Meyden, R., van der Torre, L. (eds.) Handbook of Deontic Logic and Normative Reasoning, vol. 2, pp. 655–760. College Publications, London (2021)
12. Lam, H., Hashmi, M.: Enabling reasoning with legalruleml. Theory Practice of Logic Programming **19**(1), 1–26 (2019). https://doi.org/10.1017/S1471068418000339
13. Mohun, J., Roberts, A.: Cracking the code: Rulemaking for humans and machines. OECD working papers on public governance, OECD, Paris, France (2020). https://doi.org/10.1787/3afe6ba5-en
14. Morris, J.: Spreadsheets for legal reasoning: the continued promise of declarative logic programming in law. Master of laws thesis, University of Alberta, Alberta (2020)
15. Palmirani, M., Governatori, G., Athan, T., Boley, H., Paschke, A., Wyner, A.: LegalRuleML core specification version 1.0. OASIS Committee Specification 2, OASIS (2020), https://docs.oasis-open.org/legalruleml/legalruleml-core-spec/v1.0/cs02/legalruleml-core-spec-v1.0-cs02.html
16. Robaldo, L., Bartolini, C., Palmirani, M., Rossi, A., Martoni, M., Lenzini, G.: Formalizing GDPR provisions in reified I/O logic: The DAPRECO knowledge base. Journal of Logic Language and Information **29**(4), 401–449 (2020). https://doi.org/10.1007/S10849-019-09309-Z
17. Robaldo, L., Batsakis, S., Calegari, R., Calimeri, F., Fujita, M., Governatori, G., Morelli, M.C., Pacenza, F., Pisano, G., Satoh, K., Tachmazidis, I., Zangari, J.: Compliance checking on first-order knowledge with conflicting and compensatory norms. a comparison among currently available technologies. Artificial Intelligence and Law **32**(2), 505–555 (2024). https://doi.org/10.1007/s10506-023-09360-z
18. Steen, A., Fuenmayor, D.: Bridging between legalruleml and TPTP for automated normative reasoning. In: Governatori, G., Turhan, A. (eds.) Rules and Reasoning - 6th International Joint Conference on Rules and Reasoning, RuleML+RR 2022. Lecture Notes in Computer Science, vol. 13752, pp. 244–260. Springer (2022). https://doi.org/10.1007/978-3-031-21541-4“˙16

Catch the Platypus! Negated Conditionals as a Challenge for Machine Translation from Natural Language into Logical Formalisms using Large Language Models

Bianca Steffes[1][0009−0001−9784−8942] and Diogo Sasdelli[2][0000−0002−6504−9812]

[1] Saarland Informatics Campus, Chair of Legal Informatics, Saarland University, Campus A5.4, 66123 Saarbrücken, Germany
bianca.steffes@uni-saarland.de

[2] Department for E-Governance, University for Continuing Education Krems, Dr.-Karl-Dorrekt-Straße 30, 3500 Krems an der Donau, Austria
diogo.sasdelli@donau-uni.ac.at

Abstract. One of the most promising applications of large language models in the legal domain concerns the automated conversion of natural language legal texts into logical formalisms, i.e., automated formalisation. Major challenges to these approaches emerge from the semantic fuzziness of natural language, which leads to sentences that are particularly difficult to formalise – we call these sentences "platypus sentences". For example, the natural negation of a sentence in natural language may have different, context-dependent meanings, which often do not correspond to the logical negation of a respective formalisation of said sentence. In other words, natural negations and formalised negations often diverge from one another. This problem is further intensified when natural language conditionals (i.e., negated "if... then..." sentences) are negated. The paper at hand investigates how current large language models (GPT-5, Llama and LogicLinguist) deal with automated formalisation of negated conditionals. Our results indicate that these systems still cannot reliably deliver correct formalisations, although results can be enhanced, e.g., by prompt engineering.

Keywords: Machine Translation · Formalisation · Conditionals · Material Implication · Natural Language · Logic · Negation · Legal AI · Large Language Models (LLMs).

1 Introduction

Many applications in the wide area of LegalTech – in particular those related to automated compliance – require an adequate formal representation of legal norms, mostly by means of some logical formalism like deontic logic, predicate logic or directly through programming languages (cf., e.g., [7], [10], [18]). For example, to comply with traffic rules – and to be able to reliably verify this compliance –, an autonomous vehicle should include some compliance checking

mechanism containing a formal representation of the respective traffic rules. However, formalising legal norms can be a rather laborious task, involving many steps and requiring cooperation among specialists of various disciplines [27]. Hence, legal tech in general would enormously benefit from approaches that could at least partially automate the process of formalising legal norms.

Large language models (LLMs) offer a convenient method for translating natural language texts into logical formalisms, i.e., a method for automated formalisation. With an adequate prompt, entire arguments in natural language can be quickly translated into a logical formalism. However, depending on the specific scenario, the quality of these translations may vary substantially. For example, translations into more complicated logical formalisms (e.g., systems of reified modal logic, i.e., systems combining modalities with quantification theory) tend to be more prone to errors than translations into simpler formalisms such as propositional logic. A further, at least equally challenging problem lies in the complexity of the input language, i.e., natural language.

The paper at hand introduces the notion of "platypus sentences" – i.e., sentences which, due to the intrinsic ambiguity and context-dependence of natural language, might lead to errors when formalising them, especially when automated methods are employed – and investigates how LLMs perform when given the task of formalising dialogues containing a specific kind of such sentences, namely negated conditionals. Section 2 first briefly discusses related work on logical formalisms for legal norms, on methodologies for formalising legal norms, and on machine translation from natural language into logical formalisms, in particular when LLMs are employed and especially in the context of legal texts. Section 3 then presents the employed methodology, including a brief discussion of platypus sentences and a description of four dialogues developed to test how LLMs fare when formalising such sentences. Test results are presented in Section 4 and discussed in Section 5. Section 6 concludes and refers to future work. Overall, our results indicate that current LLMs are unable to provide reliable formalisations of platypus sentences.

2 Related Work

Logical formalisms for normative language in general and for the law in particular have been developed based on modern mathematical logic and related approaches since the 1920s (for systematic and historic overviews of these developments, cf., e.g., [22], [5], [16], [26]). These rather theoretical knowledge representation methods are sometimes also implemented, e.g., within programming languages such as `PROLOG` [30] and `PROLEG` [28], or within theorem provers [15], [31].

Specific challenges for the formalisation of natural language emerging within the legal domain have been investigated since the origins of modern logic of norms. This includes so-called *paradoxes of deontic logic* (cf., e.g., [16], [8]) and more general problems related to the semantic ambiguity of legal language. Formalisation methodologies for dealing with these challenges have been intensively investigated since the 1970s and 1980s by pioneers of legal informatics, e.g., by

Ilmar Tammelo [32], Leo Reisinger [24] and Laymen Allen [1]. The problems discussed in the paper at hand are directly inspired by previous work by Ron Klinger [11]. More recent approaches focus on the challenge of identifying legal norms beyond written legislation (e.g., norms derived from legal precedents) [3], on combining manual and automated methods to enhance the efficiency of generated formalisations [33], and on using legal visualisation methods to improve interdisciplinary communication among lawyers, logicians, computer scientists and engineers [27].

Previous research has analysed the general viability of machine translation from natural language reasoning into logical formalisms [2], including approaches based on LLMs [36], [25]. LogicLinguist, for example, is a specialised model for translating natural language statements into first-order logic [37]. In particular, the capability of such systems to detect logical fallacies has been analysed, with some results indicating that LLMs generally perform rather poorly on these tasks [9], while better results can be achieved by more streamlined systems adopting neurosymbolic approaches [13]. In the legal domain, machine translation methodologies using LLMs have been developed for outputs in `PROLEG` [19] and in `PROLOG` [38], the latter particularly stressing the value of also employing controlled natural languages (CNL) [35], [21], [12] as an intermediary step between natural language and logic.

While considerable previous research on the translation of natural language into logical formalisms is available, there is still little work on translation paradoxes emerging from natural language ambiguities. Even *LOGIC* [9], a data set especially designed for logical fallacy detection, mainly focusses on fallacies based on logically incorrect inferences (e.g., *ad populum* fallacies like "Everyone should like coffee: 95% of teachers do!"). Current research is lacking an analysis of the translation of natural language sentences which are logically consistent, but ambiguous in their translation into logical formalisms.

3 Methodology

In this section, we first introduce the notion of "platypus sentences", i.e., natural language sentences which are particularly difficult to formalise due to their semantic ambiguity (Section 3.1). Then, we describe short natural language dialogues containing platypus sentences consisting of negated natural language conditionals. Although all these sentences have the same syntactic structure, they have fundamentally different meanings, so that their proper formalisations in propositional logic strongly differ from one another. To test the capabilities of current LLMs with respect to formalising these sentences, we select different models to prompt for translating these dialogues into logical formulae (Section 3.3). Finally, since this task, as indicated by previous related work [9], is to be considered as a challenging task, we decide to use different prompts to thoroughly grasp the models' capabilities (Section 3.4).

3.1 Platypus sentences

Ideally, formal logic should correspond to Leibniz' notion of a *characteristica universalis*, i.e., a universal system of symbols capable of univocally representing every conceivable object, so that concepts and relations among these objects could be mapped to sets of ordered n-tuples among the respective symbols (for details, cf., e.g., [14], [29]). As a *characteristica universalis*, formal logic is generally not context-dependent: the meaning of logical symbols does not change depending on the logical formulae in which they occur. In this sense, semantics is embedded within syntax. This is, of course, not the case with natural language, which is rich in ambiguities, synonyms, homonyms, idioms, metaphors, figures of speech, etc.

This semantic discrepancy between natural language and logical formalisms is the source of many logical riddles, paradoxes, paralogisms and jokes. It is the root of odd deductions such as "all thieves are people; therefore: all *good* thieves are *good* people" and of "self-annihilating sentences" like "I am a firm believer in optimism; because if you don't have optimism, what else is there?" (for this and similar sentences cf., e.g., Saul Gorn's famous *Compendium of Rarely Used Cliches* [6]). Beyond serving as a source for intellectual entertainment, this semantic ambiguity of natural language can also lead to formalisation errors, especially when automated machine translation is employed. A proper formalisation should, for example, distinguish between the property of "being a cup of tea" in a sentence such as "this is a cup of coffee, not a cup of tea" and in a sentence such as "as a lawyer, formal logic is not my cup of tea". This distinction, however, presupposes an adequate consideration of the context in which these sentences are employed – a consequence of the fact that natural language is not context-independent and therefore not a *lingua characteristica* in the sense of Leibniz.

Inspired by the egg-laying, duck-billed, beaver-tailed, venomous, semiaquatic monotreme mammal, we call such context-dependent natural language sentences, which, due to their semantic ambiguity, could lead to formalisation errors (in particular when machine translation is used), *platypus sentences*. The class of platypus sentences is a wide, rather heterogenous class, which, as such, allows for several further sub-classifications. It is nonetheless relevant to analyse these sentences as a whole. Among them are sentences containing ambiguous properties (e.g., "being a cup of tea" or being "good") and sentences based on figurative language, but also sentences building on implicit assumptions derived from the specific context in which they appear (including, but not limited to, sentences with pronouns or relative adverbs such as "yesterday" or "behind").

Platypus sentences emerge not only from ambiguous names, adjectives, adverbs and other individually meaningful terms (i.e., from so-called *categorems*), but also from connectors, conjunctions etc. (i.e., from so-called *syncategorems*), which are meaningless on their own. For example, a particularly rich source of platypus sentences are conditionals, i.e., sentences with an "if... then..." structure. These sentences encompass various different semantics, e.g., the notion of material or strict implication, of causality, of conditional norms (sometimes

called "commitment" [34]), etc. This semantic ambiguity is all the more intensified when combining conditionals with negations, a further good syncategorematic source of platypus sentences. In our current investigation, we focus on syncategorematic platypus sentences. For this reason, the dialogues we prepare for testing include platypus sentences based on negated conditionals.

3.2 Dialogues

Drawing inspiration from previous related work, we develop four different dialogues containing platypus sentences consisting of negated conditionals, each leading to a different formalisation in propositional logic. The dialogue structure is important, because they provide the necessary context required to determining the concrete meaning of the platypus sentences.

Dialogue 1: The unwanted poem

Bob: It is Eve's birthday; you think if I write him a poem, he will be happy?

Alice: No, of course not!

Bob: Why not?

Alice: Eve hates poems.

The first dialogue contains the natural language conditional question of whether Eve will be happy (Q) if Bob writes Eve a poem (P). As a general thesis, this conditional could, in principle, be formalised as a material implication, i.e., by the formula $P \to Q$. However, in the particular discursive context of the dialogue, Alice's negative answer does not merely correspond to a logical negation of this formula (i.e., to the formula $\neg(P \to Q)$). Instead, she claims that Bob writing Eve a poem would actually lead to the opposite result from what Bob originally intended, i.e., Eve would not be happy; after all, he does not like poems. In other words, Alice's negative answer does not correspond to the negation of a material implication, but to a material implication with the same premise, but negated conclusion. The correct logical formula should therefore be $P \to \neg Q$.

Dialogue 2: The counsel's dilemma

Attorney: I do recognize that my client was at the scene of the crime. However, does being at the crime scene imply guilt?

Judge: No, I do not think so.

Attorney: In other words: it is not the case that: 'if my client was at the crime scene, then he is guilty'?

Judge: I agree.

Attorney: Then, it follows logically that my client is innocent.

The second dialogue shows a hypothetical discussion between an attorney and a judge. It traces back to a paper by Ron Klinger [11] and is centered arround a proposed conditional relationship between a person being at the scene of a crime (P) and that person to be guilty of the crime committed at the crime scene (Q). Again, as a general thesis, this could, in principle, be formalised by the material implication $P \rightarrow Q$. In the dialogue, the attorney tries to exploit the fact that the negation of this material implication (i.e., $\neg(P \rightarrow Q)$) is logically equivalent to the conjunction $P \wedge \neg Q$ to argue for the innocence of her client. However, in the specific context of this dialogue, the judge's denial of the natural language conditional does not amount to stating the logical negation of the general thesis (i.e., to claiming the negated material implication $\neg(P \rightarrow Q)$). Instead, the judge is merely stating that the general thesis is not being accepted, without committing to any verifunctional relationship between P and Q. In other words, the negation amounts to claiming that the propositions involved are, *ceteris paribus*, logically independent from one another. Thus, the most adequate formalisation of the judge's denial of the natural language conditional (if indeed any is required at all) would be the tautological formula $(P \wedge Q) \vee (\neg P \wedge Q) \vee (P \wedge \neg Q) \vee (\neg P \wedge \neg Q)$,[3] which, as such, is irrelevant for the rest of the argument proposed by the attorney.

Dialogue 3: Tweety, the penguin

Alice: My friend's dog was barking at my bird Tweetie yesterday, and Tweetie was very scared.

Bob: Oh, why did it not just fly away?

Alice: Tweety cannot fly!

Bob: Oh, I thought if Tweety is a bird, then it must be able to fly?

Alice: No, this is not true! Tweety is a penguin!

The third dialogue picks up the classic example of Tweety, the penguin (introduced by J. Pearl [23]), and is based on the natural language conditional of whether Tweety's "birdness" (its property of being a bird (P)), implies it being able to fly (Q). Once again, as a general thesis, this conditional could, in principle, be formalised by the material implication $P \rightarrow Q$. In this case, however, the negation of the natural language conditional is indeed adequately represented by the negation of a material implication: since Tweety is indeed both a flightless penguin and a bird at the same time, Tweety is an adequate counter-example to the general thesis expressed by $P \rightarrow Q$. Thus, the proper formalisation of Alice's negation of the natural language conditional is $\neg(P \rightarrow Q)$. In theory, this dialogue should be easier to formalise than the previous ones, because, in this case, the negation of the natural language conditional actually corresponds to the logical negation of a material implication, i.e., the formalisation that syn-

[3] This formula basically states that any combination of truth-values for P and Q is satisfiable, i.e., they can be both true or both false or one of them can be true while the other is false.

tactically most closely resembles the structure of the platypus-sentence. In this sense, this dialogue (and the next one, which, as described below, is based on the same structure as this one), can be taken as a control test.

Dialogue 4: The mysterious platypus

Alice: Yesterday at the zoo I saw a platypus! It is a mammal that lays eggs!

Bob: But if it's a mammal, it can't lay eggs, can it?

Alice: No, that's not true!

The last dialogue is a variant of the previous dialogue and addresses the conditional of whether a platypus, being a mammal (P), is not able to lay eggs ($\neg Q$). Like in the previous dialogue, the correct formalisation for the negation of this conditional is a negated material implication, albeit here with a negated consequence, i.e., $\neg(P \rightarrow \neg Q)$. We choose to include this variant of the previous dialogue to assess whether the LLMs' likely previous information on the famous Tweety problem would manifest itself in the quality of the results delivered, the expectation being that, if results' quality indeed diverge, the models should perform better with Dialogue 3 than with Dialogue 4.

3.3 Models

To grasp the capabilities of current state of the art LLMs in translating these dialogues to propositional logic, we perform tests on three different LLMs. As the most well-known and widely used LLM we chose ChatGPT for the first model. We use GPT-5, which is a GPT version promoted to be especially well-suited for reasoning tasks [20]. As the second model, we use Llama 3.3 [17] – another well-known, widely used model competing with ChatGPT. The last model we use is LogicLinguist, which is specialised for translating natural language statements into first-order logic [37]. Although it also employs GPT-4o for chat interactions, it uses *Z3 Theorem Prover* for logic solving. We interacted with all of these models via their online chat interfaces.

3.4 Prompts

We use two different prompts to evaluate the models' capabilities to translate the chosen dialogues to logical formulae.

The first prompt (Appendix A) is a simple prompt only telling the models to extract logical formulae for propositional logic for the given dialogue. This prompt is meant for evaluating a naïve approach users might take when trying to convert natural language to logical formalisms. Users might want a swift reply without a lot of prompt engineering to get results for their task.

The second prompt (Appendix B) is devised to help the models find the possible pitfalls in translating the given dialogues. The difficulty of translating a negated conditional is pointed out to the models so it can be taken into account.

Additionally, the models are tasked with checking the validity of the generated formulae for these negated implications to test whether the models can properly identify the specific difficulties in the dialogues.

4 Results

We now describe in detail how the models translated each dialogue to logical formulae. Overall, the tested models returned results of varying quality when prompted with the dialogues and prompts described above. Furthermore, models tended to perform better when the extended prompt was used. An overview of the results is provided in Table 1.

Table 1. Formalisation results for tested models; correct results in **blue bold font**.

	Prompt	GPT-5	Llama	LogicLinguist
Dialogue 1	simple	$\neg(P \to Q)$	$\neg(Q \wedge \neg Q)$	$\neg(P \to Q)$
	extended	$\mathbf{P \to \neg Q}$	$\neg(P \to Q)$	$\mathbf{P \to \neg Q}$
Dialogue 2	simple	$\neg(P \to Q)$	$\neg(P \to Q)$	$\neg(P \to Q)$
	extended	$\neg(P \to Q)$	$\neg(P \to Q)$	$\neg(P \to Q)$
Dialogue 3	simple	$\mathbf{\neg(P \to Q)}$	$\neg Q$	$\neg P$
	extended	$\mathbf{\neg(P \to Q)}$	$\mathbf{\neg(P \to Q)}$	$\mathbf{\neg(P \to Q)}$
Dialogue 4	simple	$\mathbf{\neg(P \to \neg Q)}$	$\mathbf{\neg(P \to \neg Q)}$	$\mathbf{\neg(P \to \neg Q)}$
	extended	$\mathbf{\neg(P \to \neg Q)}$	$\mathbf{\neg(P \to \neg Q)}$	$\mathbf{\neg(P \to \neg Q)}$

4.1 Dialogue 1: The unwanted poem

For the simple prompt, none of the models was able to generate the correct result ($P \to \neg Q$). GPT-5 and LogicLinguist both formalised the negated conditional as $\neg(P \to Q)$, while Llama rather oddly translated it to the tautology $\neg(Q \wedge \neg Q)$. When using the extended prompt, Llama still failed to correctly formalise the platypus-sentence in the dialogue, translating it to $\neg(P \to Q)$. With the extended prompt, the other two models correctly identified three possible formalisations for the negated natural language conditional – including the correct solution. Both also correctly concluded that, among the options, $P \to \neg Q$ is the most fitting translation for the platypus-sentence in this dialogue.

4.2 Dialogue 2: The counsel's dilemma

The platypus-sentence in the second dialogue was not formalised correctly by any model for any prompt. All models derived $\neg(P \to Q)$, which is not the correct translation. However, GPT-5 and LogigLinguist (which, we recall, employs GPT-4o) noted that the judge's negation of the natural language conditional would

not, from a pragmatic point of view, amount to stating $\neg(P \rightarrow Q)$, which would thus hinder the attorney from validly deducing $\neg Q$. While this is indeed the proper reasoning that should lead to the correct formalisation, the systems were unable to deliver it. This could be explained either by (1) the fact that the formalisation is trivial for the argument; (2) the fact that the formalisation bears almost no resemblance to the respective sentence; (3) the fact that the dialogue, through the attorney's final claim, might be taken as stressing the acceptance of $\neg(P \rightarrow Q)$ by the judge. Overall, the systems were unable to adequately grasp the absurdity of the attorney's argument.

4.3 Dialogue 3: Tweety, the penguin

For this dialogue, only GPT-5 returned the correct translation $\neg(P \rightarrow Q)$ when using the simple prompt. LogicLinguist rather oddly negated only the premise ($\neg P$), while Llama negated the consequence ($\neg Q$), which comes close to the correct answer (we recall that $\neg(P \rightarrow Q)$ is equivalent to $P \wedge \neg Q$). When using the extended prompt, all three models translated the platypus-sentence in the dialogue correctly and gave a proper explanation for this formalisation. While all models derived the formula based on the background knowledge that penguins are birds and that penguins cannot fly, only GPT-5 rather briefly referred to the problem as a known problem by calling it a "classic".

4.4 Dialogue 4: The mysterious platypus

For the previous dialogue, which contains a more well-known version of the same underlying problem, Llama and LogicLinguist failed when using the simple prompt. In contrast, for this dialogue, all models returned the correct result (i.e., $\neg(P \rightarrow \neg Q)$) with all prompts. Only Llama showed a slight inaccuracy when using the simple prompt by being incosistent with its usage of symbols. More precisely, it formalised "mamals cannot lay eggs" sometimes as Q and sometimes as $\neg Q$, and thus suggested $\neg(P \rightarrow Q)$ as a formalisation for the negated conditional, although it made clear that it had taken Q to mean "mammals cannot lay eggs". These results subvert our expectations, as we had assumed that models would fare better when formalising Dialogue 3 because of it dealing with the widely known Tweety problem.

5 Discussion

Overall, among the three models, GPT-5 and LogicLinguist (which, we recall, employs GPT-4o) performed noticeably better than Llama, which was only able to deliver correct results for Dialogues 3 and 4. As described above, these tasks can be seen as *easier* tasks, since the correct formalisation very closely resembles the syntactical structure of the respective platypus-sentence.

As expected, the extended prompt consistently leads to better results. This is promising, because the extended prompt is still *general* in the sense that it

does not contain any specific information on the respective dialogues. This shows that it is possible to get better results using more detailed prompts even without having to address concrete details of specific cases.

The fact that no performance advantage could be observed when comparing the results for Dialogues 3 and 4 is interesting. It indicates that, for some reason, previous information on the Tweety problem, which the systems are likely to have had easy access to, did not play any significant role in the formalisation. Indeed, one can conjecture whether this information could be counterproductive, as the Tweety problem is actually related to a different formalisation problem – namely that of defeasible reasoning –, which, however, does not play any significant role in the formalisation of Dialogue 3. This can be observed in GPT-5's results with the extended prompt, in which it is stated that this is a "classic where a counterexample to a universal implication shows up".

This issue is further highlighted by the fact that the reasoning presented by the models, while often leading to the correct results, was not, in its entirety, coherent. This is most clear in the results delivered by LogicLinguist, which, for both Dialogues 1 and 4, starts its answer by stressing that the dialogue illustrates that one ought not to confuse the negation of an implication with another implication. This is only true for Dialogue 4, not for Dialogue 1 in which, in fact, the negation of the conditional does correspond to a "different implication" (i.e., an implication with same premise, but different conclusion). Incidentally, this underlying problem of confusing the negation of a conditional with another conditional is at the core of the famous *Barbershop Paradox*, a material implication paradox formulated by Lewis Carroll at the end of the 19^{th} Century [4].

This incoherence in the underlying reasoning displayed by the systems could be the cause of the odd results delivered by Llama (Dialogues 1 and 3) and by LogicLinguist (Dialogue 3) when using the simple prompt. Moreover, these results indicate that merely providing more information to the systems might not lead to better results, as the information may have the effect of misleading the systems away from the right formalisation.

6 Conclusion

Translating natural language to logical formulae is a challenging task for LLMs. In this paper, we showed that there are some kinds of sentences which may be especially hard to translate to logical formulae ("platypus sentences"). Inspecting one type of such difficult sentences – negated conditionals – we tested current state of the art LLMs' capabilities in translating these sentences to logical formulae in propositional logic. Our experiments showed that using only a simple prompt often results in incorrect translations, while even a more intricate prompt does not guarantee a correct solution.

As platypus sentences are a legitimate difficulty when translating natural language to logical formulae, current and future systems devised for such a purpose should be evaluated on such sentences to gauge their capabilities. Further

research is needed to analyse the performance of state of the art LLMs when confronted with other types of platypus sentences. Additionally, given the small number of examples addressed in this paper, more examples which may vary in length or complexity need to be considered for further investigations.

Overall, the fact that the models largely failed to deliver proper results in the most difficult cases of Dialogues 1 and 2 – with all three systems failing in the case of Dialogue 2 and Llama and LogicLinguist failing even in the easier Dialogues 3 and 4 with the simple prompt – indicates that LLM-based machine formalisation is still far from being a reliable source for adequate formalisations of natural language, especially when ambiguous sentences are involved. Hence, if they are to ever catch the platypus, LLMs ought better start putting their shoulder to the wheel – ideally without losing time searching for a wheel or trying to develop an actual shoulder.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Allen, L.: Formalizing hohfeldian analysis to clarify the multiple senses of 'legal right': A powerful lens for the electronic age. S. Cal. L. Rev. 48 pp. 428–487 (1974)
2. Angeli, G., Manning, C.D.: NaturalLI: Natural logic inference for common sense reasoning. In: Moschitti, A., Pang, B., Daelemans, W. (eds.) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 534–545. Association for Computational Linguistics, Doha, Qatar (Oct 2014). https://doi.org/10.3115/v1/D14-1059, https://aclanthology.org/D14-1059/
3. Borges, G., Wüst, C., Sasdelli, D., Margvelashvili, S., Klier-Ringle, S.: Making the implicit explicit: The potential of case law analysis for the formalization of legal norms. In: Borges, G., Satoh, K., Schweighofer, E. (eds.) Proceedings of the International Workshop on Methodologies for Translating Legal Norms into Formal Representations (LN2FR 2022) in association with 35th International Conference on Legal Knowledge and Information Systems (JURIX 2022). pp. 66–75 (2023)
4. Carroll, L.: A logical paradox. Mind **3**(11), 436–438 (1894)
5. Gabbay, D., Horty, J., Parent, X., van der Meyden, R., van der Torre, L. (eds.): Handbook of Deontic Logic and Normative Systems, Vol 1. College Publications, London, UK (2013)
6. Gorn, S.: Self-annihilating sentences: Saul gorn's compendium of rarely used cliches. Tech. rep., University of Pennsylvania (1992), https://repository.upenn.edu/entities/publication/3758e6d1-f02b-4b65-9615-6946fbe4e412
7. Governatori, G., Rotolo, A.: How do agents comply with norms? In: 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology. vol. 3, pp. 488–491 (2009). https://doi.org/10.1109/WI-IAT.2009.332
8. Hilpinen, R., McNamara, P.: Deontic logic: A historical survey and introduction. In: Gabbay, D., Horty, J., Parent, X., van der Meyden, R., van der Torre, L. (eds.) Handbook of Deontic Logic and Normative Systems, Vol 1, pp. 3–136. College Publications, London, UK (2013)

9. Jin, Z., Lalwani, A., Vaidhya, T., Shen, X., Ding, Y., Lyu, Z., Sachan, M., Mihalcea, R., Schoelkopf, B.: Logical fallacy detection. In: Goldberg, Y., Kozareva, Z., Zhang, Y. (eds.) Findings of the Association for Computational Linguistics: EMNLP 2022. pp. 7180–7198. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Dec 2022). https://doi.org/10.18653/v1/2022.findings-emnlp.532, https://aclanthology.org/2022.findings-emnlp.532/
10. Kampik, T., Mansour, A., Boissier, O., Kirrane, S., Padget, J., Payne, T.R., Singh, M.P., Tamma, V., Zimmermann, A.: Governance of autonomous agents on the web: Challenges and opportunities. ACM Trans. Internet Technol. **22**(4) (Nov 2022), https://doi.org/10.1145/3507910
11. Klinger, R.: The paradox of counter-conditional and its dissolution. Jurimetrics Journal **11**(4), 189–193 (1971), http://www.jstor.org/stable/29761216
12. Kowalski, R., Datoo, A.: Logical english meets legal english for swaps and derivatives. Artificial Intelligence and Law **30**(2), 163–197 (Jun 2022), https://doi.org/10.1007/s10506-021-09295-3
13. Lalwani, A., Kim, T., Chopra, L., Hahn, C., Jin, Z., Sachan, M.: Autoformalizing natural language to first-order logic: A case study in logical fallacy detection (2025), https://arxiv.org/abs/2405.02318
14. Lenzen, W.: Leibniz and the calculus ratiocinator. Philosophy of Engineering and Technology **30**, 47 – 78 (2018). https://doi.org/10.1007/978-3-319-93779-3_4, cited by: 5
15. Libal, T., Steen, A.: Nai: The normative reasoner. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law. p. 262–263. ICAIL '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3322640.3326721
16. McNamara, P., Van De Putte, F.: Deontic Logic. In: Zalta, E.N., Nodelman, U. (eds.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, Fall 2022 edn. (2022)
17. Meta AI: Llama 3.3 model cards and prompt formats (2025), https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3/, accessed: 2025-10-30
18. Mowbray, A., Chung, P., Greenleaf, G.: Representing legislative rules as code: Reducing the problems of 'scaling up'. Computer Law & Security Review **48**, 105772 (2023). https://doi.org/https://doi.org/10.1016/j.clsr.2022.105772, https://www.sciencedirect.com/science/article/pii/S0267364922001157
19. Nguyen, H.T., Wachara, F., Nishino, F., Satoh, K.: A multi-step approach in translating natural language into logical formula. In: Francesconi, E., Borges, G., Sorge, C. (eds.) Legal knowledge and information systems. Frontiers in Artificial Intelligence and Applications, IOS Press, Amsterdam and Berlin and Washington, D. C (2022). https://doi.org/10.3233/FAIA220453
20. OpenAI: Introducing gpt-5 (2025), https://openai.com/index/introducing-gpt-5/, accessed: 2025-10-30
21. Pace, G.J., Rosner, M.: A controlled language for the specification of contracts. In: Fuchs, N.E. (ed.) Controlled Natural Language. pp. 226–245. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
22. Parent, X., van der Torre, L.: Introduction to Deontic Logic and Normative Systems. College Publications, Rickmansworth, UK (2018)
23. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1988)

24. Reisinger, L.: Probleme der symbolisierung und formalisierung im recht. In: Winkler, G. (ed.) Rechtstheorie und Rechtsinformatik, pp. 22–50. Springer, Vienna, Austria (1975)
25. Ryu, H., Kim, G., Lee, H.S., Yang, E.: Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. arXiv preprint arXiv:2410.08047 (2024)
26. Sasdelli, D.: Können Maschinen Rechtsfälle entscheiden? Felix Meiner Verlag (2025). https://doi.org/10.28937/978-3-7873-4900-5
27. Sasdelli, D., Steffes, B., Herrmann, M., Chitashvili, M., Wüst, C.: A normal form for representing legal norms and its visualisation through normative diagrams. Annual International Conference on Digital Government Research **26** (Jun 2025). https://doi.org/10.59490/dgo.2025.1036, https://proceedings.open.tudelft.nl/DGO2025/article/view/1036
28. Satoh, K.: PROLEG: Practical legal reasoning system. In: Warren, D.S., Dahl, V., Eiter, T., Hermenegildo, M.V., Kowalski, R., Rossi, F. (eds.) Prolog: The Next 50 Years, pp. 277–283. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-35254-6_23
29. Schneider, M.: Leibniz' konzeption der "characteristica universalis" zwischen 1677 und 1690. Revue Internationale de Philosophie **48**(188 (2)), 213–236 (1994), http://www.jstor.org/stable/23949527
30. Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T.: The british nationality act as a logic program. Commun. ACM **29**(5), 370–386 (May 1986), https://doi.org/10.1145/5689.5920
31. Steen, A., Benzmüller, C.: The higher-order prover leo-iii. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) Automated Reasoning. pp. 108–116. Springer International Publishing, Cham (2018)
32. Tammelo, I.: Modern Logic in the Service of Law. Springer, Vienna, Austria (1978)
33. Witt, A., Huggins, A., Governatori, G., Buckley, J.: Encoding legislation: a methodology for enhancing technical validation, legal alignment and interdisciplinarity. Artificial Intelligence and Law **32**(2), 293–324 (Jun 2024), https://doi.org/10.1007/s10506-023-09350-1
34. von Wright, G.H.: I. deontic logic. Mind **60**(237), 1–15 (1951). https://doi.org/10.1093/mind/lx.237.1
35. Wyner, A., Angelov, K., Barzdins, G., Damljanovic, D., Davis, B., Fuchs, N., Hoefler, S., Jones, K., Kaljurand, K., Kuhn, T., Luts, M., Pool, J., Rosner, M., Schwitter, R., Sowa, J.: On controlled natural languages: Properties and prospects. In: Fuchs, N.E. (ed.) Controlled Natural Language. pp. 281–289. Springer, Berlin (2010)
36. Yang, Y., Xiong, S., Payani, A., Shareghi, E., Fekri, F.: Harnessing the power of large language models for natural language to first-order logic translation. arXiv preprint arXiv:2305.15541 (2023)
37. YesChat.ai: Logic linguist – logic translation tool. https://www.yeschat.ai/gpts-9t557oyNbN2-Logic-Linguist (2025), accessed: 2025-10-31
38. Zin, M., Borges, G., Satoh, K., Fungwacharakorn, W.: Towards machine-readable traffic laws: Formalizing traffic rules into prolog using llms. In: Maranhão, J. (ed.) Proceedings of ICAIL2025, pp. 317–326. ACM (2025). https://doi.org/10.1145/3769126.3769204

A Simple Prompt

Extract logical formulae (propositional logic) from the following dialogue.

B Extended Prompt

Extract logical formulae (propositional logic) from the following dialogue. Pay special attention to the negations of implications. Check their validity very carefully.

When Legal Articles Resist Formalisation

Ludi van Leeuwen[1][0000−0003−3165−4376], Tadeusz Zbiegień[2][0000−0001−9052−6978], and Cor Steging[1][0000−0001−6887−1687]

[1] Bernoulli Institute of Mathematics, Computer Science and Artificial Intelligence, University of Groningen
[2] Department of Legal Theory, Jagiellonian University

Abstract. Formal representations of legal text have long been central to research in AI and Law. In this study, we examine the process of formalising legal articles through a co-design approach conducted with a legal expert. Using the ANGELIC framework, we explore the formalisation of Japanese Civil Code articles and analyse where and why some articles 'resist' unambiguous formalisation. Through this analysis, we identify four lessons learnt about the limits of formalizations. These concern the inherent complexity and dynamic nature of the law, the concept of model granularity, and interpretive ambiguity in legal text. We hope these lessons can inform future efforts to design, apply, and evaluate formalisation methods in the legal domain.

Keywords: Legal formalisation · AI and Law · Knowledge representation · Legal reasoning · Limitations of formal models · Symbolic AI.

1 Introduction

There is a long-standing tradition of modelling legal knowledge in formal representations in the AI & Law domain. Such approaches include rule-based logics, case-based structures, ontologies, and argumentation frameworks [28,2,32,30]. Such representations can be used to reason with, to support domain understanding, and to enable automatic decision-making. Recently, while machine learning approaches are dominant, there has been a surge in neuro-symbolic approaches that combine data-driven AI techniques with formal representations [22]. This paper reflects on the practice of formalising legal articles. Drawing on concrete examples, we highlight situations in which the process of formalisation encounters resistance, revealing limitations, ambiguities, and structural challenges inherent to legal text.

In this paper, we report on the co-design of formal representations of legal articles, conducted with a legal expert, aiming to model a diverse set of provisions from the Japanese Civil Code. While several articles could be represented successfully, others proved difficult or nearly impossible to formalise unambiguously under our initial assumptions. This paper distills the lessons learned from these challenges and highlights design aspects that require explicit attention to produce usable formal models.

2 Background

Formalising law can be defined in terms of translation [29]: it is a process of translating legal provisions and norms into a chosen formal representation and has a long and extensive tradition in legal scholarship [26,24]. At the same time, legal provisions are known for the number of problems they generate. It seems that in many cases it is the specificity of legal language itself that makes it a rewarding subject of research. This includes in particular works on the issues of vagueness [12,23], defeasibility [16,8], context sensitivity [20] or reasoning based on values [6]. These issues have been reflected in a number of modern works that attempt to harness these properties in computational form. Previous work in this domain concerned, among others, the defeasibility of legal reasoning [7], application of fuzzy [10] and temporal logic [14].

2.1 ANGELIC

In our study, we represent the legal articles as ANGELIC Domain Models (ADMs). The ANGELIC methodology, later updated to ANGELIC II, is an approach for representing and reasoning about legal domains [1,3]. It has been used to model a variety of domains, such as US Trade Secrets [5] and the European Convention on Human Rights [9,4]. The ADMs that model legal articles are based on Abstract Dialectical Frameworks (ADFs), a generalisation of argumentation frameworks capable of modelling argumentation [11,19]. In these models, legal articles are organised hierarchically: the root node represents the article's verdict, which is determined by a set of issues, each in turn defined by a set of factors. Factors are legally relevant fact patterns that may or may not apply to a given case [2].

In ANGELIC, factors are divided into two types. Baselevel factors form the leaves of the hierarchy and are assigned a truth value (True', False', or 'Unknown') for each case, which is referred to as factor ascription. Abstract factors capture intermediary legal concepts and are defined by other abstract or baselevel factors. The values of abstract factors, as well as issues and the verdict, are calculated using accept and reject conditions along with a default value, applying logical operators (AND, OR, NOT) evaluated under three-valued logic [25].

2.2 Japanese Civil Code

In this study, we focus on modelling a selection of articles from the Japanese Civil Code, a comprehensive body of law governing private legal relations in Japan, including contracts, property, family, and obligations. The Civil Code is highly structured, with articles generally short and narrowly scoped, making it potentially suitable for formalisation. Our work is inspired in part by previous work in the COLIEE competition [15,31], a legal competition that provides benchmarks for legal information extraction and entailment tasks using the Japanese Civil Code.

For this study, we model the legal articles directly as a verdict determined by a set of factors, without explicitly representing intermediate issues. This simplification reflects the small scope of most articles. Despite this, modelling even these apparently straightforward provisions revealed challenges related to interpretive ambiguities, contextual dependencies and the interplay of multiple legal concepts, highlighting the limits of formalisation even in well-defined domains.

3 Constructing ADMs of Legal Articles

3.1 Overview of the approach

Our approach was based on a co-design process in collaboration with a legal expert. Together, we constructed ADMs for a selection of articles from the Japanese Civil Code. The co-design process involved iterative interpretation, where we discussed how each article could be represented in the ADM structure and how its legal concepts could be formalised as factors and logical relations. The goal was not necessarily to produce complete models but more so to examine how the process of formalisation unfolds when technical and legal perspectives interact. Through this collaboration, we explored both the strengths and the practical limits of formalising legal text.

3.2 Effective formalisations

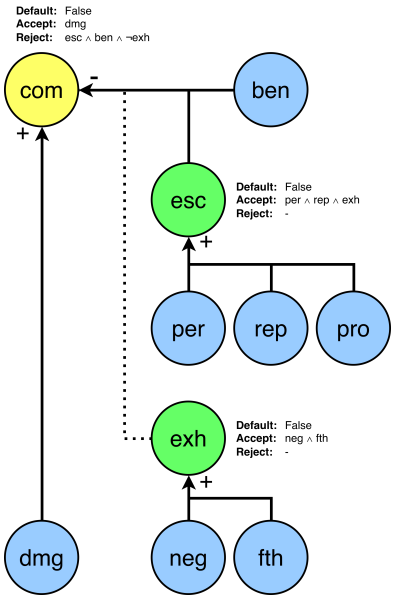
Our approach proved to be effective for a number of articles. To illustrate a successful example, Figure 1 presents the ADM corresponding to Article 698 (see Figure 1a for the text of the article). The verdict, shown in yellow in Figure 1b, concerns whether the manager is liable to compensate for damages. This outcome is determined through the ascription of the base-level factors (blue), which in turn define the values of the abstract factors (green). Together, these factors, and the accept and reject conditions, yield the final value of the verdict.

We found that ADMs can be constructed most successfully for articles with a clear and logical structure that corresponds to the verdict. The methodology in these cases can be considered *effective* in the sense that it enabled the identification of the article's relevant characteristics and factors, yielding an ADM that can determine the verdict concerning it.

We have however encountered a number of informative edge cases. Those cases allowed us to draw several lessons concerning not so much the specific formalisation method used, but more broadly the general problem of formalising legal articles as such. We dive deeper into some of these edge cases in the next sections.

> If a manager engages in benevolent intervention in another's business in order to allow a principal to escape imminent danger to the principal's person, reputation, or property, the manager is not liable to compensate for damage resulting from this unless the manager has acted in bad faith or with gross negligence.

(a) Article 698 of the Japanese Civil Code.



(b) ADM of Article 698

Verdict:

com The manager is liable to compensate for damages

Abstract factors:

esc The manager's actions aimed to allow the principal to escape imminent danger

exh The manager exhibited gross negligence or acted in bad faith

Baselevel factors:

ben The manager's intervention was benevolent

dmg There were damages resulting from the intervention

per The actions aimed to allow the escape of imminent danger to the person

rep The actions aimed to allow the escape of imminent danger to reputation

pro The actions aimed to allow the escape of imminent danger to property

neg The manager exhibited gross negligence

fth The manager acted in bad faith

(c) Verdict and factors of the ADM

Fig. 1: Article 698 of the Japanese Civil Code (a) alongside its ANGELIC Domain Model (b) and its verdict and factors (c).

3.3 Determining the verdict

In the case of Article 698 (Figure 1a), we determined that the *verdict* should be whether the manager is liable to compensate for damages. This answers the question, '*should the manager compensate for the damages?*'. Although this appears to be the most straightforward question to ask, a legal article can invite

several different questions. In such cases, it becomes more difficult to determine what the verdict of the ADM should represent.

We illustrate this concept using Article 593 of the Japanese Civil Code, shown in Figure 2a. Through a co-design session with the legal expert, we developed the ADM shown in Figure 2, where the verdict was defined as whether the loan for use is effective. This formulation allows reasoning about cases where the central question is: '*is the loan for use effective*?'.

However, other questions can be asked about this article, in which the constructed ADM cannot be applied. For example, two bar exam questions concerning Article 593 are shown in Figure 3, where examinees must determine whether a legal entailment exists between Article 593 and the statements in Figure 3. This task corresponds to Task 4 of the COLIEE competition [15].

When it comes to the first statement (H30-24-U), the implied question is: '*May B demand compensation for damages due to default based on the loan for use contract from A?*'. In the second statement statement (R1-23-3), the implied question is: '*Is a loan for use contract effective when not made in writing?*'. Strictly speaking, neither of these two questions can be answered directly ascribing factors to the ADM in Figure 2, as the questions do not match the question we had in mind for the verdict node when designing the ADM. To answer these questions, we would need a different ADM, tailored to those specific questions.

3.4 Complexity, dynamics and temporal effects

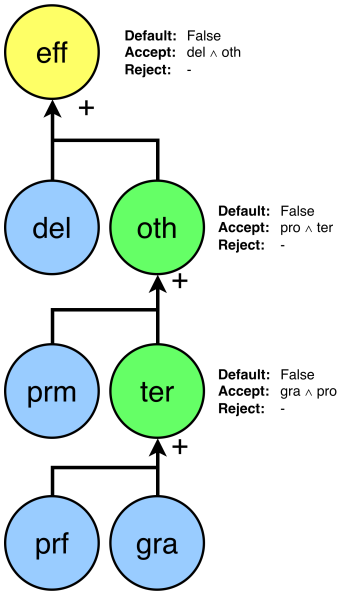
The ADM of Article 593 (Figure 2) also illustrates issues with regards to the symbolic structure and the affordances needed to represent legal articles. For example, we see that the modal verb *promise* is used. Additionally, the article contains reference to temporal aspects, such as a thing being returned *after* a contract is terminated, and the contract being terminated *after* the other party uses and makes profit of the thing. Furthermore, one needs to perform entity disambiguation to resolve references to *the borrowed thing*, *party* and *other party*. Hence, in this example alone, a formalism needs to account for logical connectives as well as modal, temporal and entity-level distinctions.

While ADMs are designed to correctly represent the structure of the logical connectors, they are in principle not designed to represent these other distinctions explicitly using deontic or temporal logic. Choosing a formalism hence means picking a (sub)set of relevant attributes of legal reasoning to represent, and leaving the rest implicit.

It is also the case that we often cannot reasonably answer legal questions only based on a single article. Instead, we need to read the article in the context of bigger network of interrelated articles, and consider *law as a complex and dynamic system* [21,17,18]. For example, to answer questions in Figure 3, the ADM in Figure 2 alone is insufficient and we would require additional articles to perform the legal reasoning necessary. Strictly speaking, in order for our formalisation to be able to answer legal questions adequately, it would have to take into account the fact that it is rare for a given legal question to be answered on the basis of a single provision, usually, a number of different regulations must be

A loan for use becomes effective if one of the parties promises to deliver a certain thing, and the other party promises to return the thing when the contract is terminated after the other party gratuitously uses and makes profit of the borrowed thing.

(a) Article 593 of the Japanese Civil Code.



(b) ADM of Article 593

Verdict:

eff The loan for use is effective

Abstract factors:

oth The other party promises to return the thing when the contract is terminated after [the other party] gratuitously uses and makes profit of the borrowed thing.

ter The contract is terminated after the other party gratuitously uses and makes a profit from the borrowed thing.

Baselevel factors:

del One of the parties promised to deliver a certain thing

prm The other party promises to return the borrowed thing when the contract is terminated

prf The other party makes a profit from the borrowed thing

gra The other party gratuitously uses the borrowed thing

(c) Verdict and factors of the ADM

Fig. 2: Article 593 of the Japanese Civil Code (a) along with its ANGELIC Domain Model (b) and its verdict and factors (c).

taken into account, which combine into different structures depending on how the question is asked.

3.5 Model granularity

In the ADM of Article 593, the verdict is determined by two factors, one of which in turn are determined by sub-factors, and one of those by sub-sub-factors as shown in Figure 2. This raises a broader methodological question about the appropriate level of detail when decomposing factors in ADMs. For example, should factor *oth* be further broken down into its constituent sub-factors, or is it sufficient to treat it as a single baselevel factor? This leads to the open issue of

H30-24-U: A who is the owner of a building and B made a loan agreement where B borrows the building for free only while A is working abroad, and then, however, A made a loan contract with third party C, and transferred the building. In such case, B may not demand compensation for damages due to default based on the loan for use contract from A.

R1-23-3: No contract of loan for use shall be effective unless it is made in writing.

Fig. 3: Two bar exam questions concerning Article 593.

model granularity. On the one hand, there is a natural inclination to subdivide the model into as many explicit conditions as possible. For example, in case law when determining whether a vague term such as 'gratuitous usage' applies. On the other hand, excessive granularity risks undermining clarity: the model may become unreadable, and each additional node, particularly when not directly supported by the text of the provision, and increases the risk of falling into the trap of explicitly encoding certain modelling choices.

3.6 Model uncertainty and scope ambiguity

In terms of ADMs, decision trees or graphs in general, *model uncertainty* in a legal context will refer to the uncertainty arising from the possibility of creating multiple (equally or unequally) justifiable edges with various logical operators assigned between the nodes representing the conditions. Simply put, it refers to a situation where it is possible to construct similarly justified models of the same article or legal norm resulting from it, which differ in their logical structure. Such model uncertainty can be caused by *scope ambiguity*, where logical operators like 'and' or 'or' are used without clear specification of their range, allowing multiple plausible ways to structure the logical relationships.

To illustrate this concept, we show an artificial article based on Article 178a of the Polish Criminal Code in Figure 4a [3]. In this article, a scope ambiguity is evident due to the use of or' and and' without clarity on the range of these connectors. This ambiguity permits at least two different ADMs, each representing one possible reading. Both readings are shown in Figures 4c and 4d. While seemingly superficial, this example illustrates model uncertainty due to scope ambiguity clearly, and reflects the kinds of interpretative disagreements that frequently arise in legal analysis. Importantly, this is not a classic problem of linguistic ambiguity and the scope of meaning of individual concepts. The uncertainty does not arise at the level of the meaning of individual concepts or

[3] This example comes from a webinar held by the Government Legislation Centre (RCL) on 21 August 2025[27]

```
Anyone who drives a vehicle
under the influence of alcohol
or intoxicating substances and
poses a threat is subject to
punishment.
```

(a) Artificial article based on Article 178a of the Polish Criminal Code.

Verdict:

pun Subject to punishment?

Baselevel factors:

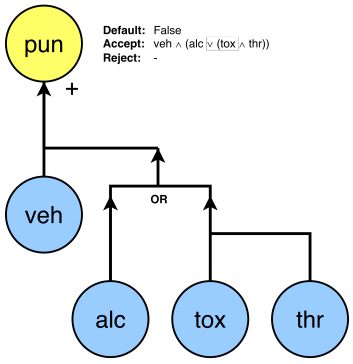
veh Drives a vehile

alc Under influence of alcohol

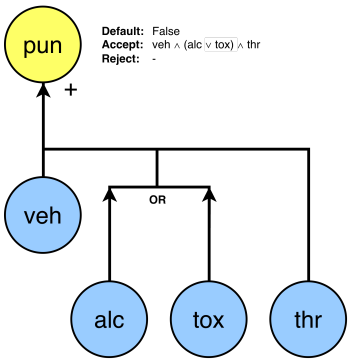
tox Under influence of intoxicating substances

thr Poses a threat

(b) Verdict and factors of the ADMs



(c) First ADM interpretation



(d) Second ADM interpretation

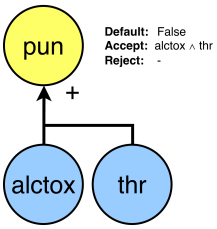
Fig. 4: Artificial article based on Article 178a of the Polish Criminal Code (A) alongside two different ANGELIC Domain Models the (c & d) and its verdict and factors (b).

the scope of their meaning, but at the *level of the relationships between these concepts.* Because of this, formalising articles can also help expose such scope ambiguity, which can be useful for legal scholars and personnel.

3.7 Artificially generated formalisations

Since the advent of Large Language Models (LLMs), research has investigated the possibility of automatically extracting formal representations from legal texts [33]. In a previous study, we found that automatically generated ADMs were often ineffective [31]. Examining these artificial formalisations does further highlight the observations we have made during our co-design process with a human legal expert.

For example, when prompting ChatGPT to formalise an ADM of the article in Figure 4a, the LLM shows a persistent preference (10 out of 10 runs) for the second reading (see Figure 4d). Even with additional prompting, ChatGPT does not generate the first reading as shown in Figure 4c. We also see that the LLM takes a less granular approach, as it collapses the 'or' statement into one baselevel



(a) GPT-generated ADM

Verdict:

pun Is the driver subject to punishment?

Baselevel factors:

alctox Is the driver under the influence of alcohol or intoxicating substances?

thr Does the driver pose a threat to others or public safety?

(b) Verdict and factors of the ADM

Fig. 5: The ANGELIC Domain Models of the article from Figure 4a as generated by ChatGPT (a) and its generated verdict and factors (b).

factor The structure of the generated ADMs is consistent across runs, with only minor paraphrasing of the text within factors, as would be expected from a language model. The consistent preference of ChatGPT for the second reading implies that it does not expose the scope ambiguity of this legal article, which can be harmful if the formalisation is used to reason with by legal professionals.

4 Lessons and Design Implications

In our analysis, we used the process of formalisation through ADMs as a prototypical example to illustrate broader challenges in representing legal articles formally. In many cases, the ADM structure enables the construction of models that can be used to reason about legal questions in a structured and interpretable manner. However, several fundamental challenges remain. From our observations, we distil several key lessons and corresponding design implications for future work

4.1 Lesson 1: No single formalisation fits all legal questions

As discussed in Section 3.3, no single ADM or formalisation can capture all possible questions that may arise from a given article. Although the content of legal articles often appears to imply a specific question they are meant to address, the reality is more complex. In practice, the same article may give rise to multiple legitimate questions, each requiring a distinct formalisation. The same article can also often also be a building block to answer other questions relating to other articles.

4.2 Lesson 2: Law is complex, temporal and dynamic

In Section 3.4, we showed that although legal provisions are written in texts that at first appear clearly structured and stable, in reality, the law, the interpretation of articles, the reconstruction of norms, and the arguments surrounding them are highly susceptible to change and instability. Following a long-standing tradition

in legal theory, we distinguish between an *article* (or *legal provision*), understood as an editorial unit of legal text, and the corresponding *legal norm*, which describes the structure of an obligation or duty and is reconstructed through interpretation from one or more provisions. Law is therefore a complex and dynamic system. Despite a long tradition of formalisation in law, the actual, effective modelling of these two facts remains a crucial challenge, and future work is needed that focuses on these temporal and referential aspects of legal norms.

4.3 Lesson 3: Granularity is an explicit design choice

As shown in Section 3.5, the level of detail, or granularity, of a model is an explicit design choice with significant implications for its structure and effectiveness. For a formalisation to be effective, however understood, the granularity of the model must therefore be taken into account. There is always a trade-off between descriptive accuracy and interpretability. In striving for a more precise representation, we may lose readability and, consequently, practical applicability.

The appropriate level of granularity depends on the purpose of the formalisation. Is the goal to construct an algorithm capable of automating a process defined by legal norms? Is it to study the nature of legal norms or to model legal reasoning itself? Or is the aim to optimise performance on a computational benchmark? In some cases, the purpose may even be exploratory or conceptual. These questions should precede any formalisation effort, as the intended function of the model directly determines the depth and structure of its design.

4.4 Lesson 4: A single article can yield multiple valid models

Model uncertainty arises when a single legal article can be represented in multiple equally justified ways. One source of this uncertainty, as shown in Section 3.6, is scope ambiguity, where connectors such as and' or or' can be interpreted in different ways, producing distinct ADMs.

Recognising that legal norms are derived from many different sources, that their meaning may evolve over time, and that the linguistic boundaries of legal terms are rarely precise, is only a starting point. It should be noted that all these building blocks can be combined in various ways, none of which should be considered "the one and only correct" (especially before a decision is made to settle this type of legal dispute in a given case, and even then we only obtain certainty *ex post* in that one case – the decision-making pattern does not necessarily have to be repeated in subsequent similar or even identical cases, even if there are a number of legal and procedural safeguards in place to ensure this). It is precisely this inherent indeterminacy that we refer to as *model uncertainty*, and it is one of the features that makes law an especially challenging and intellectually rich domain. In this context, research addressing uncertainty directly seems to be warranted. Methods that avoid creating a false (and, in the context of law, dangerous) sense of stability and instead capture the uncertainty intrinsic to legal reasoning are essential. In particular, research on uncertainty quantification in legal models appears to be a promising direction.

5 Discussion and conclusion

While this study focused primarily on hard cases, we do not want to suggest that law consists only of such difficult cases subject to varying levels and types of uncertainty. A legal system in which every interpretation required complex logical or interpretative operations would be difficult to imagine. In practice, many articles and cases are relatively straightforward [13], which enables formalisation and supports the development of e-government services and the automation of administrative procedures in many countries. However, it is the edge cases that provide the most insight into both the formalisation process and the law itself. While we used the ANGELIC methodology in this study, our aim is not to evaluate the specific limitations or capabilities of this framework, but rather to examine the properties of law itself as an object of study. Different formalisation methods might highlight other issues, but our focus is on the structure of norms and the relationships between them, rather than on the fulfilment of specific conditions, which has already received substantial attention.

In this study, we analysed the process of formalising legal provisions through a co-design approach with a legal expert. Our experience demonstrated that, while many articles can be effectively formalised, certain provisions resist formalisation due to inherent characteristics of law. From these cases, we derived four key lessons: there is not a single formalisation suitable for every question; law is complex, temporal, and dynamic; model granularity is an explicit design choice; and multiple representations of the same article can exist, giving rise to model uncertainty. We argue that these lessons are broadly applicable and should inform future research on legal formalisation, guiding both the development of formal methods and the interpretation of law as a structured domain.

References

1. Al-Abdulkarim, L., Atkinson, K., Bench-Capon, T.: A methodology for designing systems to reason with legal cases using abstract dialectical frameworks. Artif. Intell. Law **24**(1), 1–49 (Mar 2016). https://doi.org/10.1007/s10506-016-9178-1
2. Aleven, V.: Teaching Case-based Argumentation through a Model and Examples. Ph.D. thesis, University of Pittsburgh (1997)
3. Atkinson, K., Bench-Capon, T.: Angelic ii: An improved methodology for representing legal domain knowledge. In: Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law. pp. 12–21. ICAIL '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3594536.3595137
4. Atkinson, K., Collenette, J., Bench-Capon, T., Dzehtsiarou, K.: Practical tools from formal models: the echr as a case study. In: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law. pp. 170–174. ICAIL '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3462757.3466095
5. Bench-Capon, T., Gordon, T.F.: Implementing a theory of a legal domain. In: Legal Knowledge and Information Systems, pp. 13–22. IOS Press (2022)

6. Bench-Capon, T., Modgil, S.: Norms and value based reasoning: Justifying compliance and violation. Artificial Intelligence and Law **25**, 29–64 (2017). https://doi.org/10.1007/s10506-017-9194-9
7. Billi, M., Calegari, R., Contissa, G., Lagioia, F., Pisano, G., Sartor, G., Sartor, G.: Argumentation and defeasible reasoning in the law. J **4**(4), 897–914 (2021). https://doi.org/10.3390/j4040061
8. Brożek, B.: Law and defeasibility. Revus (23) (2014). https://doi.org/10.4000/revus.3110
9. Collenette, J., Atkinson, K., Bench-Capon, T.: Explainable AI tools for legal reasoning about cases: A study on the European Court of Human Rights. Artificial Intelligence **317**, 103861 (2023)
10. da Costa Pereira, C., Tettamanzi, A.G.B., Liao, B., Malerba, A., Rotolo, A., van der Torre, L.: Combining fuzzy logic and formal argumentation for legal interpretation. In: Proceedings of the 16th Edition of the International Conference on Articial Intelligence and Law. p. 49–58. ICAIL '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3086512.3086532
11. Dung, P.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence **77**(2), 321–357 (1995)
12. Endicott, T.A.O.: Vagueness and legal theory. Legal Theory **3**(1), 37–63 (1997). https://doi.org/10.1017/S135232520000063X
13. Fischman, J.B.: How many cases are easy? Journal of Legal Analysis **13**(1), 595–656 (2021). https://doi.org/10.1093/jla/laaa010
14. Ghari, M.: A formalization of the protagoras court paradox in a temporal logic of epistemic and normative reasons. Artificial Intelligence and Law **32**(2), 325–367 (2024). https://doi.org/10.1007/s10506-023-09351-0
15. Goebel, R., Kano, Y., Kim, M., Rabelo, J., Satoh, K., Yoshioka, M.: Overview of Benchmark Datasets and Methods for the Legal Information Extraction/Entailment Competition (COLIEE) 2024. In: Suzumura, T., Bono, M. (eds.) New Frontiers in Artificial Intelligence. pp. 109–124. Springer Nature Singapore, Singapore (2024)
16. Governatori, G.: An ASP implementation of defeasible deontic logic. Künstliche Intelligenz **38**, 79–88 (2024). https://doi.org/10.1007/s13218-024-00854-9
17. Hage, J., Verheij, B.: The law as a dynamic interconnected system of states of affairs: a legal top ontology. International Journal of Human-Computer Studies **51**(6), 1043–1077 (1999). https://doi.org/10.1006/ijhc.1999.0297
18. Jones, G.T.: Dynamical jurisprudence: Law as a complex system. Georgia State University Law Review **24**(4) (2012)
19. Keshavarzi Zafarghandi, A., Verbrugge, R., Verheij, B.: Discussion games for preferred semantics of abstract dialectical frameworks. In: Kern-Isberner, G., Ognjanović, Z. (eds.) Symbolic and Quantitative Approaches to Reasoning with Uncertainty. vol. 11726, pp. 62–73. Springer (2019)
20. Kompa, N.: The role of vagueness and context sensitivity in legal interpretation. In: Keil, G., Poscher, R. (eds.) Vagueness and Law: Philosophical and Legal Perspectives. Oxford University Press (2016). https://doi.org/10.1093/acprof:oso/9780198782889.003.0010
21. Moreso, J.J.: Legal dynamics. In: Legal Indeterminacy and Constitutional Interpretation, Law and Philosophy Library, vol. 37, pp. 101–116. Springer, Dordrecht (1998). https://doi.org/10.1007/978-94-015-9123-2_4

22. Mumford, J., Atkinson, K., Bench-Capon, T.: Reasoning with Legal Cases: A Hybrid ADF-ML Approach (12 2022). https://doi.org/10.3233/FAIA220452
23. Poscher, R.: Ambiguity and vagueness in legal interpretation. In: Solan, L.M., Tiersma, P.M. (eds.) The Oxford Handbook of Language and Law. Oxford University Press (2012). https://doi.org/10.1093/oxfordhb/9780199572120.013.0010
24. Prakken, H., Sartor, G.: Law and logic: A review from an argumentation perspective. Artificial Intelligence **227**, 214–245 (2015). https://doi.org/10.1016/j.artint.2015.06.005
25. Priest, G.: Many-valued Logics, p. 456–475. Cambridge Introductions to Philosophy, Cambridge University Press (2008)
26. Royakkers, L.M.M.: Extending Deontic Logic for the Formalisation of Legal Rules, Law and Philosophy Library, vol. 36. Springer Science+Business Media Dordrecht, Dordrecht, 1 edn. (1998). https://doi.org/10.1007/978-94-015-9099-0
27. Rządowe Centrum Legislacji: Rola funktorów prawdziwościowych w prawidłowym konstruowaniu przepisów [the role of truth-functional operators in the proper drafting of legal provisions]. Webinarium [Webinar] (Aug 2025), accessed: 7 November 2025
28. Satoh, K., Asai, K., Kogawa, T., Kubota, M., Nakamura, M., Nishigai, Y., Shirakawa, K., Takano, C.: Proleg: An implementation of the presupposed ultimate fact theory of japanese civil code by prolog technology. In: Onada, T., Bekki, D., McCready, E. (eds.) New Frontiers in Artificial Intelligence. pp. 153–164. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
29. Schafer, B.: Formalising law, or the return of the golem. In: Research Handbook on Law and Technology, chap. 5. Edward Elgar Publishing, Cheltenham, UK (2023). https://doi.org/10.4337/9781803921327.00012, retrieved November 4, 2025
30. Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T.: The British Nationality Act as a logic program. Communications of the ACM **29**(5), 370–386 (May 1986). https://doi.org/10.1145/5689.5920
31. Steging, C., van Leeuwen, L.: A hybrid approach to legal textual entailment. In: Eighteenth International Workshop on Juris-Informatics (JURISIN 2024). Hamamatsu, Japan (5 2024)
32. Verheij, B.: Formalizing arguments, rules and cases. In: ICAIL '17: Proceedings of the Sixteenth International Conference for Artificial Intelligence and Law. pp. 199–208. ACM, New York, London, United Kingdom (6 2017)
33. Zin, M.M., Satoh, K., Borges, G.: Leveraging llm for identification and extraction of normative statements. In: Legal Knowledge and Information Systems, pp. 215–225. IOS Press (2024)

Legal NER: Evaluating the impact of LLM-Generated Annotations on NER Performance for Administrative Decisions

Harry Nan[1][0009−0003−8291−8818], Samaneh Khoshrou[2][0000−0002−6317−9453], and Johan Wolswinkel[1][0000−0001−8404−5027]

[1] Tilburg University, Tilburg Law School, the Netherlands
[2] Department of Intelligent Systems, Tilburg School of Humanities and Digital Sciences

Abstract. Named Entity Recognition (NER) is a widely adopted task for Information Extraction (IE), but requires high-quality annotations, which are costly and time-consuming to obtain. Large Language Models (LLMs) offer a potential solution by generating pseudo-annotations, yet their reliability in domain-specific legal contexts remains underexplored. This study investigates the use of LLM-generated annotations to expand the training set for supervised NER models. To achieve this, a dataset of legal entities is constructed annotated by human experts on a low-resource domain and language, namely Dutch administrative decisions. LLM-generated annotations are generated using a schema-driven few-shot prompt and evaluated against the human-labeled dataset. Subsequently, two different NER models are trained using three different sets of training data: human-annotated only, LLM-annotated only, and a finetuned LLM-annotated NER model with human-annotated data. The results show that LLM annotations are promising for legal entities that can be defined explicitly, but are unreliable for legal entities that require a deeper contextual understanding than what is explicitly stated in the text or in the prompt. However, finetuning a NER model trained on LLM-generated labels with human annotations slightly outperforms models trained on human-annotated data only. Our findings highlight the potential of hybrid supervision strategies to scale low-resource legal NER tasks while maintaining human-level accuracy.

Keywords: Named Entity Recognition· Information Extraction· Relationship Extraction · Large Language Models· Annotation

1 Introduction

Named Entity Recognition (NER) is considered a foundational technique for Information Extraction (IE) [18,20]. By identifying and classifying relevant entities (e.g. people or dates), NER allows for a structured representation of unstructured texts. While Large Language Models (LLMs) have already shown promising results for various tasks in Natural Language Processing (NLP), such as text classification and question answering, their application on NER and IE tasks re-

mains limited, with supervised deep learning methods remaining the most widely adopted methods for domain-specific tasks [14, 20, 23].

Legal Entity Recognition (LER) specifically deals with the extraction of relevant 'legal entities' from legal texts. Unlike domain-agnostic NER tasks that mostly focus on isolated entities (e.g. names, dates, or locations), these legal entities are determined by the (domain-specific) relationships that characterize the specific role of an entity. LER is therefore not just a task of identifying (domain-agnostic) entities (e.g. people), but also of selecting between those entities by assigning specific capacities (e.g. the person receiving a decision).

Considering legal information extraction in general, transformer-based architectures have already shown strong multilingual performance for NER [7, 31, 32]. The success of these supervised NER models, however, typically depends on large volumes of high-quality annotated data [30]. In many low-resource domains, the acquisition of such annotations is both costly and labor-intensive. Recent studies have therefore explored the use of LLMs to generate pseudo-annotations for NER tasks [3, 17, 28]. However, since these studies focus primarily on general-purpose or high-resource NER datasets, they provide limited guidance on how to handle domain-specific structures and relationships between entities.

In this work, we make two main contributions to the field of legal NLP and legal information extraction in particular. First, we introduce a new annotated dataset specifically designed for entity extraction in the low-resource domain of Dutch administrative decisions.[3] This dataset provides a valuable benchmark for research in information extraction from administrative and legal texts, and addresses a notable gap in available Dutch-language resources.

Second, we investigate the potential of LLMs to generate pseudo-annotations that can augment and enhance the performance of supervised NER models. Through a systematic evaluation, we assess both the accuracy of LLM-generated pseudo-annotations and the resulting improvements in model performance when these annotations are incorporated into training data. This dual focus offers insights into how LLMs can be effectively leveraged to scale low-resource datasets and to advance hybrid annotation strategies that combine human expertise with automated labeling.

Specifically, this study addresses two research questions:

- *To what extent can LLMs generate accurate pseudo-annotations for unsupervised NER, compared to human-labeled data?*
- *To what extent does the inclusion of LLM-generated pseudo-annotations improve the performance of supervised NER models?*

This study shows that LLMs have the potential to generate a large but unreliable set of pseudo-annotations, and the quality of these annotations declines when labeling requires specialized legal domain knowledge or deeper contextual understanding. However, augmenting human annotations with LLM-generated annotations can increase the performance of NER tasks, as it can expand training coverage by allowing for an effective way of scaling, while preserving the accuracy and reliability of the human-annotated dataset.

2 Related Work

NER has been considered a core task in structuring legal texts, enabling applications such as case retrieval, legal analytics, and argument mining [14, 20].

[3] GitHub link to the dataset

NER methods, particularly transformer-based methods, have been increasingly applied to low-resource legal domains and underrepresented languages, including languages such as Portuguese [31], German [7], and Turkish [32], where annotated corpora are scarce. Nonetheless, the limited amount of data across multilingual and domain-specific corpora remains challenging for IE-tasks such as NER [1]. This holds even more for the related task of Relationship Extraction (RE), which aims to extract relationships between identified entities [1, 24].

Recent work has therefore focused on developing few-shot [28], weakly supervised [17], or limited supervision techniques [26] to reduce the reliance on large annotated datasets while still enabling supervised models to achieve strong and reliable performance [20, 23]. To counter this bottleneck of scarce annotated datasets, LLMs have been used to reduce the required work needed to obtain high-quality annotated data. Some studies have applied LLMs to assist legal researchers by improving annotation efficiency and annotator agreement [11], or apply hybrid methods to perform IE-tasks directly [16]. Other studies have shown that LLM-generated annotations, when combined with human annotations, can achieve human-level performance in domain-specific settings [3, 17, 24].

Ensuring accuracy and reliability in LLMs requires more than sheer scale, as it depends critically on the quality of supervision. Recent studies show that models fine-tuned on human-curated datasets consistently achieve superior, or at least comparable, performance to those trained solely on LLM-generated labels [15]. Moreover, even a relatively small proportion of human-labeled examples can improve accuracy and reasoning ability of LLMs [2]. Human-labeled data are therefore particularly effective in improving correctness, depth of reasoning, and robustness to ambiguity, while synthetic data such as pseudo-annotations provide unmatched scalability [12]. Consequently, the most effective strategy is a hybrid one, where human annotations ensure quality and reliability, and synthetic data is employed to broaden coverage in a cost-efficient manner.

A common approach to generating annotations through LLMs is to frame the task as a generative problem, often through question-answering (QA) techniques [23, 25]. Research shows that explicit label definitions, fixed output formats and decomposed QA strategies improve NER performance [25]. Additionally, integrating entity extraction with relation extraction has been found to be effective [24]. Few- and zero-shot prompting techniques have also been applied to NER [25], showing that LLMs can achieve reliable entity labels in low-resource settings. However, most research focuses on general-purpose datasets and largely ignores domain-specific contexts, relationships between entities, and the impact of pseudo-annotations on supervised NER model performance, as these are often used to replace human annotations [1, 17, 20]. But these relationships are especially of importance in the legal domain, where legal entities derive their meaning from their specific roles and the relationships that define them.

In the legal domain, texts contain domain-specific characteristics such as specialized vocabulary, formal syntax, and structured relationships between entities [29]. Adapting transformer-based NER models, such as BERT, through domain-specific pre-training has been shown to enhance performance in both le-

gal classification and LER tasks [6]. To further incorporate domain-specific information, some studies have embedded entity relationships directly into prompts, improving LLM's performance on NER tasks [24].

In addition, joint NER–RE approaches have been explored. These approaches combine NER with an RE task, punishing the model for incorrectly identifying entities or relationships between these entities. Despite the more reliable results of these joint approaches for RE [9], their effect on NER performance remains less conclusive: while some studies report modest improvements, particularly in domains with structural dependencies [21, 22], others find little or no advantage over baseline NER models [9]. This inconsistency highlights a critical gap in understanding how domain-specific structures and relationships can be effectively leveraged to advance NER in the legal domain.

In summary, prior research highlights two key challenges: (1) supervised methods remain dominant for NER but are costly to apply in low-resource legal domains; and (2) while LLMs offer promising strategies for reducing annotation burden, most work has focused so far on general-purpose datasets rather than domain-specific (legal) corpora. These gaps underscore the need to investigate whether LLM-generated pseudo-annotations, enriched with domain-specific relationships, can improve LER performance in legal texts. Our work addresses these both challenges by evaluating LLM-generated annotations in a low-resource domain and language, namely a wide variety of Dutch administrative decisions, and examining the impact of incorporating LLM-generated annotations into two different supervised NER models.

3 Methodology

Figure 1 shows an overview of the methodology of this paper. Firstly, as described in section 3.1, a Dutch dataset is created by collecting documents and applying sentence extraction techniques, whereas a list of relevant legal entities to be extracted is also defined. Secondly, section 3.2 describes the annotation process, which resulted in the gold data. Section 3.3 describes how LLMs are applied to scale up annotated data (LLM annotation), generating silver data, and section 3.4 describes the creation of the NER(-RE) models, and the experimental setup to understand how LLM-generated annotations influence NER performance.

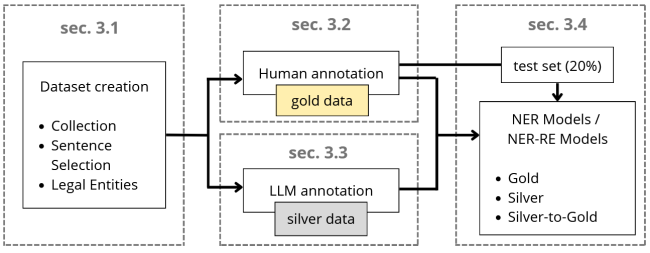


Fig. 1: Flowchart of the methodology.

3.1 Dataset creation

An administrative decision is an order which is not of a general nature, including rejection of an application for such an order (*Algemene wet bestuursrecht*, art. 1:3(2)). A similar definition is provided in the Model Rules, although the latter is more explicit about the constitutive elements of an administrative decision. In general, an administrative decision is understood as a specific type of administrative action addressed to one or more individualized persons and adopted unilaterally by a public authority to determine one or more concrete cases with a legally binding effect [19]. From these legal definitions, five key entities of administrative decisions can be derived which are useful for downstream IE-tasks, as these five legal entities should be present in every administrative decision, irrespective of its substance. First, two types of legal subjects [13] are involved in every administrative decision: (1) a person receiving the decision ('recipient'), and (2) a public authority issuing the decision ('authority'). Next, every decision has a legally binding effect, thus conferring certain rights or obligations to the recipient (legal act [13]). In particular, such a legal effect consists of the combination of (3) a legal action (e.g., to grant or to revoke) related to (4) a legal object (e.g., a license or a fine). Finally, since an administrative decision is taken unilaterally, it should always be based on (5) a legal basis (competence [13]): a statutory provision that authorizes the authority to act. While other legal entities can also be present in certain types of administrative decisions (such as the type of misconduct in case of a enforcement decision), these five entities should be present in all types of administrative decisions. Thus, extracting these five legal entities is essential in terms of generalizability.

To answer the research questions and to ensure generalizable results, a highly diverse set of administrative decisions (such as permits and sanctions) from seven different public authorities in the Netherlands is used. All these decisions are publicly accessible. In total, 20,798 administrative decisions (available in PDF format) were scraped from the different websites of these public authorities and were saved in txt-format using pdfplumber where possible. See Table 1 for dataset statistics from the seven different authorities.

Legal documents such as administrative decisions can be quite long [29], but the five legal entities identified above are usually present together in only a few individual sentences in the text. To restrict the task of legal entity recognition to these few relevant sentences, rule-based methods have been applied. Based on Dutch law (*Algemene wet bestuursrecht*) and a qualitative analysis of the data, combinations of legal action (verbs) and legal object (nouns) pairs have been defined that consistently and almost exhaustively express the legal effect of a decision (for example: *grant* and *permit*)[4]. Next, the texts were split into sentences using SpaCy[5], after which regular expressions were applied to identify whether one of these pairs of legal action and legal object is present in the sentence. If such a pair was identified, this sentence was stored for subsequent

[4] See GitHub for the action–object pairs and their derivation from the *Algemene wet bestuursrecht*.

[5] SpaCy's pretrained model nl_core_news_lg

processing as it might contain one or more of the five legal entities defined above. A total of 14,793 documents were found that contained this sentence pattern at least once, which resulted in a total of 38,766 sentences found. See Table 1. Documents in which no such sentence pattern was found, were not machine readable, or were not administrative decisions.

Table 1: Overview of sentence extraction and characteristics across authorities.

Authority	Found Documents	Docs With Sentences	Total Sentences	Avg Sent./ Doc (All)	Avg Sent./ Doc (With Sent.)	Mean Tok. Sent. Len	Median Tok. Sent. Len	Min / Max Tok. Sent. Len
ACM	4,725	1,764 (37.3%)	10,767	1.19	6.01	31.70	27.0	7 / **192**
ANVS	**10,948**	9,975 (91.1%)	**16,703**	1.53	1.67	29.53	29.0	**4** / 80
DNB	729	317 (43.5%)	2,589	2.85	**8.17**	31.68	30.0	8 / 75
Gemeente Rotterdam	3,791	2,261 (59.6%)	6,702	1.65	2.96	**35.17**	**38.0**	18 / 44
KSA	390	239 (61.3%)	1,524	**3.90**	6.38	29.85	28.0	5 / 76
Provincie Drenthe	74	72 (97.3%)	203	2.74	2.82	16.49	18.0	6 / 35
Rijksoverheid	141	**138 (97.9%)**	278	1.97	2.01	23.39	21.0	7 / 65
Total	20,798	14,793 (71.1%)	38,766	1.82	4.00	28.14	25.0	4 / 192

3.2 Data annotation

To ensure a diverse set of annotated data, a selection of up to 150 sentences was made for each public authority. For one specific authority (ACM), even 300 sentences were extracted since its decisions were rich in variety (150 sentences from enforcement decisions and 150 sentences from permit decisions), whereas for another authority (Gemeente Rotterdam) only 60 sentences were selected due to their poor variety. This resulted in a total of 1,101 sentences. After selecting these sentences, two law students (Bachelors and Masters) annotated the five legal entities in these sentences (recipient, authority, legal basis, legal object, and legal action) using Lawnotation. The annotations were based on an annotation protocol[6] drafted for this project and tested beforehand. One third of the sentences were annotated by both annotators to measure annotator agreement between the two annotators, which resulted in an overall Cohen's Kappa score of 0.47 (with a raw agreement of 0.78 (recipient), 0.58 (authority), 0.51 (legal object), 0.46 (legal action), 0.39 (legal basis) for the individual features), indicating moderate agreement. Most disagreements were due to boundary mismatches and cases where one annotator labeled a token as belonging to a feature while the other left it unannotated. The moderate agreement can largely be attributed to ambiguities in determining whether a sentence expressed the actual operative part of a decision or merely referred to a hypothetical, past, or future decision. This indicates limitations in the sentence selection techniques used. Additionally, annotators often differed in judging whether such sentences represented a binding legal effect or merely explanatory context.

To ensure consistency of the annotated data, a legal expert validated all annotations and adjusted the annotations where necessary (e.g., where disagreement between annotators was found), thus acting as an (expert) reviewer [4]. The resulting set of annotated data was split in two subsets: 20% of the data served as test set, while the other 80% served as training data for the NER model. Table 2 shows the results thereof, demonstrating the amount of entities found per authority, and the average length of each type of entity in tokens.

[6] GitHub link to annotation protocol

Table 2: Entity counts and mean token length per authority and entity type

Authority	Recipient	Authority	Legal Object	Legal Action	Legal Basis
ACM (293)	76 / 3.00	105 / 2.96	99 / 1.25	101 / 1.00	58 / 8.24
ANVS (149)	98 / 4.39	3 / 1.00	112 / 1.00	112 / 1.01	4 / 14.00
DNB (149)	31 / 1.55	28 / 1.18	32 / 1.88	33 / 1.00	14 / 6.79
G. R'dam (60)	36 / 1.00	36 / 1.00	41 / 1.00	41 / 1.00	30 / 8.53
KSA (150)	28 / 2.25	46 / 3.96	37 / 2.16	37 / 1.00	25 / 6.52
P. Drenthe (150)	71 / 1.00	80 / 1.05	82 / 1.01	82 / 1.00	3 / 5.67
Rijksoverheid (150)	50 / 1.06	64 / 1.09	84 / 1.04	88 / 1.00	22 / 11.32
Overall (1,101)	**390 / 2.38**	**362 / 1.99**	**487 / 1.21**	**494 / 1.00**	**156 / 8.42**

3.3 Pseudo-labeling using GPT

GPT-5 was used for this study to expand the training data for the NER model. Based on recent advances in LLM-based NER, a schema-driven few-shot prompt[7] was designed. The prompt is based on recent work for prompt engineering, which has reconceptualized NER tasks for LLMs as a prompting task. For example, schema-driven instructions are used to guide the LLM to generate entities in a structured manner. Ideas from previous studies (see section 2) are implemented in the prompt by treating the NER task as a structured generation task aligned with domain-specific definitions, as seen in Table 3. The LLM-generated annotations were evaluated using Precision (P), Recall (R), and F1, by comparing it to the human-annotated dataset (n=1,101). Additionally, a total of 7,228 additional sentences were annotated using the LLM. Table 3 gives an overview of the prompt structure.

Table 3: Structured overview of the prompt.

Prompt structure	Implementation	Idea
Introduction	Introduce role of LLM, and frame the task as a QA-task.	Generative task for entity extraction [25].
Entity descriptions	Entities are defined individually in a schema-driven way. A legal explanation, with possible relations to other entities is included, with entity examples.	Schema-driven prompting [23] and highlighting entity relations [24].
Normalization	Explaining rules for normalization (deduplication, abstain, span/boundary), and show expected output with an example JSON structure.	Normalization and fixed output [25].
Examples	Provide three real examples.	Few-shot prompting.

3.4 Experimental setup

This study trains two different NER models on three different subsets of data, which results in six different approaches, as shown in Figure 1. A random 20% of the human-annotated data was held out as a test set to evaluate each model. To prevent data leakage, any overlap in sentences between the training and test sets were removed from the test set, resulting in 184 sentences in the test set.

First, a token-level BERT NER model based on the Dutch BERT variant (BERTje) [8] was trained on three data subsets. BERTje was chosen because its Dutch-specific pretraining achieves state-of-the-art results on Dutch NLP tasks, making it a reliable choice for token-level NER in Dutch legal texts [8]. The model assigns an entity label to each token using its context-aware representation. A

[7] GitHub link to full prompt

Gold model was trained on the remaining 80% (n=883) of the human-annotated sentences, of which 20% was used as a validation set. In parallel, a Silver model was trained on the LLM-generated annotations (n=7,228), of which 20% was used for validation. Finally, a Silver-to-Gold model was created, by finetuning the Silver model with human-annotated data, thereby continuing training on human-annotated data. The dataset was tokenized with the BERTje tokenizer [8].

Next, a more sophisticated NER model was created by combining NER with RE. To facilitate this, inspired by spERT [9], a span-based joint NER-RE model was trained, based on BERTje [8]. This model represents candidate spans with contextual embeddings, labels them as entities, and simultaneously predicts relationships between spans. Since relationships between entities were not annotated explicitly, relationships were constructed automatically by pairing entities according to a pre-set schema. Four relationship types were defined for this project based on the legal definitions of administrative decisions (as described in section 3.1): (1) *recipient* receives *legal object*, (2) *legal action* concerns *legal object*, (3) *legal basis* authorizes *authority*, and (4) *authority* performs *legal action*. See Figure 2 for an example of the NER and RE task. The model was trained with a combined loss function, combining the entity classification loss with the relationship classification loss. Three datasets were used to train a Gold, Silver and Silver-to-Gold model, identically to the token-level NER model.

Fig. 2: Example of the NER (highlighted) and RE (red arrows).

All six approaches were evaluated using token-level annotations from the held-out golden test set (n=184). Precision (P), Recall (R), and F1 were computed at the token level. The evaluation followed a strict matching criterion, in which contiguous tokens forming a single entity span were treated as one unit; partial overlaps between predicted and gold spans were therefore considered incorrect.

4 Results

4.1 LLM pseudo-annotation performance

Table 4 shows the quality of the pseudo-generated annotations by GPT-5, by comparing LLM-generated annotations with human annotations (n=1,101). High F1-scores can be observed for the entities of 'authority' (F1=0.91) and 'legal object' (F1=0.85). However, for authorities that impose sanctions (ACM, DNB, KSA), these scores are lower. This can be explained by the more complex nature of sanctioning decisions, which often contain lengthy reasoning before the relevant operative clause (see also Table 1). In contrast to entitlement decisions (such as a license or permit), enforcement decisions (such as a fine) often display the legal object with an adjectival premodifier, an adjective that appears before

a noun to modify it (such as: *administrative* fine, which is also seen in Table 2), which the LLM failed to capture, suggesting systematic boundary annotation errors. Nonetheless, the high F1-scores for 'authority' and 'legal object' can be explained by the uniform way in which these entities are defined in legislation, and because the form of these features is relatively consistent, making it easy for LLMs to capture these.

Table 4: LLM pseudo-label quality per-authority Precision (P), Recall (R), F1, and total TP + FP + FN count (#) per entity.

	Recipient				**Authority**				**Legal Object**				**Legal Action**				**Legal Basis**			
	P	R	F1	#	P	R	F1	#	P	R	F1	#	P	R	F1	#	P	R	F1	#
ACM	.05	.05	.05	150	.51	.74	.60	178	.36	.90	.51	261	.38	.75	.51	223	.00	.00	.00	59
ANVS	.07	.09	.08	213	.60	1.0	.75	5	.80	.98	.88	140	.71	.60	.65	140	.00	.00	.00	6
DNB	.17	.07	.09	41	.29	.96	.45	93	.01	.03	.01	138	.27	.82	.41	105	.55	.43	.48	19
G. R'dam	.69	1.0	.82	52	.66	1.0	.79	55	.68	1.0	.81	60	1.0	.56	.72	41	1.0	.67	.80	30
KSA	.13	.11	.12	48	.36	.94	.52	12	.23	.81	.36	139	.26	.76	.39	116	.80	.48	.60	28
P. Dre.	.96	1.0	.98	74	.94	.99	.96	85	.46	.96	.62	176	.41	.94	.57	194	.00	.00	.00	3
Rijks.	.77	.94	.85	64	.80	.95	.87	79	.67	.98	.80	124	.33	.30	.31	141	1.0	.05	.09	22
Combined	**.49**	**.45**	**.47**	561	**.90**	**.93**	**.91**	387	**.80**	**.90**	**.85**	588	**.73**	**.67**	**.70**	605	**.85**	**.29**	**.43**	142

The model shows moderate performance for the entity of 'legal action' (F1=0.70), which, despite being formally defined in legislation and expert practice, exhibit considerable variation in form and often depend on the procedural context. Additionally, verbs that do not capture the specific type of legal action at issue (such as 'deciding') were wrongly captured by the LLM, indicating shortcomings in the understanding of the entity 'legal action' by LLMs.

In contrast, the entities of 'recipient' and 'legal basis' yield lower scores (recipient: F1 = 0.47; legal basis: F1 = 0.43). These entities are typically longer, more heterogeneous in form, and more complex from a legal-linguistic perspective, which makes them difficult for the LLM to identify consistently. Performance for 'recipient' is higher among authorities that frequently issue their decisions in a letter format (e.g., Provincie Drenthe, Rijksoverheid, Gemeente Rotterdam), where the recipient is often expressed through short and direct referential tokens such as 'you', as also reflected in Table 2. Interestingly, precision scores for legal basis are relatively high (P=0.85) compared to its recall (R=0.29), suggesting that while LLMs can correctly capture clearly identifiable patterns, they fail to identify many less explicit instances or relevant legal provisions that it has not seen before. This highlights a limitation of LLMs: while they are consistent in recognizing clear patterns and expressions, they may struggle with entities that they have not seen before in the prompt, require contextual inference, or require common-sense understanding beyond what is written.

4.2 NER(-RE) performance

Table 5 shows the results of the token-based NER and span-based joint NER-RE models, with precision, recall and F1-scores for each of the three subsets of annotated data: Gold (human annotations only), Silver (LLM annotations only), and Silver-to-Gold (Silver model finetuned on human annotations).[8]

[8] GitHub link to more detailed results per authority.

Table 5: Per-label performance (P, R, F1) of three approaches for both BERT-based NER and spERT-based joint NER-RE models.

Label	**NER**									**Joint NER-RE**								
	Gold			**Silver**			**S.-to-Gold**			**Gold**			**Silver**			**S.-to-Gold**		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Recipient	.77	.86	.82	.22	.34	.26	.83	.92	.87	1.0	.97	.98	.95	.32	.48	1.0	.98	.99
Authority	.80	.90	.85	.58	.84	.69	.83	.90	.86	.98	.93	.95	.98	.91	.95	1.0	.97	.99
Legal object	.97	.85	.91	.43	.85	.57	.95	.88	.91	.96	.94	.95	.99	.85	.91	.99	.98	.98
Legal action	.96	.89	.92	.41	.61	.49	.96	.89	.92	.96	.94	.95	.96	.92	.94	1.0	1.0	1.0
Legal basis	.76	.91	.83	.53	.25	.34	.83	.94	.88	.94	.91	.92	1.0	.41	.58	1.0	.94	.97
Micro avg	.87	.88	.87	.41	.63	.50	.89	.90	.89	.97	.94	.95	.98	.74	.84	1.00	.98	.99

These results show that both Gold models achieve high micro average F1-scores (NER: 0.87; NER-RE: 0.95), providing a strong baseline and demonstrating good performance on human annotations. In contrast, the Silver models, trained exclusively on LLM-generated pseudo-labels, perform worse (NER: 0.50; NER-RE: 0.84), largely due to inconsistencies in LLM annotations for complex or implicit entities such as 'recipient' and 'legal basis'. Importantly, when the Silver models are subsequently fine-tuned with human annotations (Silver-to-Gold), their performance surpasses the gold baselines, improving the micro average F1-scores by 0.02 and 0.04 respectively for the NER and NER-RE models. This finding indicates that combining imperfect but large-scale LLM-generated annotations with smaller sets of human annotations can effectively expand coverage and improve generalization in low-resource legal NER tasks.

Across datasets of different public authorities, performance remains highest for legal entities with standardized and explicit phrasing, such as public authority, legal object, and legal action, while entities that are more context-dependent or variable in expression, particularly recipient and legal basis, remain challenging. Similarly as described in section 4.1, standardized decisions (e.g., letters) achieve better results compared to decisions that are less standardized. These observations confirm that model performance depends strongly on both the linguistic structure of the texts and the degree of standardization in the way how an administrative decision is drafted.

Interestingly, the Silver NER–RE model performs better than expected given the quality of its LLM-generated training labels (Table 4). This indicates that the model may be able to generalize beyond the noise present in the pseudo-labels. The joint NER–RE architecture likely encourages structurally coherent predictions by linking entities through predefined relationships. In addition, the large volume of pseudo-labeled data increases the model's exposure to diverse linguistic contexts, effectively broadening its training coverage. These aspects suggest that the model benefits from implicit regularization and denoising effects, which may explain the high precision and F1-scores observed despite the noisy supervision.

5 Discussion

5.1 Pseudo-annotations by LLM

As seen in Table 4 and as demonstrated in section 4, LLMs are consistent in identifying features with clear patterns and little linguistic variety, but inconsistently capture features that have a lot of linguistic variety and may require more contextual understanding or common sense.

To illustrate this, legal entities such as the legal object and the public authority are defined explicitly in legislation and require specific domain knowledge. The (legally) formalized nature of administrative decisions means that there is little variation in terminology and syntactic structure for these 'juridicized' features. In contrast, features such as the recipient which are not as 'juridicized' and do not require specific domain knowledge (as any citizen can be the recipient of an administrative decision), score relatively low. Since the latter features rely more on common sense reasoning, LLMs struggle to identify these entities correctly, as these features require understanding of everyday knowledge or contextual inference beyond what is explicitly stated. However, when these entities are written concisely or uniformly, their scores improve.

These results are in line with other studies, as, for instance, LLMs have demonstrated limitations in abstract common-sense reasoning, often failing to grasp relationships or contexts that humans intuitively understand [10]. Additionally, the findings are also in line with other studies [5], where LLMs which are not domain-specific show potential when applying few-shotting approaches, but face challenges with boundary errors, false positives, and incomplete recall. The results show limitations of the LLM to handle contextual variations for certain features, suggesting that tailoring models to the legal domain or specific (legal) language may be necessary to achieve more reliable results. Furthermore, the achieved scores are highly dependent on the prompt, thereby confirming previous work that has indicated that improving the prompt could significantly increase performance [10]. All in all, LLMs can provide useful baseline annotations that reduce manual work, but expert correction remains essential, especially for ambiguous features.

5.2 Impact of pseudo-annotations on supervised NER

Table 5 presents the performance of both NER models across the three training setups. The Gold models, trained solely on human annotations, achieve strong results and form a reliable baseline, consistent with prior work showing the superiority of supervised methods over LLM-only approaches [20, 23]. As expected, the Silver models perform considerably worse due to noise in the LLM-generated labels. However, when the Silver models are fine-tuned on human annotations (Silver-to-Gold), their performance slightly exceeds that of the Gold models, indicating that weakly supervised LLM-generated annotations can broaden training coverage and improve generalization in low-resource LER tasks, albeit with modest gains.

Comparing the NER model with the more sophisticated joint NER-RE model, we observe that every approach of the joint model outperforms the token-based NER model on the same data. Although indicative, future research should investigate whether these gains stem specifically from the inclusion of relationship extraction, by comparing a span-based NER task with a similar span-based NER-RE task.

Compared to earlier studies, which found that weakly supervised models generally underperform fully supervised ones [15, 17], our findings show that finetuning a Silver model on Gold data can slightly surpass human-only baselines. Unlike prior work that combined human and weak labels simultaneously during training [17], our Silver-to-Gold approach separates the learning phases, first leveraging the scale of pseudo-labels, then refining with human annotations. The Silver-to-Gold model appears to generalize better, particularly by expanding coverage and improving representation of ambiguous cases. Nevertheless, inconsistencies in the pseudo-labels can still introduce noise, emphasizing the need for careful integration of LLM-generated data.

Qualitative analysis shows that improvements in the Silver-to-Gold models primarily stem from more accurate span boundaries and better coverage of alternative expressions for entities (e.g., abbreviations or referential phrases).

Several limitations should be noted. Some sentences in the human-annotated dataset were included despite lacking a legal effect, revealing shortcomings in the sentence selection process. Certain legal action–object pairs were overrepresented, encouraging models to overfit to common patterns. Moreover, entities expressed as pronouns in letter-style decisions create coreference issues. This study also did not evaluate alternative NER or LLM architectures.

Finally, the findings offer insights into transferability: domains with highly formalized or pattern-consistent language (e.g., biomedical, financial, or regulatory texts) may benefit similarly from pseudo-annotations, whereas domains with greater linguistic variability may experience reduced gains. Transfer to other languages likewise depends on the availability of strong monolingual pretrained models; languages lacking such resources may not replicate the improvements seen with BERTje. Nonetheless, the Silver-to-Gold strategy remains promising for low-resource NER more broadly.

Future research could explore structure-aware or visually rich document NER approaches [20, 27] to assess generalizability.

6 Conclusion

This study examined the use of LLM-generated pseudo-annotations to augment supervised NER models in a low-resource legal domain. GPT-5 can produce large-scale annotations, but with uneven quality: entities that are explicitly defined and consistently expressed (e.g., *authority*, *legal object*) are captured reliably, while longer or context-dependent entities (e.g., *recipient*, *legal basis*) remain challenging.

LLM-generated annotations were used to expand the training data for two NER models (a token-based model and a joint NER-RE model) and their impact was evaluated across different training settings. Although models trained solely on LLM-generated data perform worse, finetuning these models with a smaller set of human annotations yields performance that slightly surpasses models trained only on human-labeled data. This shows that weakly supervised LLM-generated annotations can broaden training coverage and improve generalization in low-resource legal NER tasks.

Overall, the findings demonstrate the value of hybrid annotation strategies that combine scalable LLM-generated labels with targeted human supervision, offering a practical approach for developing more scalable and accurate legal IE systems.

References

1. Abdullah, M.H.A., Aziz, N., Abdulkadir, S.J., Alhussian, H.S.A., Talpur, N.: Systematic literature review of information extraction from textual data: recent methods, applications, trends, and challenges. IEEE Access **11**, 10535–10562 (2023), doi: 10.1109/ACCESS.2023.3240898
2. Ashok, D., May, J.: A little human data goes a long way. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 381–413. Association for Computational Linguistics (2025), doi: 10.18653/v1/2025.acl-short.30
3. Bogdanov, S., Constantin, A., Bernard, T., Crabbé, B., Bernard, E.P.: NuNER: Entity recognition encoder pre-training via LLM-annotated data. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. pp. 11829–11841. Association for Computational Linguistics (2024), doi: 10.18653/v1/2024.emnlp-main.660
4. Braun, D.: I beg to differ: how disagreement is handled in the annotation of legal machine learning data sets. Artificial intelligence and law **32**(3), 839–862 (2024), doi: 10.1007/s10506-023-09369-4
5. Breton, J., Billami, M.M., Chevalier, M., Nguyen, H.T., Satoh, K., Trojahn, C., Zin, M.M.: Leveraging LLMs for legal terms extraction with limited annotated data. Artificial Intelligence and Law pp. 1–27 (2025), doi: 10.1007/s10506-025-09448-8
6. Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., Androutsopoulos, I.: LEGAL-BERT: The muppets straight out of law school. In: Findings of the Association for Computational Linguistics: EMNLP 2020. pp. 2898–2904. Association for Computational Linguistics (2020), doi: 10.18653/v1/2020.findings-emnlp.261
7. Darji, H., Mitrović, J., Granitzer, M.: German BERT model for legal named entity recognition. In: Proceedings of the 15th International Conference on Agents and Artificial Intelligence. vol. 3, p. 723–728. SciTePress (2023), doi: 10.5220/0011749400003393
8. De Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., Nissim, M.: BERTje: A dutch BERT model. arXiv preprint arXiv:1912.09582 (2019), doi: 10.48550/arXiv.1912.09582
9. Eberts, M., Ulges, A.: Span-based joint entity and relation extraction with transformer pre-training. arXiv preprint arXiv:1909.07755 (2019), doi: 10.3233/FAIA200321
10. Gawin, C., Sun, Y., Kejriwal, M.: Navigating semantic relations: Challenges for language models in abstract common-sense reasoning. In: Companion Proceedings of the ACM on Web Conference 2025. pp. 971–975 (2025), doi: 10.1145/3701716.3715472
11. Gray, M., Savelka, J., Oliver, W., Ashley, K.: Can GPT alleviate the burden of annotation? In: Legal Knowledge and Information Systems, pp. 157–166. IOS Press (2023), doi: 10.3233/FAIA230961
12. Guo, X., Chen, Y.: Generative ai for synthetic data generation: Methods, challenges and the future. arXiv preprint arXiv:2403.04190 (2024), doi: 10.48550/arXiv.2403.04190
13. Hage, J.: Basic concepts of law. In: Introduction to law, pp. 33–52. Springer (2017), doi: 10.1007/978-3-319-57252-9_3
14. Keraghel, I., Morbieu, S., Nadif, M.: Recent advances in named entity recognition: A comprehensive survey and comparative study. arXiv preprint arXiv:2401.10825 (2024), doi: 10.48550/arXiv.2401.10825
15. Møller, A.G., Pera, A., Dalsgaard, J., Aiello, L.: The parrot dilemma: Human-labeled vs. LLM-augmented data in classification tasks. In: Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 179–192. Association for Computational Linguistics (2024), doi: 10.18653/v1/2024.eacl-short.17

16. Nan, H., Marx, M., Wolswinkel, J.: Combining rule-based and machine learning methods for efficient information extraction from enforcement decisions. In: Legal Knowledge and Information Systems, pp. 321–326. IOS Press (2024), doi: 10.3233/FAIA241262
17. Oliveira, V., Nogueira, G., Faleiros, T., Marcacini, R.: Combining prompt-based language models and weak supervision for labeling named entity recognition on legal documents. Artificial Intelligence and Law **33**(2), 361–381 (2025), doi: 10.1007/s10506-023-09388-1
18. Premasiri, D., Ranasinghe, T., Mitkov, R., El-Haj, M., Frommholz, I.: Survey on legal information extraction: current status and open challenges: D. premasiri et al. Knowledge and Information Systems pp. 1–72 (2025), doi: 10.1007/s10115-025-02600-5
19. Research Network on European Administrative Law: ReNEUAL model rules on EU administrative procedure (2014), https://www.reneual.eu/images/Home/ReNEUAL-Model_Rules-Compilation_BooksI_VI_2014-09-03.pdf
20. Seow, W.L., Chaturvedi, I., Hogarth, A., Mao, R., Cambria, E.: A review of named entity recognition: from learning methods to modelling paradigms and tasks. Artificial Intelligence Review **58**(10), 1–87 (2025), doi: 10.1007/s10462-025-11321-8
21. Sui, D., Zeng, X., Chen, Y., Liu, K., Zhao, J.: Joint entity and relation extraction with set prediction networks. IEEE transactions on neural networks and learning systems **35**(9), 12784–12795 (2023), doi: 10.1109/TNNLS.2023.3264735
22. Wadden, D., Wennberg, U., Luan, Y., Hajishirzi, H.: Entity, relation, and event extraction with contextualized span representations. arXiv preprint arXiv:1909.03546 (2019), doi: 10.48550/arXiv.1909.03546
23. Wang, S., Sun, X., Li, X., Ouyang, R., Wu, F., Zhang, T., Li, J., Wang, G., Guo, C.: GPT-NER: Named entity recognition via large language models. In: Findings of the Association for Computational Linguistics: NAACL 2025. pp. 4257–4275. Association for Computational Linguistics (2025), doi: 10.18653/v1/2025.findings-naacl.239
24. Xiao, L., Xu, Y., Zhao, J.: LLM-DER: A named entity recognition method based on large language models for chinese coal chemical domain. arXiv preprint arXiv:2409.10077 (2024), doi: 10.48550/arXiv.2409.10077
25. Xie, T., Li, Q., Zhang, J., Zhang, Y., Liu, Z., Wang, H.: Empirical study of zero-shot ner with chatgpt. arXiv preprint arXiv:2310.10035 (2023), doi: 10.48550/arXiv.2310.10035
26. Xie, T., Li, Q., Zhang, Y., Liu, Z., Wang, H.: Self-improving for zero-shot named entity recognition with large language models. arXiv preprint arXiv:2311.08921 (2023), doi: 10.48550/arXiv.2311.08921
27. Yang, H.W., Agrawal, A.: Extracting complex named entities in legal documents via weakly supervised object detection. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 3349–3353 (2023), doi: 10.1145/3539618.3591852
28. Ye, J., Xu, N., Wang, Y., Zhou, J., Zhang, Q., Gui, T., Huang, X.: LLM-DA: Data augmentation via large language models for few-shot named entity recognition. arXiv preprint arXiv:2402.14568 (2024), doi: 10.48550/arXiv.2402.14568
29. Zadgaonkar, A.V., Agrawal, A.J.: An overview of information extraction techniques for legal document analysis and processing. International Journal of Electrical & Computer Engineering (2088-8708) **11**(6) (2021), doi: 10.11591/ijece.v11i6.pp5450-5457
30. Zhuang, B., Liu, J., Pan, Z., He, H., Weng, Y., Shen, C.: A survey on efficient training of transformers. arXiv preprint arXiv:2302.01107 (2023), doi: 10.48550/arXiv.2302.01107
31. Zorzanelli Costa, M., Faria Robson, D., Vieira, T.B.P., Bourguet, J.R., Guizzardi, G., Almeida, J.P.A.: Automated semantic annotation pipeline for brazilian judicial decisions. In: Legal Knowledge and Information Systems. vol. 395, pp. 226–238. IOS (2024), doi: 10.3233/FAIA241248
32. Çetindağ, C., Yazıcıoğlu, B., Koç, A.: Named-entity recognition in turkish legal texts. Natural Language Engineering **29**(3), 615–642 (2023), doi: 10.1017/S1351324922000304

Structured Four-Stage Legal Translation:
From Natural-Language Traffic Rules to PROLOG

May Myo Zin[1 [0000−0003−1315−7704]], Wachara Fungwacharakorn[1 [0000−0001−9294−3118]], Ken Satoh[1 [0000−0002−9309−4602]], and Katsumi Nitta[1 [0000−0002−9018−8603]]

Center for Juris-Informatics, ROIS-DS, Tokyo, Japan
{maymyozin,wacharaf,ksatoh}@nii.ac.jp, knitta00@gmail.com

Abstract. Traffic regulations are written for human interpretation and therefore rely on shared background knowledge and flexible phrasing, which inherently introduce ambiguity, context dependence, and semantic underspecification. These linguistic characteristics conflict with the precision required by computational reasoning engines such as Prolog, which demand explicit logical structure. This study evaluates two baseline translation approaches, Natural Language to Prolog ($NL \rightarrow Prolog$) and Logical English to Prolog ($LE \rightarrow Prolog$), and introduces a new reasoning-guided translation framework called Structured Four-Stage Legal Translation ($S4L \rightarrow Prolog$). The proposed S4L framework performs semantic role extraction, scene completion, logical mapping, and Prolog rule generation within a single guided prompt, enabling direct translation of raw traffic rules into executable logic without human intervention. A benchmark consisting of twenty real-world traffic rules was used to evaluate each approach in terms of syntactic validity, semantic correctness, and logical completeness. $S4L \rightarrow Prolog$ achieves the highest accuracy, correctly formalizing 75 percent of the rules, while $NL \rightarrow Prolog$ reaches 60 percent and $LE \rightarrow Prolog$ reaches 55 percent. Qualitative analysis further shows that S4L captures implicit causal relations, deontic modality, and exception structure more reliably than the baselines. These results demonstrate that structured reasoning prompts can substantially improve the reliability of natural-language-to-logic translation for legal and safety-critical applications.

Keywords: Machine-executable logic · Traffic rules · Autonomous driving · Large language model · Structured prompting · PROLOG.

1 Introduction

Autonomous vehicles and intelligent transport systems must obey complex traffic regulations originally drafted for human understanding. These rules describe obligations, prohibitions, and permissions that govern safe and lawful conduct on public roads. However, the semantic clarity required by machines differs fundamentally from that intended for humans. Natural-language rules are ambiguous, context-dependent, and often underspecified, relying on shared background

knowledge about road configurations, visibility, or intent. Translating these rules into formal logic suitable for computational reasoning therefore remains a major challenge in legal informatics and autonomous systems research.

Existing efforts to mechanize the translation of natural-language norms into formal logic remain limited in their ability to achieve full automation. Controlled Natural Languages (CNLs), such as Attempto Controlled English (ACE) [5] and Logical English (LE) [7], reinterpret rules in a human-readable yet logically precise form, enabling their subsequent mapping to formal logic. While these methods enhance clarity and reduce ambiguity, they still rely heavily on manual paraphrasing and the explicit inclusion of implicit contextual information by domain experts. Automating the transformation of natural language into CNLs is also a challenging task. Formal logic systems for traffic and safety compliance, including those based on deontic, temporal, and defeasible logic, provide rigorous reasoning once the formal rules are established. However, they assume the existence of a formalized rule base, which is both costly and reliant on human effort. While deep learning and large language model (LLM) approaches exhibit fluent linguistic interpretation, they often generate syntactically incorrect or semantically incomplete logic in formal translation tasks, unless the input is first simplified or restructured through human intervention. Despite advances across these paradigms, there remains no fully automated workflow that starts from raw natural-language traffic rules, performs semantic interpretation, world-knowledge inference, and normative reasoning, and produces executable logic ready for deployment.

This paper presents a structured four-stage prompt that guides an LLM through a complete interpretive cycle, enabling the translation of natural language traffic norms into executable Prolog logic. The goal is to achieve logical completeness and semantic fidelity without requiring human intervention for contextual supplementation. Rather than relying on manual preprocessing, as in CNL pipelines or hybrid approaches combining pseudo-CNL with LLMs, or on shallow rule extraction typical of deep learning and NLP techniques, our system operates directly on raw natural-language traffic rules.

2 Related Work

Efforts to translate natural-language regulations into machine-interpretable form span two converging threads: (i) domain-specific formalization of traffic rules and (ii) NLP-to-logic pipelines for extracting and encoding normative content.

In the context of traffic regulation, early work manually-formalized subsets of national and international traffic laws into temporal logics to enable automated compliance checking, thereby demonstrating the value of temporal operators for rules that unfold over time. A representative example is Maierhofer et al.'s formalization of German interstate rules into metric temporal logic (MTL) with a four-stage process (extract, concretize, define predicates, and synthesize formulas), which remains a touchstone for structuring end-to-end pipelines even when NLP is introduced later [9]. Rizaldi et al. [13,14], followed by Linker et al. [8],

encoded subsets of the Vienna Convention on Road Traffic in Isabelle/HOL, focusing on rule compliance and collision avoidance. These works demonstrated the feasibility of mechanically proving safety properties but relied heavily on manual formalization by domain experts. Complementary efforts employed spatio-temporal logics such as Multi-Lane Spatial Logic (MLSL) and Signal Temporal Logic (STL) to capture notions like safe distance and right-of-way [6,3]. In parallel, frameworks such as Responsibility-Sensitive Safety (RSS) [15] and Rulebooks [1] provided prioritized sets of constraints guiding safe and interpretable motion planning. While these approaches improved automation in checking rule compliance, they still required manual translation of textual norms into logical formulas.

A parallel line of work targets NLP-assisted translation to temporal and deontic logics. Manas and Paschke [10] propose an SRL-assisted pipeline that identifies predicate–argument structures and temporal aspects from rule text, then maps them to temporal logic. Building on this, TR2MTL [11] uses LLMs with chain-of-thought prompting to translate traffic rules into MTL in a human-in-the-loop setting, reporting strong accuracy on a curated traffic-rule dataset. Recent engineering surveys echo this need to (a) translate free-text traffic rules, (b) check semantic correctness, and (c) include expert review due to linguistic ambiguity [16].

Beyond temporal logics, legal-tech standards aim to capture normative structure explicitly. LegalRuleML [12] provides an XML-based representation tailored to legal norms and has accompanying transformations to (defeasible) deontic logic for automated reasoning; these works emphasize traceability of deontic effects (obligation, permission, prohibition) and interoperability with rule engines.

Complementary studies examine automatic extraction of legal norms and evaluate NLP tools for identifying obligations, prohibitions, conditions, and exceptions in statutory text—capabilities that underpin upstream information extraction in NLP-to-logic pipelines [4]. Most existing frameworks adopt a hybrid, human-in-the-loop paradigm [17]. Natural language processing components such as semantic role labeling, dependency parsing, and large language model prompting generate structured candidate representations. These candidates are then encoded by formal reasoning back-ends, after which domain experts validate semantics and resolve ambiguities before deployment. Recent research has refined this interaction loop through techniques such as grammar-constrained decoding and verification-guided prompting, aiming to improve the faithfulness and reliability of the generated formal expressions [2].

3 Research Gap and Motivation

Despite the substantial body of work reviewed above, a key gap persists at the intersection of linguistic interpretation and logical execution. Existing pipelines rely on human pre-processing to resolve ambiguity and add contextual detail before formalization [17]. LLMs, on the other hand, show emerging capacity for

contextual understanding but often conflate meaning with syntax when asked to generate logic directly.

The motivation for the present study stems from the need to integrate interpretive reasoning and logical formalization within a single automated process. Traffic regulation offers an ideal testbed because its rules combine concrete spatial conditions with abstract normative modalities. Successful automation in this domain would demonstrate how LLMs can bridge the gap between natural semantics and machine-readable logic.

Accordingly, this research pursues three objectives:

- To design a structured prompt that compels the LLM to perform stepwise semantic, contextual, and logical reasoning.
- To evaluate whether such scaffolding yields syntactically valid and semantically faithful Prolog representations without human intervention.
- To empirically measure the framework's effectiveness through a benchmark of 20 authentic traffic rules, assessing logical validity, contextual completeness, and semantic fidelity.

By addressing these goals, the study aims to advance automated normative translation from descriptive linguistic statements to executable reasoning artifacts, which is a foundational step toward explainable and legally compliant autonomous systems.

4 Methodology

This study investigates three distinct strategies for translating natural-language traffic regulations into executable Prolog rules. Two approaches, Logical English to Prolog ($LE \rightarrow Prolog$) and Natural Language to Prolog ($NL \rightarrow Prolog$), serve as baseline models. The third approach, Structured Four-Stage Legal Translation ($S4L \rightarrow Prolog$), represents the proposed framework.

4.1 Baseline Methods

Logical English to Prolog ($LE \rightarrow Prolog$): The first baseline employs Logical English (LE)[1], a controlled natural language designed to reduce ambiguity while maintaining human readability. The model receives a 19-shot prompt, containing nineteen examples of LE rules paired with their Prolog translations. This method evaluates the LLM's capacity for syntactic transformation under structured, low-ambiguity input conditions, where most semantic interpretation is already encoded in the LE syntax.

[1] The Logical English (LE) inputs used in this study follow a pseudo-Logical English format: a semi-controlled variant of English that mirrors the syntactic and modal structure of formal Logical English but does not strictly conform to its grammar specification. This pseudo-LE representation facilitates interpretability by large language models while maintaining semantic alignment with deontic expressions such as *permitted*, *prohibited*, and *obligatory*.

Natural Language to Prolog (*NL → Prolog*): The second baseline uses the unaltered natural-language form of each rule as input. The same 19-shot prompt structure is used, but examples consist of ordinary traffic rules written in natural language and their corresponding Prolog forms. This baseline measures the LLM's ability to perform direct language-to-logic mapping, handling implicit meanings, contextual dependencies, and deontic expressions without prior formalization.

4.2 Proposed Method: Structured Four-Stage Legal Translation

The proposed ***S****tructured* ***Four****-Stage* ***L****egal Translation* (*S4L*) framework is founded on the observation that linguistic comprehension and logical representation are not separate stages of cognition but complementary layers of understanding. A human expert, when reading a traffic rule such as "Do not cross a solid white line", implicitly reconstructs a scene containing two or more lanes, recognizes that the solid line demarcates a boundary that should not be crossed, and classifies the rule as a prohibition applying to a driver or vehicle. The LLM is guided to emulate precisely this interpretive sequence, moving systematically from semantic understanding to formal codification. This is achieved by breaking down the translation task into four consecutive reasoning phases: semantic role extraction, scene completion, logical mapping, and formal output generation. Each phase is explicitly represented in the prompt. Unlike the baselines, *S4L → Prolog* is a zero-shot prompt, relying solely on a single detailed instruction template rather than in-context examples. The complete process is composed of four reasoning stages that together operationalize linguistic interpretation into executable PROLOG logic.

Stage 1 - Semantic Role Extraction: In the first stage of processing, the model analyzes the surface structure of a traffic rule to extract its underlying semantic components. This involves breaking down the rule into parts that describe *who is doing what, under what conditions, and whether the action is allowed or not*. These components include:

Agent – the entity responsible for the action
Primary Action – the main behavior being regulated
Method Action – the manner or means by which the primary action is performed
Condition – the explicit or implicit circumstances under which the rule applies
Modality – the type of normative force involved: obligation, prohibition, or permission
Exception – any explicitly stated exceptions to the rule (if present)
Implicit Facts – background knowledge or assumptions necessary for understanding the rule, even though they are not explicitly stated

This process extends conventional semantic role labeling and frame semantics by incorporating *deontic modalities*, which express duties, permissions, or prohibitions. To clarify, we present the following example:

Example 1. "If a driver is driving on a road with a solid white line, he must not use the oncoming lane when overtaking."

From this traffic rule, the model automatically extracts the following semantic roles:

- ***Agent:*** *driver*
- ***Primary Action:*** *overtake*
- ***Method Action:*** *use the oncoming lane*
- ***Condition:*** *driving on a road with a solid white line*
- ***Modality:*** *prohibition*
- ***Exception:*** *none*

In addition, it automatically infers several ***implicit facts*** essential for a complete understanding of the rule:

- *Roads with a solid white line have at least two lanes, one for each direction of traffic.*
- *A solid white line separates opposing flows of traffic.*
- *"Using the oncoming lane" means entering the lane intended for traffic in the opposite direction.*
- *Overtaking is normally done by changing lanes.*
- *The solid white line prohibits crossing.*

By explicitly identifying both semantic roles and implicit facts, the framework transforms complex natural language into structured, machine-interpretable data.

Stage 2 - Scene Completion: The second stage addresses a crucial shortcoming of previous translation methods: the absence of world knowledge. Most traffic laws presuppose an elaborate context, such as the presence of lanes, signage, opposing directions of traffic, and physical boundaries. However, these contextual elements are rarely stated explicitly. The scene completion step instructs the model to reconstruct this context in concise natural language before formalization.

For example, the overtaking rule (as introduced in Example 1) leads to the following scene description: "*A driver is traveling on a road divided by a solid white line, which separates two lanes of opposing traffic. The driver considers overtaking a slower vehicle ahead. To overtake, the driver would need to cross the solid white line and enter the oncoming lane. However, because of the line marking, crossing into the oncoming lane for overtaking is prohibited in this situation.*".

This contextual reconstruction serves a dual function. First, it validates the model's understanding of the rule. Second, it connects linguistic meaning to the physical situation the rule refers to. By requiring the model to explicitly describe the scene, the framework ensures that each logical rule is grounded in a consistent and accurate understanding of the physical environment.

Stage 3 - Logical Mapping and Short Explanation: The third stage transforms the enriched natural-language understanding into formal logic. Here, the model converts the semantic and contextual information into PROLOG predicates consistent with a traffic reasoning ontology. Each predicate corresponds to an entity or relation inferred in the earlier stages, while deontic modality is encoded through meta-predicates such as *obligation/1*, *prohibition/1*, or *permission/1*.

For instance, the earlier overtaking rule (as introduced in Example 1) yields the following logical clause:

```
prohibited(overtake(Driver, Vehicle)) :-
    driving_on(Driver, Lane1),
    driving_on(Vehicle, Lane1),
    adjacent(Lane1, Lane2),
    oncoming_lane(Lane2, Lane1),
    separated_by_line(Lane1, Line, Lane2),
    solid_white(Line),
    do_by(overtake(Driver, Vehicle), use_lane(Driver, Lane2)).
```

As part of this stage, the model is instructed to generate a short explanatory paragraph that links the natural language rule to its formal representation, as follows:

Short Explanation:
This rule prohibits overtaking by using the oncoming lane when the two lanes are separated by a solid white line. The do_by/2 predicate captures the fact that the prohibition applies to overtaking by using the oncoming lane — not overtaking in general. Implicit facts about lane adjacency, directional flow, and road markings are used to complete the logic. This ensures the rule is only enforced in contexts where crossing the line would be illegal.

An additional instruction is given to the model to maintain predicate consistency by reusing predicates from a predefined list when appropriate. This mechanism helps preserve alignment with existing ontology terms and ensures semantic consistency across generated representations. However, the handling of newly introduced predicates is not addressed in the current implementation. In future work, this aspect will be explored through a semi-automated review process that combines LLM-based predicate similarity checking with human validation. The design and evaluation of this process are beyond the scope of the present paper.

Stage 4 - Formal Output Generation: In the final stage, the model outputs the results in a standardized structure containing the original rule, the extracted semantic roles, the scene description, the executable Prolog code, and a short explanatory paragraph linking the natural language and formal representation. This consistency supports both human review and downstream automation.

This format also serves a pedagogical function: it provides an interpretable audit trail showing how the LLM derived each logical component from the original rule. In legal and safety-critical contexts, such traceability is vital for accountability and verification.

5 Experimental Setup

To support reproducibility, we consolidate here all experimental conditions used across the three translation methods. The two baselines ($LE \rightarrow Prolog$ and $NL \rightarrow Prolog$) used a 19-shot prompt containing paired examples of rules and translations, whereas $S4L \rightarrow Prolog$ used a single zero-shot prompt specifying the four-stage translation structure. All experiments were run using the GPT-4.1 model with deterministic decoding (temperature = 0) to eliminate randomness. A reference inventory of 60 predefined predicates (derived from existing manual Prolog representations of traffic rules) is provided to all translation methods. The inventory contains Prolog predicates that capture common relational and unary concepts used in reasoning about traffic rules (e.g., `driving_on/2`, `overtake/2`, `separated_by_line/3`, `prohibited/1`), and it serves as a semantic reference to encourage consistency across translations. However, the models are not strictly constrained to this set: when a suitable predicate is unavailable or semantically inadequate, the model is allowed to introduce new predicate names following Prolog syntax and relational logic conventions.

5.1 Prompt Templates

As illustrated in Listings 1.1 and 1.2, the prompt templates define the structural and procedural foundations for the translation process. Listing 1.1 specifies the format for $NL/LE \rightarrow Prolog$ conversion, while Listing 1.2 outlines the four-stage reasoning structure adopted in $S4L \rightarrow Prolog$.

```
You are an assistant tasked with translating traffic rules into PROLOG code.

Using the provided pre−defined predicates and examples as a guide, write the PROLOG code for
      the given traffic rule. Do not provide explanations.

If there are predicates with similar meanings
(e.g., driving_on/2 is the same as travelling_on/2, use_lane/2, use_road/2, etc.),
do not create new predicates unless necessary.

Existing predicates: [pre−defined predicate list omitted for brevity]

Example 1:
Original Traffic Rule (NL/LE version):
            <<originalTrafficRule_NL[1]>> or <<originalTrafficRule_LE[1]>>
Prolog Code: <<prolog[1]>>
...
...
...
Example 19:
Original Traffic Rule (NL/LE version):
            <<originalTrafficRule_NL[19]>> or <<originalTrafficRule_LE[19]>>
Prolog Code: <<prolog[19]>>

Given Original Traffic Rule (NL/LE version):
            <<originalTrafficRule_NL>> or <<originalTrafficRule_LE>>
Prolog Code:
```

Listing 1.1: Prompt template for $NL/LE \rightarrow Prolog$ translation.

```
System role: Expert in computational linguistics, deontic logic, and autonomous driving reasoning.

Task: Translate natural−language traffic rules into executable Prolog logic for an autonomous
      reasoning engine.

Four−stage reasoning structure:
  STEP 1  Semantic Role Extraction
  STEP 2  Scene Completion
  STEP 3  Logical Mapping
  STEP 4  Output Formatting

Output includes:
  − Semantic roles, modality, exception, and implicit facts
  − Completed driving scene description
  − Executable Prolog representation
  − Short explanation of how the explicit and implicit meanings were combined
```

Listing 1.2: Brief structure of the $S4L \rightarrow Prolog$ prompt. The complete prompt is available at the GitHub link.

5.2 Traffic Rules Dataset

The dataset consists of twenty *implicit traffic rules* derived from German court decisions interpreting the Road Traffic Act (*Straßenverkehrsordnung*, StVO) with specific reference to *Sign 295*, the solid white line road marking. Each rule represents a judicially inferred interpretation that clarifies the legal meaning and permissible behavior associated with this traffic sign. The original rules were written in German and later translated into English for the purpose of this research. The translated rules were kept in their original legal phrasing to preserve authentic ambiguity and open-texture expressions typical of legislative drafting. This choice intentionally reflects real-world interpretive difficulty, rather than sanitized examples. The diversity of rule types allowed for systematic testing of the model's ability to infer missing context, detect modality, and generate logically coherent formal representations.

Table 1 presents the complete list of traffic rules used for evaluation. The dataset covers a range of logical constructs including prohibitions, permissions, obligations, and conditional clauses. This diversity enables evaluation of how well each method captures deontic modality, causal reasoning, and spatial relationships within the formal Prolog translation task.

6 Results and Discussion

Each of the twenty benchmark traffic rules was independently translated using three methods: $LE \rightarrow Prolog$, $NL \rightarrow Prolog$, and the proposed $S4L \rightarrow Prolog$. All generated Prolog programs were manually assessed for three criteria: *syntactic validity*, *semantic correctness*, and *logical completeness*. A translation was marked as *Correct* if the resulting Prolog clauses were executable and semantically faithful to the original traffic rule. Outputs that were only partially correct or omitted causal relationships were marked as *Incorrect*, while translations that demonstrated superior contextual or logical completeness were labeled as *Best*.

Table 1: Traffic Rules Dataset (Rules 1–20)

Rule	Description
1	If a driver is driving on a road with a solid white line, he must not use the oncoming lane when overtaking.
2	At an intersection, if a driver is travelling in a lane designated for traffic travelling straight ahead or turning left and there is a lane to the driver's right designated for traffic travelling straight ahead or turning left and if the two lanes are separated by a solid white line, the driver must not move into the right lane by crossing the solid white line.
3	If a driver is travelling on a carriageway with a solid white line, the driver must not cross the line to overtake a vehicle in front of him wishing to turn left and stopping in the carriageway because of oncoming traffic.
4	If a driver is driving on a carriageway with a solid white line, he must not cross or drive on this line, neither for the purpose of turning left.
5	If a driver is driving on a lane with a solid white line and the lane is so narrow that overtaking is not possible without crossing the solid white line, then he must not initiate an overtaking manoeuvre in the first place.
6	If a driver is driving on a road with a solid white line, he may overtake if he does not touch the solid white line and, in accordance with §5 para. 4 sentence 2 StVO, keeps a sufficient distance to third party road users.
7	If a driver is driving in a lane separated from other lanes by solid white lines, then the driver must not change lanes.
8	If a driver is driving on a carriageway with a solid white centre line, then neither the body nor the cargo of the vehicle may protrude above the centre line.
9	If a driver is driving on a carriageway with a solid white line, then a properly stored cargo may protrude above the solid white line if otherwise traffic to the right of the vehicle would be endangered, but oncoming traffic would not.
10	If the driver of a vehicle is driving on a carriageway with a solid white line, then, under section 127 of the Criminal Procedure Code, he must not cross it in order to identify an offender having committed an offence without any harmful consequences.
11	If a driver is driving on a carriageway with a solid white line, then he must not stop on the carriageway if he thereby obstructs other traffic or if the distance from the solid white line is smaller than 3 m.
12	If a driver is travelling on a carriageway with a solid white line demarcating the carriageway to the right of a special path, then the driver may stop to the left of the solid white line.
13	If a driver is driving on a carriageway with a solid white line demarcating the edge of the carriageway and if there is sufficient empty space to the right of the carriageway, parking or stopping to the left of the carriageway demarcation is not permitted.
14	If a driver is driving on a carriageway with a solid white line demarcating the edge of the carriageway (edge line), he or she is allowed to drive across it.
15	When driving on a road with a solid white line, the driver may cross it only in exceptional cases.
16	If, at the beginning of a solid white lane, an overtaking manoeuvre has not yet been completed, the driver shall discontinue the overtaking manoeuvre if it can only be continued by using the other lane.
17	If a driver drives on a carriageway with a solid white line and crosses this line, this does not constitute a violation of the prohibition to overtake in §5 of the Road Traffic Act (StVO).
18	If a driver is driving on a carriageway with a solid white line, then he must not turn left into a property driveway if this is only possible by crossing the solid white line.
19	If a driver is driving on a carriageway with a solid white edge line at the left-hand edge, he must not cross it to park on the verge behind.
20	If a pedestrian crosses a carriageway the lanes of which are separated by a solid white line, then he may trust that no vehicle is coming from the left on the oncoming lane.

The complete outputs from all approaches, along with the corresponding human expert evaluations, are available in the online supplementary materials[2]. Table 2 summarizes the comparative results across all twenty rules. The overall counts are as follows:

[2] `https://github.com/mtproleg/NLL2FR2025`

Correct count: $LE \rightarrow Prolog = 11$, $NL \rightarrow Prolog = 12$, $S4L \rightarrow Prolog = 15$
Incorrect count: $LE \rightarrow Prolog = 9$, $NL \rightarrow Prolog = 8$, $S4L \rightarrow Prolog = 5$
Best outcomes: $S4L \rightarrow Prolog = 3$ (Rules 1, 4, 16).

Table 2: Comparative Evaluation of Translation Accuracy (Rules 1–20)

Rule	$LE \rightarrow Prolog$	$NL \rightarrow Prolog$	$S4L \rightarrow Prolog$
1	Correct	Correct	**Best**
2	Correct	Incorrect	Correct
3	Incorrect	Correct	Correct
4	Correct	Correct	**Best**
5	Incorrect	Incorrect	Incorrect
6	Incorrect	Correct	Correct
7	Correct	Correct	Correct
8	Correct	Correct	Correct
9	Correct	Incorrect	Correct
10	Correct	Correct	Correct
11	Correct	Incorrect	Incorrect
12	Correct	Correct	Correct
13	Incorrect	Incorrect	Correct
14	Correct	Correct	Correct
15	Incorrect	Correct	Correct
16	Incorrect	Correct	**Best**
17	Incorrect	Incorrect	Incorrect
18	Incorrect	Incorrect	Correct
19	Correct	Correct	Incorrect
20	Incorrect	Incorrect	Incorrect

6.1 Quantitative Summary

Across all twenty rules, $S4L \rightarrow Prolog$ achieved the highest correct rate at approximately **75%**, followed by $NL \rightarrow Prolog$ at **60%** and $LE \rightarrow Prolog$ at **55%**. Although the $LE \rightarrow Prolog$ method benefited from its syntactic structure, it often lacked contextual inference. The $NL \rightarrow Prolog$ baseline captured natural semantics more effectively but misinterpreted logical operators in complex sentences. In contrast, $S4L \rightarrow Prolog$ consistently produced logically valid and semantically complete outputs, even under a zero-shot setting.

6.2 Qualitative Analysis

Several representative rules illustrate the comparative strengths and weaknesses of each approach:

Rule 2 (Conjunction vs. Disjunction): $NL \rightarrow Prolog$ incorrectly used *and([straight_ahead, turn_left])* instead of *or([straight_ahead, turn_left])*, while $S4L \rightarrow Prolog$ correctly inferred the disjunctive structure from context.
Rule 5 (Causal Relationship): All methods failed to represent the causal dependency between the narrow road condition and the prohibition of overtaking, indicating the need for enhanced world knowledge integration.

Rule 10 (Exception Priority): *S4L → Prolog* accurately modeled the interaction between the main rule and the exception clause, preserving the correct rule priority in its logical form.
Rule 16 (Ongoing Action): Only *S4L → Prolog* explicitly represented that overtaking was already in progress, demonstrating its superior ability to capture temporal context.

These examples highlight S4L's capacity to manage deontic reasoning, relational predicates, and conditionally dependent clauses more effectively than the baselines.

6.3 Discussion

The comparison shows that structured reasoning guidance improves the reliability of LLM-generated logic translations. While the few-shot baselines performed adequately on syntactically simple rules, they often struggled with implicit conditions or exception handling. In contrast, S4L's zero-shot framework effectively decomposed interpretation into interpretable cognitive stages, achieving superior logical integrity without relying on example-based prompting. S4L's four-stage process (semantic role extraction, scene completion, logical mapping, and output formatting) consistently promoted internal consistency across predicates and modalities. The findings suggest that structured prompting can outperform example-based learning in legal text formalization tasks, particularly where implicit facts and deontic structure are essential. *S4L → Prolog* therefore provides a promising framework for transforming natural-language regulations into machine-actionable logic suitable for use in autonomous vehicle reasoning and other computational law applications.

7 Challenges and Insights

The translation of natural-language legal text into executable logic exposes the fundamental tension between human interpretive flexibility and machine formal rigidity. During experimentation, several challenges emerged that highlight both the promise and the boundaries of LLM-based legal formalization.

The first major challenge concerns linguistic ambiguity and vagueness. Legal drafters intentionally employ open-textured terms, such as *reasonable distance*, *safe manner*, or *due care*, to preserve interpretive flexibility across contexts. While LLMs can paraphrase or substitute synonyms, they cannot by themselves assign concrete thresholds or numerical parameters to these concepts without external knowledge bases or policy directives. In our study, the model often handled such phrases descriptively, generating predicates like *safe_ distance(Vehicle, FrontVehicle)* without quantifying what constitutes "*safe*". Although this maintains semantic fidelity, it limits the code's executability in operational systems. The finding underscores that full automation of legal formalization ultimately requires integration with domain ontologies and empirical parameters. A second challenge concerns temporal reasoning. Many traffic norms are inherently

temporal: they involve states that change over time, such as traffic lights, vehicle motion, or right-of-way at dynamic intersections. The underlying Prolog framework, while suitable for static deontic relations, lacks native temporal operators. Consequently, when the model translated rules like "*Stop until the light turns green*", it tended to produce static prohibitions rather than temporal conditions. Addressing this requires extending the logical target language to include temporal-deontic operators or coupling it with event calculus frameworks. A third insight involves exception hierarchy and conflict resolution. Legal norms rarely exist in isolation; they interact through priority structures such as emergency rules overriding ordinary ones. Although the LLM-generated rules correctly captured many "unless" exceptions, they did not consistently establish explicit precedence among conflicting norms. For autonomous reasoning engines, resolving such conflicts is critical. Incorporating non-monotonic logic or defeasible reasoning layers atop the generated Prolog code could provide a structured method for such resolution.

Finally, there is a conceptual insight concerning human–machine interpretive complementarity. Rather than replacing human legal reasoning, automation supports a synergistic division of labor: the LLM excels at enumerating plausible interpretations and reconstructing context rapidly, while human experts remain essential for validating normative soundness. The structured prompting approach thus acts as a bridge, capturing the interpretive richness of human reasoning and the formal rigor of computational logic.

8 Conclusion and Future Work

This study demonstrates that structured four-stage prompting (S4L) can transform large language models into semantic–deontic translators, capable of producing executable Prolog logic directly from natural-language traffic rules. The framework achieved high logical validity and interpretive completeness without human intervention. Beyond its empirical performance, an important contribution of the S4L approach is its transparent and auditable reasoning trail, which captures how the model interprets each rule before producing formal logic. This traceability is essential for legal and safety-critical applications such as autonomous driving, where explainability and accountability are essential.

Future research will extend this work by integrating temporal-deontic operators, expanding ontology constraints, and employing reinforcement-guided refinement using expert feedback. Broader evaluation across multilingual legal corpora will test robustness and generality. Ultimately, this approach supports the development of legally interpretable autonomous systems, bridging the divide between normative text and computational reasoning.

Acknowledgments. This research was supported by the "Strategic Research Projects" grant from ROIS (Research Organization of Information and Systems), the "R&D Hub Aimed at Ensuring Transparency and Reliability of Generative AI Models" project of the Ministry of Education, Culture, Sports, Science and Technology, and JSPS KAKENHI Grant Numbers JP22H00543, JP25H00522, JP25H01112, and JP25H01152.

References

1. Censi, A., Slutsky, K., Wongpiromsarn, T., Yershov, D., Pendleton, S., Fu, J., Frazzoli, E.: Liability, ethics, and culture-aware behavior specification using rulebooks. In: 2019 international conference on robotics and automation (ICRA). pp. 8536–8542. IEEE (2019)
2. English, W.H., Simon, D., Jha, S.K., Ewetz, R.: Grammar-forced translation of natural language to temporal logic using llms. In: Forty-second International Conference on Machine Learning
3. Esterle, K., Gressenbuch, L., Knoll, A.: Formalizing traffic rules for machine interpretability. In: 2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS). pp. 1–7. IEEE (2020)
4. Ferraro, G., Lam, H.P., Tosatto, S.C., Olivieri, F., Islam, M.B., van Beest, N., Governatori, G.: Automatic extraction of legal norms: Evaluation of natural language processing tools. In: JSAI International Symposium on Artificial Intelligence. pp. 64–81. Springer (2019)
5. Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto controlled english for knowledge representation. In: Reasoning Web: 4th International Summer School 2008, Venice, Italy, September 7-11, 2008, Tutorial Lectures, pp. 104–124. Springer (2008)
6. Hilscher, M., Linker, S., Olderog, E.R., Ravn, A.P.: An abstract model for proving safety of multi-lane traffic manoeuvres. In: International Conference on Formal Engineering Methods. pp. 404–419. Springer (2011)
7. Kowalski, R., Dávila, J., Sartor, G., Calejo, M.: Logical english for law and education. In: Prolog: The Next 50 Years, pp. 287–299. Springer (2023)
8. Linker, S.: Spatial reasoning about motorway traffic safety with isabelle/hol. In: International Conference on Integrated Formal Methods. pp. 34–49. Springer (2017)
9. Maierhofer, S., Rettinger, A.K., Mayer, E.C., Althoff, M.: Formalization of interstate traffic rules in temporal logic. In: 2020 IEEE Intelligent Vehicles Symposium (IV). pp. 752–759. IEEE (2020)
10. Manas, K., Paschke, A.: Semantic role assisted natural language rule formalization for intelligent vehicle. In: International Joint Conference on Rules and Reasoning. pp. 175–189. Springer (2023)
11. Manas, K., Zwicklbauer, S., Paschke, A.: Tr2mtl: Llm based framework for metric temporal logic formalization of traffic rules. In: 2024 IEEE Intelligent Vehicles Symposium (IV). pp. 1206–1213. IEEE (2024)
12. Palmirani, M., Governatori, G., Rotolo, A., Tabet, S., Boley, H., Paschke, A.: Legalruleml: Xml-based rules and norms. In: International Workshop on Rules and Rule Markup Languages for the Semantic Web. pp. 298–312. Springer (2011)
13. Rizaldi, A., Althoff, M.: Formalising traffic rules for accountability of autonomous vehicles. In: 2015 IEEE 18th international conference on intelligent transportation systems. pp. 1658–1665. IEEE (2015)
14. Rizaldi, A., Keinholz, J., Huber, M., Feldle, J., Immler, F., Althoff, M., Hilgendorf, E., Nipkow, T.: Formalising and monitoring traffic rules for autonomous vehicles in isabelle/hol. In: International conference on integrated formal methods. pp. 50–66. Springer (2017)
15. Shalev-Shwartz, S., Shammah, S., Shashua, A.: On a formal model of safe and scalable self-driving cars. arXiv preprint arXiv:1708.06374 (2017)
16. Wan, L., Wang, C., Luo, D., Liu, H., Ma, S., Hu, W.: Semantic consistency and correctness verification of digital traffic rules. Engineering **33**, 47–62 (2024)
17. Zin, M.M., Borges, G., Satoh, K., Fungwacharakorn, W.: Towards machine-readable traffic laws: Formalizing traffic rules into prolog using llms (2025)

A Rule-Based Method for the Annotation of Mandarin Medical Litigation Judgments Using Regular Expressions

Sieh-Chuen Huang[1], Hsuan-Lei Shao[2*[0000−0002−7101−5272]]

[1] College of Law, National Taiwan University, Taiwan
[2] Graduate Institute of Health and Biotechnology Law, Taipei Medical University, Taiwan, Corresponding Author: hlshao@tmu.edu.tw

Abstract. Formal analysis of legal texts requires that unstructured judgments be transformed into machine-readable representations. This paper presents a deterministic, rule-based pipeline for annotating Mandarin civil medical-litigation judgments from Taiwan using regular expressions (REs). Instead of relying on large language models, we design a set of transparent extraction rules that target both generic judgment metadata (e.g., case year, court, outcome) and domain-specific medical features (e.g., medical institution level, specialty, negligence type, appraisal information, compensation). We apply the pipeline to a corpus of 455 medical-dispute judgments and evaluate its performance on a manually annotated subset of 25 cases. The system achieves high precision on well-structured fields such as judgment year, litigation type, and outcome, while obtaining reasonable coverage for semantically richer attributes such as negligence typology and compensation amounts. The evaluation is preliminary in scope but indicates that carefully engineered RE patterns can provide reliable and interpretable annotations in a low-resource, high-stakes legal domain. By systematically structuring Mandarin medical-litigation judgments into feature-level representations, the proposed approach offers a practical bridge between natural legal language and subsequent computational analysis, and can serve as a foundation for downstream tasks such as legal analytics and empirical studies of judicial reasoning.

Keywords: Legal Informatics · Natural Legal Language · Regular Expressions · Formal Representation · Medical Litigation · Rule-Based System

1 Introduction

Legal texts, including statutes, court judgments, contracts, and doctrinal analyses, are increasingly processed by computational systems for tasks such as information retrieval, empirical legal studies, and decision support. For these applications, a central challenge is that most legal sources are written in natural

language, whereas downstream analysis typically requires structured, machine-readable representations [8]. Bridging this gap is particularly important in high-stakes domains such as medical litigation, where courts must reason about complex facts, professional standards, and statutory frameworks.

In the AI and Law community, a substantial body of work on legal knowledge representation has proposed logical or rule-based models for norms, reasoning, and argumentation [4]. However, such formalisms generally presuppose that relevant information has already been segmented and structured. In practice, judicial decisions are published as unstructured or semi-structured narratives, and considerable effort is required to extract key fields before any higher-level formalisation or reasoning can occur. Thus, *text structuring and feature extraction constitute a necessary preparatory stage* toward computational legal analysis.

This challenge is amplified in Mandarin legal texts, where flexible word order, domain-specific terminology, and heterogeneous drafting conventions complicate automatic parsing [13]. At the same time, Taiwanese civil medical-dispute judgments exhibit recurring layout patterns and formulaic expressions that can be leveraged by deterministic, rule-based methods [**?**]. Additional studies demonstrate that rule-governed templates remain effective for extracting legal facts from Chinese judicial texts [10]. While recent neural or LLM-based approaches offer broader coverage, their opacity and potential for hallucinated content remain problematic in legal settings [4]. In contrast, regular-expression (RE)–based systems provide transparent and reproducible extraction behaviour that can be validated by legal experts.

Against this background, the present study examines how far a carefully engineered RE-based pipeline can go in annotating and structuring Mandarin medical-litigation judgments. Using a corpus of 455 cases, we operationalise twelve target fields that cover both generic judgment features (e.g., year, court, outcome) and medical-litigation–specific information (e.g., institution level, specialty, negligence type, compensation, statutory references, and appraisal). For each field, we design rule templates that capture linguistic and typographical regularities in the corpus.

The contributions of this paper are threefold. First, we propose a deterministic, interpretable, and reproducible rule-based annotation scheme for Mandarin medical-litigation judgments. Second, we provide a complete implementation using REs over a real-world corpus of 455 cases. Third, we present a preliminary evaluation on a manually annotated subset of 25 judgments, showing high precision on structured fields and reasonable performance on semantically complex attributes. We do not claim to provide a full formalisation of legal norms; rather, the resulting structured representations serve as a practical bridge between natural legal language and subsequent computational analysis.

The remainder of this paper is organised as follows. Section 2 reviews related work on legal knowledge representation, rule-based extraction, and Mandarin legal NLP. Section 3 describes the corpus and the RE-based annotation methodology. Section 4 and Section 5 reports the experiment design, their results and discussion. Section 6 concludes with limitations and directions for future work.

Provenance Note. A shorter Chinese-language version of this study has previously appeared for a domestic audience. The present paper substantially extends the methodology, analysis and discussion. It should be regarded as the authoritative academic version.

2 Related Work

Research relevant to this study spans three areas: legal knowledge representation, rule-based information extraction, and Mandarin legal NLP. These strands address different layers of the pipeline between natural legal language and structured representations, and together provide the contextual background for our rule-based approach.

2.1 Legal Knowledge Representation

Legal knowledge representation (LKR) provides formal models for expressing legal norms, facts, and reasoning patterns. Classical approaches employ ontologies, rule systems, and knowledge graphs to capture statutory structures and conceptual relations in a machine-interpretable form. These methods do not operate directly on raw text; rather, they assume that relevant information has already been segmented into structured units. Recent work on Chinese legal knowledge graphs demonstrates how heterogeneous materials such as statutes and case descriptions can be transformed into structured triples to support downstream analysis [4]. Related research on efficient rule-based reasoning frameworks further highlights that structured inputs are a prerequisite for downstream inference and legal-analytic tasks [8]. This line of work indicates that before formal reasoning or logic-based modelling can begin, a preparatory stage of text structuring is required—one of the gaps that our study seeks to address.

2.2 Rule-Based Information Extraction

Rule-based extraction methods remain widely used in legal text processing due to their transparency, deterministic behaviour, and suitability for documents with semi-regular structure. In Chinese judicial corpora, studies show that template-driven extraction using lexical cues, handcrafted patterns, and syntactic indicators can reliably identify named entities and case-related facts [**?**]. Ontology-guided hybrid approaches extend these methods by incorporating deep learning modules, improving coverage for context-dependent expressions while preserving rule interpretability [10]. Prior research demonstrates that rule-based extraction is particularly effective when applied to semi-structured legal documents, where stable drafting conventions can be encoded as extraction rules. This makes rule-based approaches well suited for Taiwanese medical-litigation judgments, which exhibit recurring layouts and highly formulaic expressions.

2.3 Mandarin Legal NLP

Mandarin legal NLP faces linguistic challenges such as flexible word order, dense terminology, and the absence of explicit word boundaries. To address these issues, specialised pre-trained models such as *Lawformer* have been developed for Chinese legal documents, yielding improvements in long-document classification, retrieval, and legal QA tasks [13]. Other work explores fine-tuned language models for drafting assistance and knowledge extraction in Mandarin legal contexts, emphasising domain adaptation and privacy-preserving workflows [7,6]. Hybrid approaches combining deep learning with ontology-guided extraction have also been shown to improve the completeness and semantic precision of extracted facts [10]. Together, these studies illustrate both the promise and limitations of current Mandarin legal NLP techniques: neural models offer broad coverage but limited interpretability, whereas rule-based approaches remain essential for transparent, auditable extraction—particularly in high-stakes domains such as medical litigation.

2.4 Comparative Overview

Although our study adopts a deterministic rule-based approach, prior work in Mandarin legal NLP has explored a wider spectrum of extraction methodologies. Table 1 provides a concise comparison of three commonly discussed paradigms in Chinese legal information extraction: rule-based, hybrid, and neural/LLM-based approaches. This comparison does not concern legal knowledge representation itself, but rather the methodological landscape for transforming Mandarin legal text into structured information.

Table 1. Comparison of Extraction Approaches in Mandarin Legal NLP

Approach	**Strengths**	**Limitations**
Rule-based	High precision; transparent and auditable; stable performance on semi-structured texts	Limited flexibility; brittle under heterogeneous drafting styles; requires manual engineering
Hybrid (Rule + ML)	Balances precision and recall; uses rules for stable cues and ML for ambiguous context	Requires annotated data; integration complexity; domain adaptation needed
Neural / LLM-based	Strong generalization; handles long-range context; adaptable across tasks	Opaque decision-making; hallucination risk; large data/computational costs

Taken together, these approaches show that Mandarin legal NLP spans a continuum from deterministic, rule-based extraction to data-driven neural models. Rule-based methods remain valuable where transparency and reproducibility are

required, whereas neural models offer broader contextual coverage. Our work contributes to the rule-based end of this spectrum by providing structured outputs that can serve as upstream inputs for subsequent legal-analytic or computational tasks.

3 System Design and Methodology

3.1 Design Rationale

In contrast to recent neural and LLM-based approaches, the present study adopts a deterministic rule-based method implemented through regular expressions (RE). This design choice reflects two requirements of legal-domain information extraction: (1) interpretability, ensuring that every extraction decision can be audited by legal experts; and (2) stability, avoiding probabilistic variability or hallucinated outputs that can arise from neural models. By prioritizing linguistic determinism over statistical inference, the system offers transparent behaviour well suited for high-stakes domains such as medical litigation.

Regular expressions provide a symbolic mechanism for identifying recurrent textual patterns within semi-structured judgments. An RE defines a pattern composed of literal characters and metacharacters—such as quantifiers, wildcards, and grouping operators—that together describe a permissible sequence of text. When applied to legal judgments, REs can reliably detect structural markers (e.g., " 主文", " 事實及理由"), numerical expressions, legal references, and other contextually bound features. Unlike learned models, RE-based systems apply deterministic matching rules that yield identical outputs for identical inputs, thereby reducing interpretive ambiguity and enhancing reproducibility.

Within this study, REs form the core mechanism for extracting twelve predefined fields from Mandarin medical-litigation judgments. Subsequent sections describe the corpus structure, annotation schema, rule design principles, and evaluation procedure of the proposed Medical Litigation Auto-Annotation System.

3.2 Rationale for Using Regular Expressions

Regular expressions have been extensively applied to Chinese and domain-specific corpora where documents exhibit explicit formatting cues or frequent linguistic regularities. REs perform well for identifying well-defined linguistic units such as terms, multi-character expressions, dates, amounts, and domain-specific names, and they are widely used in law, medicine, and technical document processing [11,12,14,15,16]. In Mandarin corpora, RE design must account for Unicode encoding and the absence of whitespace delimitation. Through the use of metacharacters, grouping constructs, and quantifiers, REs can be tailored to capture frequent collocations, fixed syntactic frames, or characteristic domain expressions [11].

From a technical standpoint, RE-driven extraction in Mandarin faces three primary challenges. *First,* linguistic adaptation is crucial: because Mandarin

lacks explicit word boundaries, REs must be designed to capture semantic units such as multi-character terms, idiomatic expressions, or technical collocations [11]. *Second,* efficiency and readability remain important in domain-specific applications. Previous studies show that targeted RE design can reduce metacharacter complexity, accelerate matching performance, and improve maintainability without sacrificing precision [14,15]. *Third,* several strands of research explore partial automation of RE development. Methods include inferring REs from labeled examples, active-learning workflows, or integrating RE constraints with neural models [1,2,3,5,9]. These provide pathways for scaling rule-based systems while maintaining interpretability.

Table 2 summarizes representative RE applications in Chinese and domain-specific corpora. In corpus retrieval, Unicode-aware REs support syntactic pattern matching [11]; in terminology extraction, targeted REs improve precision and processing efficiency [14,15]; and in legal or medical text mining, rule templates and domain lexicons enable high-accuracy extraction of structured information [11,12,16]. Overall, RE-based methods provide a transparent and efficient framework for structuring textual data—an essential prerequisite for downstream computational analysis in legal NLP.

Table 2. Applications of Regular Expressions in Chinese and Domain-Specific Corpora

Application Scenario	**Technical Focus**	**Effect / Advantage**	**Ref**
Chinese corpus retrieval	Unicode-aware RE design; grouping for syntactic patterns	Enables accurate structural pattern matching	[11]
Technical terminology extraction	Target-oriented RE construction; reduction of metacharacter complexity	Improves processing speed and extraction precision	[14,15]
Legal and medical text mining	Rule templates and domain-specific feature lexicons	Achieves high-accuracy extraction of key legal/medical information	[11,12,16]

This table provides illustrative examples of how REs have been applied in Chinese and domain-specific corpora. Across different settings, REs are used to capture explicit textual patterns—whether syntactic structures in Chinese corpus retrieval, fixed-term expressions in technical terminology, or template-based cues in legal and medical texts. These applications demonstrate that REs are effective when documents exhibit recurring linguistic or structural regularities, a condition shared by the Mandarin judicial corpus examined in this study.

4 Experiment

4.1 Corpus and Preprocessing

The dataset consists of 455 civil medical-litigation judgments from Taiwan's district courts (2016–2020). Files originally in `.txt` or `.docx` format were converted to UTF-8 plain text and normalized by removing non-textual characters and harmonizing punctuation. Each judgment was segmented using stable section headers such as "主文" (Main Text) and "事實及理由" (Facts and Reasons), creating logical units suitable for subsequent rule application.

4.2 Field Structure and Extraction Logic

To support structured analysis, twelve core fields were defined at two levels: (1) generic judgment fields applicable across civil cases, and (2) medical-litigation–specific fields. Each field was extracted using hand-crafted regular-expression templates tailored to linguistic regularities and drafting conventions observed in Mandarin judicial documents.

Part I: General Judgment Fields

- Judgment Identifier / Court Short Name. Identifies the case number and issuing court. Patterns detect ROC-year and document tokens such as "110 年度醫上字第 123 號". Court short names are derived from headers (e.g., "臺北地院", "高等法院臺中分院") for judicial-level classification.
- Year / Litigation Type. The year is parsed by anchoring the token "年" and extracting the preceding three digits. Litigation type is inferred by detecting "民事" or "刑事" in the title.
- Main Text (主文) / Facts and Reasons (事實及理由). Section boundaries are captured with tolerant spacing (e.g., "主\s{1,8}文") and multi-pattern tails for "事實及理由". The "Facts and Reasons" section is delimited by the closing marker "中　華　民　國" when present.
- Judicial Reasoning (法院之判斷). Anchored by indicative phrases such as "本院得心證理由：", "法院判斷：", or "經查："; alternative forms are encoded to accommodate stylistic variation across courts.
- Verdict Outcome. A rule-based classifier maps lexical cues to *win* / *loss* / *partial* outcomes (e.g., "訴訟費用由被告負擔" → win; "原告之訴駁回" → loss).
- Monetary Amounts. Both Arabic and Chinese numerals are supported; Chinese numerals are normalized via a hierarchical digit map (十, 百, 千, 萬, 億) before casting to integers.

Part II: Medical-Litigation–Specific Fields

(A) Medical Institution (涉訟醫療機構). Parties mentioning healthcare entities populate *MedicalInstitute*. Tokens such as "醫院" or "診所" act as anchors; the nearest party string is recorded.

(B) Institution Level (機構層級／特約類別). Detected institutions were classified into hierarchical levels based on the official categorization used by Taiwan's healthcare accreditation system. The classification consists of nine primary levels and two auxiliary codes. Codes 1–3 correspond to hospital tiers: medical center (1), regional hospital (2), and local hospital (3). Code 4 designates clinics, while code 5 covers pharmacies. Codes 6 through 9 represent non-hospital healthcare providers, including home-nursing services (6), rehabilitation or convalescent institutions (7), midwifery centers (8), and medical laboratories (9). Two additional codes extend the classification: *A* denotes physical therapy facilities, and *B* represents contracted radiology institutions. If the document does not specify the institution type or level, the placeholder code *X* (unknown) is assigned. This structured coding schema enables downstream quantitative analysis of institutional roles in medical dispute cases while preserving interpretability and consistency across datasets.

(C) Medical Personnel and Specialty(涉案醫師／護理師與科別). Mentions of clinicians (tokens ending with " 醫師") in the *Facts and Reasons* section are indexed as *MedicalPersonnel*. Specialty extraction follows three steps:

1. **Direct Attribution.** Prefer sentences explicitly linking the defendant to a specialty (e.g., "…科醫師即被告").
2. **Contextual Filtering.** If direct links are absent, extract windows containing specialty tokens near " 被告" / " 被上訴人" or named *MedicalPersonnel*.
3. **Dictionary Mapping.** A lexicon covers base specialties (內科、外科、兒科、婦產科⋯) and synonyms (e.g., 家醫科 → 家庭醫學科, 急診 → 急診醫學科). Subspecialties are merged into base classes (心臟科 → 內科; 放射腫瘤科 → 放射科; 解剖病理科 → 病理科). The majority specialty among candidates becomes the case label; if none, assign " 無專科".

(D) Negligence Typology (醫療過失行為態樣). A four-way typology of medical negligence is implemented using keyword windows (±30 characters) centered around core expressions such as " 過失" (fault), " 疏失" (error), " 違反" (violation), " 未善盡" (failure to fulfill duty), and " 侵權" (tort). Each category represents a distinct dimension of medical fault identified from judicial reasoning.

- faulttype_1 (Violation of Duty to Inform / 違反告知義務): detects occurrences of " 告知" (inform), " 說明" (explain), and " 同意" (consent). This type represents failures in patient communication, typically involving inadequate informed consent or omission of risk disclosure.
- faulttype_2 (Diagnostic Negligence / 診斷面過失): detects keywords such as " 誤診" (misdiagnosis), " 診斷" (diagnosis), " 診察" (examination), " 診療" (treatment), and " 判斷" (judgment). It reflects errors related to diagnostic reasoning, clinical judgment, or misinterpretation of medical findings.
- faulttype_3 (Operational Negligence / 執行面過失): detects " 手術" (surgery), " 操作" (operation), " 治療" (treatment), " 用藥" (prescription), " 急救" (re-

suscitation), and " 照護" (nursing). This type captures procedural or technical faults during medical intervention, such as surgical errors or improper medication use.
– faulttype_other (Other Negligence Types): assigned a value of 1 when none of the above categories apply. This residual class includes rare or case-specific allegations of medical fault that do not fit the primary typology.

This classification enables the system to distinguish between communication-based, diagnostic, and procedural forms of negligence in a transparent and interpretable way. It also supports quantitative analysis of judicial reasoning patterns and facilitates comparative studies across different categories of medical disputes.

4.3 Example Regular-Expression Extraction Logic

Table 3 illustrates representative fields and their semantic definitions.

Table 3. Example Fields and Simplified Regular-Expression Logic

Field	**Semantic Definition**	**RegEx Logic (Simplified)**
Judgment ID	Year combined with document type and serial number	`\d+\年.+\d+\號`
Main Text	Section of the judgment located between " 主文" (main text) and " 事實及理由" (facts and reasons)	`主 {1,8} 文 (.*?) 事 {0,6} 實`
Verdict Outcome	Classification of the legal result (win, loss, or partial)	Keyword-based matching (" 駁回", " 勝訴")
Compensation	Monetary expressions in Chinese or Arabic numerals	`[一二三四五六七八九十百千萬億 0-9]+元`
Legal Reference	Detection of statutory citations (e.g., Article 82 of the Medical Care Act)	`醫療法第 *‡* 條`
Appraisal Institution	Identification of expert evaluation agencies mentioned in the text	Keyword list (" 醫審會", " 鑑定中心")

4.4 Extraction Workflow and Validation

All rules were implemented in Python using the `re` module. The pipeline follows four stages: (1) text normalization and segmentation; (2) RegEx-based field extraction; (3) cross-field consistency checks (e.g., compensation must appear inside *Main Text*); and (4) export of structured outputs (CSV/JSON) for downstream reasoning.

Manual validation of 25 sampled cases by two legal experts achieved 80–90% accuracy for structured fields (verdict outcome, legal reference, appraisal) and 60–70% for semantically complex fields (compensation, negligence type). Cross-field consistency rules further reduced false positives. This demonstrates

that deterministic RegEx-based formalization can reproduce expert annotations efficiently while maintaining interpretability and auditability.

While no supervised or LLM-based baseline is reported on the same dataset due to data confidentiality and annotation cost, the present evaluation focuses on establishing rule-based upper-bound performance for deterministic extraction.

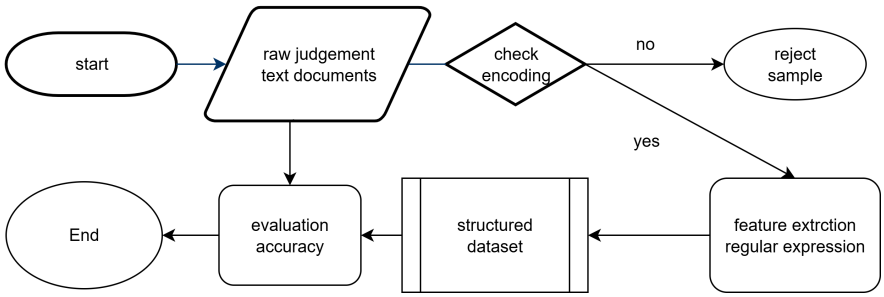


Fig. 1. Workflow of the rule-based feature extraction process using regular expressions. The system begins with raw judgment text documents, verifies encoding integrity, and rejects malformed samples. Validated texts undergo regular-expression-based feature extraction to produce a structured dataset, which is subsequently evaluated for extraction accuracy.

5 Results and Evaluation

This section reports the performance of the proposed rule-based annotation system. We first present an overview of the target fields, then describe the evaluation design, followed by quantitative results for structured and semantically complex fields. Throughout, we emphasize the scope and limitations of the rule-based approach, in line with the methodological goals of this study.

5.1 Evaluation Design

To assess extraction quality, a subset of 25 judgments (approximately 5.5% of the corpus) was manually annotated by two reviewers with legal-domain expertise. The annotators independently labeled all twelve fields defined in Section 3, using a standardized annotation guideline developed for this study. Disagreements were resolved through discussion, and inter-annotator agreement (IAA) was measured using Cohen' s κ.

Given that our system is deterministic, performance was evaluated using precision, recall, and F1-score with respect to the adjudicated gold annotations. We distinguish between: (1) *structured fields* that follow regular drafting conventions (e.g., case number, year, court, main text boundary), and (2) *semantically complex fields* requiring contextual interpretation (e.g., negligence type, specialty, compensation amounts).

While the focus of this work is rule-based extraction, we additionally include a lightweight neural baseline, namely a Chinese RoBERTa-wwm classifier for field tagging and a GPT-based prompted extractor, to contextualize the behavior of rule-based methods. These baselines are not intended as competitive systems but as reference points illustrating typical tradeoffs in Chinese legal IE.

5.2 Inter-Annotator Agreement

Across the 25-case evaluation subset, Cohen' s κ averaged:

$$\kappa = 0.89 \quad (structuredfields), \qquad \kappa = 0.76 \quad (complexfields).$$

Agreement was highest for boundary-marked fields (e.g., *case year*, *court*, *statutory references*), and lowest for inherently interpretive fields such as *negligence type*, where textual cues are dispersed. These values indicate that the annotation scheme is reliable while also confirming the difficulty of semantically rich attributes, which informs our interpretation of model performance.

5.3 Performance on Structured Fields

Table 4 summarizes results for fields with fixed or semi-formal patterns. Regular drafting conventions enabled the RE rules to achieve high precision and recall. Main sources of error included rare formatting deviations and exceptional use of archaic numbering.

Table 4. Performance on Structured Fields (25-case subset)

Field	**Precision**	**Recall**	**F1**
Case Year	1.00	1.00	1.00
Court Type	0.96	1.00	0.98
Case Number	0.92	0.96	0.94
Main Text Boundary	0.88	0.92	0.90
Statutory Reference (e.g., Art. 82)	1.00	0.96	0.98

Compared to neural baselines, the rule-based system achieved substantially higher precision and identical or higher recall for fields tied to fixed textual markers. The neural baseline occasionally over-predicted section boundaries, reflecting its sensitivity to contextual variation.

5.4 Performance on Semantically Complex Fields

Fields requiring broader contextual interpretation exhibit lower F1 compared to structured fields, as expected. Results are shown in Table 5.

Errors arose primarily in: (1) dispersed or implicit cues (e.g., specialty inferred from procedural descriptions), (2) complex negligence narratives involving

Table 5. Performance on Semantically Complex Fields (25-case subset)

Field	Precision	Recall	F1
Medical Specialty	0.82	0.78	0.80
Negligence Type	0.71	0.68	0.69
Compensation Amount	0.84	0.76	0.80
Appraisal (鑑定)	0.92	0.88	0.90

multiple allegations across sections, and (3) compensation amounts expressed in mixed formats (e.g., Mandarin numerals, hybrid long-form expressions).

Interestingly, the neural baseline exhibited higher recall on negligence cues but substantially lower precision due to overgeneralization—consistent with findings in other legal IE studies.

5.5 Error Analysis

A qualitative review of errors reveals three recurring phenomena:

- **Boundary drift:** Variants of " 事實及理由" occasionally merged argumentative and factual segments, causing partial misalignment in RE boundaries.
- **Lexical ambiguity:** Terms such as " 診治" can indicate either diagnostic or execution-stage negligence, requiring domain knowledge beyond surface matching.
- **Mixed numeral formats:** Compensation amounts expressed with nested Mandarin numerals led to under-segmentation (e.g., " 貳佰萬元整").

These insights suggest opportunities for hybridization, e.g., embedding semantic constraints or lightweight classifiers, without compromising the interpretability central to the rule-based design.

6 Conclusion and Future Work

This paper presented a deterministic rule-based pipeline for structuring Mandarin medical-litigation judgments using regular expressions (RE). The system demonstrates that, in domains where drafting conventions exhibit stable linguistic patterns, RE-based extraction offers a transparent and reproducible mechanism for converting unstructured judgments into machine-readable fields. The results show that such methods achieve high accuracy for structurally well-defined attributes, while performance on semantically complex fields reflects the inherent variability and implicitness of judicial writing.

The structured outputs generated through this pipeline provide a practical foundation for downstream empirical analysis, such as examining trends in medical disputes, studying reasoning patterns across courts, or constructing domain-specific legal datasets. These representations do not constitute a full formalisation of legal norms, but rather serve as an intermediate layer that enables subsequent computational tasks to operate on consistent, interpretable features.

Several directions for future work remain. First, integrating lightweight semantic validation or ontology-guided checks may help improve consistency across heterogeneous drafting styles. Second, hybrid approaches that combine rule-based templates with selective neural components could enhance coverage for fields where cues are highly dispersed or implicit, while still maintaining the auditability required in legal applications. Third, extending the pipeline to additional legal domains: such as criminal, administrative, or insurance disputes. They would allow a clearer assessment of its portability across different judgment genres.

Overall, this study highlights the value of interpretable, rule-based extraction methods in preparing Mandarin legal texts for computational analysis. By focusing on deterministic structuring rather than predictive modeling, the approach provides a reliable and legally transparent pathway for transforming narrative judgments into analyzable data, supporting future research in legal informatics, empirical studies, and corpus-based examination of judicial practices.

References

1. Bartoli, A., De Lorenzo, A., Medvet, E., Tarlao, F.: Inference of regular expressions for text extraction from examples. IEEE Transactions on Knowledge and Data Engineering **28**, 1217–1230 (2016). https://doi.org/10.1109/TKDE.2016.2557978
2. Bartoli, A., De Lorenzo, A., Medvet, E., Tarlao, F.: Active learning of regular expressions for entity extraction. IEEE Transactions on Cybernetics **48**, 1067–1080 (2018). https://doi.org/10.1109/TCYB.2017.2680466
3. Brauer, F., Rieger, R., Mocan, A., Barczynski, W.: Enabling information extraction by inference of regular expressions from sample entities. In: Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM). pp. 1285–1294 (2011). https://doi.org/10.1145/2063576.2063763
4. Li, J., Qian, L., Liu, P., Liu, T.: Construction of legal knowledge graph based on knowledge-enhanced large language models. Information **15**(11), 666 (2024). https://doi.org/10.3390/info15110666
5. Li, Y., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S., Jagadish, H.V.: Regular expression learning for information extraction. In: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 21–30 (2008). https://doi.org/10.3115/1613715.1613719
6. Lin, C., Cheng, P.: Legal documents drafting with fine-tuned pre-trained large language model (2024)
7. Lin, C., Cheng, P.: Assisting drafting of chinese legal documents using fine-tuned pre-trained large language models. Review of Socionetwork Strategies **19**, 83–110 (2025). https://doi.org/10.1007/s12626-025-00179-5
8. Liu, Q., Islam, M.K., Governatori, G.: Towards an efficient rule-based framework for legal reasoning. Knowledge-Based Systems **224**, 107082 (2021). https://doi.org/10.1016/j.knosys.2021.107082
9. Liu, Z., Chen, X., Wang, H., Liu, X.: Integrating regular expressions into neural networks for relation extraction. Expert Systems with Applications **252**, 124252 (2024). https://doi.org/10.1016/j.eswa.2024.124252
10. Ren, Y., Han, J., Lin, Y., Mei, X., Zhang, L.: An ontology-based and deep learning-driven method for extracting legal facts from chinese legal texts. Electronics **11**(12), 1821 (2022). https://doi.org/10.3390/electronics11121821

11. Si, Y., Zhou, W., Gai, J.: Research and implementation of data extraction method based on nlp. In: Proceedings of the 2020 IEEE 14th International Conference on Anti-counterfeiting, Security, and Identification (ASID). pp. 11–15 (2020). https://doi.org/10.1109/ASID50160.2020.9271745
12. Spositto, O., Bossero, J., Moreno, E., Ledesma, V., Matteo, L.: Lexical analysis using regular expressions for information retrieval from a legal corpus. In: Information Technology and Systems, pp. 312–324. Springer (2021). https://doi.org/10.1007/978-3-031-05903-$2_2$1
13. Xiao, C., Hu, X., Liu, Z., Tu, C., Sun, M.: Lawformer: A pre-trained language model for chinese legal long documents. AI Open **2**, 79–84 (2021). https://doi.org/10.1016/j.aiopen.2021.06.003
14. Yang, Z., Jiang, X.: A simplified application of regular expressions: With the extraction of chinese cultural terms as an example. In: Proceedings of the 2009 ISECS International Colloquium on Computing, Communication, Control, and Management. vol. 1, pp. 439–442 (2009). https://doi.org/10.1109/CCCM.2009.5268087
15. Yao, Z.J., Huang, D.G., Ji, X.Y.: Application of regular expressions to extraction of chinese cultural terms with their english translations **50**, 291–295 (2010)
16. Zhang, J., Zhang, Y.: Regular expression and application in information extraction. Computer Knowledge and Technology **5**(15), 3867–3868 (2009)

Using LLMs to Model Arguments in U.S. Supreme Court Briefs: Preliminary Report *

Heng Zheng[1], Dexter Williams[2], and Bertram Ludäscher[3]

[1] School of Information Science, University of Kentucky, USA hengzheng@uky.edu
[2] The Information School, University of Wisconsin-Madison, USA djwilliams22@wisc.edu
[3] School of Information Sciences, University of Illinois, Urbana-Champaign, USA ludaesch@illinois.edu

Abstract. Supreme Court briefs can impact the decision-making of Supreme Court cases. We show how to represent the briefs for a U.S. Supreme Court case as a formal argument model that relies on the briefs' existing structure. We explore how to generate argument models with large language models (LLMs) by using them to identify attacks between arguments.

Keywords: U.S. Supreme Court · large language model · computational argumentation · argument mining

1 Introduction

Supreme Court briefs can impact the decision-making of Supreme Court cases [4]. Some U.S. Supreme Court cases have received extensive attention outside of law, such as the *Roe v. Wade*[4] case being overturned, receiving over 70 million discussions on Twitter (now X) in about one year (January 2022 to January 2023) [3]. Helping the public understand how the Supreme Court makes decisions to mitigate misunderstandings of the legal decision-making process is important [13].

Computational argumentation methods can assist in analyzing Supreme Court briefs by identifying the arguments presented in the briefs and mapping the relationships between them. Here, we investigate:

1. How can we model U.S. Supreme Court briefs with computational argumentation?
2. Can we use LLMs to automatically generate the models?

We show how to model the Supreme Court briefs as argument models and explore the potential for LLMs to assist in the modeling process. This can assist non-experts in comprehending complex legal knowledge in the briefs, such as how different stakeholders (petitioners, respondents, and amici curiae) argue for their stances and against other parties.

* This paper is an extended version of our JURIX 2025 poster paper: "Modeling U.S. Supreme Court Briefs with Computational Argumentation".

[4] *Roe v. Wade*, 410 U.S. 113 (1973).

2 Related Work

Research on the U.S. Supreme Court cases focuses on the development of rules over time [9] using the Fourth Amendment's automobile exception, and oral arguments analysis [2,1]. AI techniques can also improve Supreme Court briefs analysis to advance understanding of what arguments from involved parties were considered when deciding the case, such as using machine learning to classify the ideology of briefs [5].

Transformer-based language models have been utilized to analyze legal documents, combining computational argumentation approaches by incorporating large language models with ASPIC+ to generate arguments [11] and using argument schemes with transformer-based natural language models to mine arguments in legal documents [8].

Using AI techniques, especially LLMs, to analyze Supreme Court briefs requires evaluating LLMs' performance in understanding the arguments in the brief. AI has shown its ability in argument summarization and completion using a benchmarking dataset of U.S. Supreme Court briefs [15]. AI's performance in identifying relationships between arguments varied: LLM-based approaches show good performance in distinguishing support and attack relationships [6]. However, in [6], no datasets from the legal domain were used for experimentation. In contrast, Graph Neural Network-based approaches did not perform well in identifying undercutting attacks [16]. Further investigation is still needed to determine whether LLMs can effectively identify attack relationships between arguments from opposite Supreme Court case briefs.

3 Modeling the U.S. Supreme Court Briefs with Formal Argumentation

We chose the *Acheson Hotels, LLC v. Laufer*[5] case as an example to illustrate how we model the Supreme Court briefs. Both the petitioner and the respondent received amicus briefs that supported them. We focused on the briefs filed during the merits stage instead of the certiorari stage, because not only do the judges use these briefs to help make their final judgment, but also because the petitioner and the respondent may not be opposed to having the case heard in the cert stage. Here is a short introduction to the case:

> "Deborah Laufer, a prolific litigant with physical disabilities and vision impairments, sued Acheson Hotels for failing to publish information about their accessibility on their website, which is required under the Americans with Disabilities Act (ADA).
> The district court dismissed the lawsuit, finding that Laufer lacked standing to sue because had no plans to visit the hotel and thus suffered no injury as a result of the lack of information on the website. The U.S. Court of Appeals for the First Circuit reversed, concluding that Laufer's lack of intent to book a room at the hotel operated by Acheson does not negate the fact of injury."[6]

We selected the following briefs from the petitioner, the respondent, and one amici curiae that supports the respondent:

[5] *Acheson Hotels, LLC v. Laufer*, No. 22–429 (U.S. Dec. 5, 2023).
[6] *Facts of the case* from *oyez.org*: https://www.oyez.org/cases/2023/22-429

Part of the Respondent Brief

ARGUMENT 20

I. *Havens Realty* forecloses Acheson's challenge to Ms. Laufer's standing. 22

A. *Havens Realty*'s reasoning squarely applies to Reservation Rule testers. 23

B. Ms. Laufer experienced actionable discrimination when she encountered an unlawful barrier to her full and equal enjoyment of Acheson's reservation services. 27

C. Acheson's other efforts to distinguish this case from *Havens Realty* fail. 32

II. *Havens Realty* was correctly decided. 33

.....

CONCLUSION

The judgment below should be vacated solely on the ground that Ms. Laufer's dismissal of her complaint mooted her appeal.

Part of the Petitioner Brief

ARGUMENT 14

I. TO ESTABLISH STANDING, LAUFER MUST DEMONSTRATE A CONCRETE, PARTICULARIZED, AND FORWARD-LOOKING INJURY. 14

II. LAUFER CANNOT ESTABLISH STANDING BASED ON HER ALLEGED INFORMATIONAL INJURY. 16

A. Laufer's alleged injury is not concrete. 16

B. Laufer's alleged injury is not particularized. 22

C. *Havens Realty* does not assist Laufer. 26

D. This Court's "informational standing" cases do not support Laufer. 33

.....

IV. LAUFER'S INTEREST IN ENFORCING THE ADA DOES NOT GIVE HER STANDING. 47

.....

CONCLUSION

The judgment of the First Circuit should be reversed.

Part of the Amicus brief (support respondent) from Disability Rights Education & Defense Fund, et al.

ARGUMENT 4

.....

II. THE FACT THAT A PLAINTIFF IS A TESTER DOES NOT DEPRIVE HER OF STANDING. 15

.....

III. TESTING IS ESSENTIAL TO ENFORCEMENT OF THE ADA. 19

.....

CONCLUSION

Amici agree with both parties and the United States that the Court should dismiss this case as moot. *See* Suggestion of Mootness. If it does not, then it should affirm the opinion of the First Circuit.

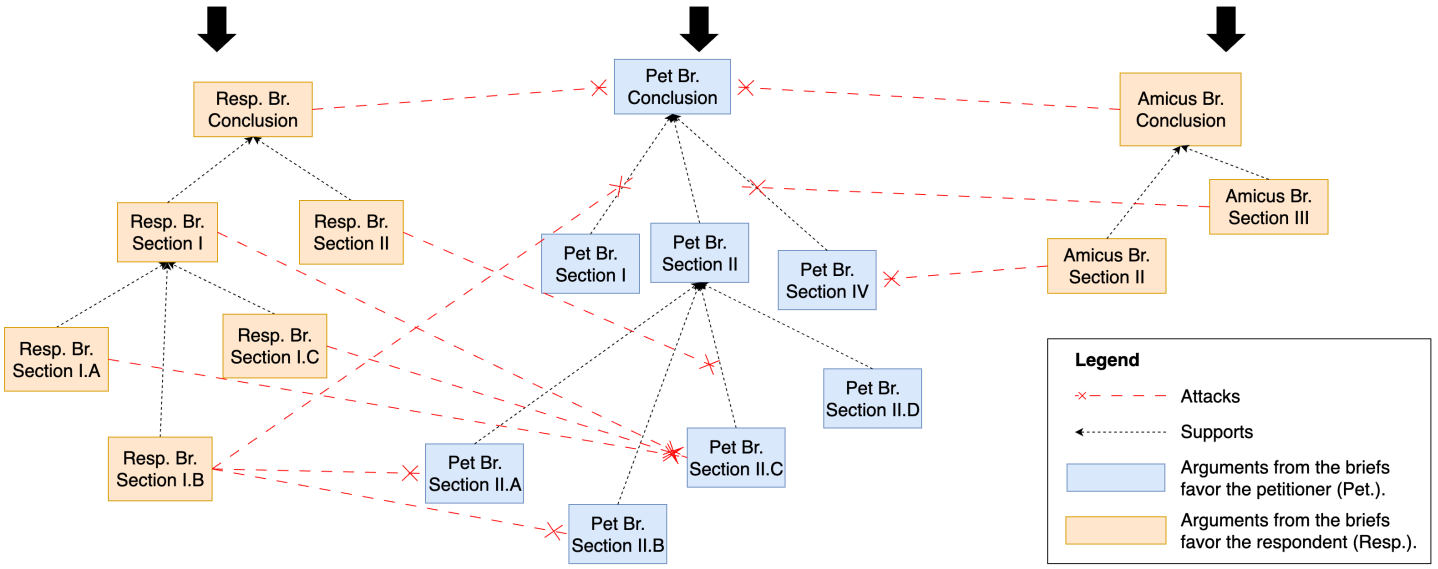


Fig. 1: Modeling excerpts from the petitioner brief (middle), the respondent brief (left), and an amicus brief (right) that supports the respondent as an argument model

1. The petitioner's (Acheson Hotels) brief[7] filed at the beginning of the merits stage.
2. The respondent's (Laufer) brief[8] that was filed in response to the petitioner's brief.
3. An amicus brief[9] filed by *Disability Rights Education & Defense Fund et al.* that supports the respondent and against the petitioner. *Disability Rights Education & Defense Fund* is an advocacy group for individuals with disabilities.

The briefs follow a strict structure [12]. Figure 1 shows the table of contents of sample briefs and how we model the *Acheson Hotels* case's briefs in an argument model. We model each brief as an argument. The subsections under the ARGUMENT section are premises that support the conclusion (content in the CONCLUSION section). Here, we deem each subsection to *support* the section and/or the conclusion independently, and the inference from the subsection (as a premise) to the section/conclusion is *defeasible*. For example, the part of the petitioner's brief shown in Figure 1 has Sections I, II, and IV that support the conclusion, and Section II is further supported by Sections II.A, II.B, II.C, and II.D. Arguments, such as the one from "Pet Br. Section II.C" to "Pet Br. Section II", are defeasible, because other arguments can attack them.

We consider attacks between arguments if they respect the temporal filing order, thus capturing how stakeholders argue in response to other parties. Therefore, attacks only exist from a later-filed brief to an earlier-filed brief, and the two briefs should not favor the same sides. The timeline for filing briefs during the merits stage is as follows (see Rules 24, 25, and 37 in [14]): The petitioner first files a brief to argue on the merits. Then, the respondent (the opposite party of the petitioner) files a brief to respond to the petitioner's argument. After that, the petitioner may reply to the respondent's argument. Amici curiae that support each party can also file briefs to present their arguments during the merits stage. By keeping the order in which attacks were made, we model the decision-making process (briefings can attack earlier but not future arguments).

Attacks are classified as *undermining*, *rebutting*, or *undercutting*: Undermining is an attack on a premise, rebutting is an attack on a conclusion, while undercutting is an attack on the inference relation between premise(s) and conclusion [10]).

Arguments in the respondent's brief and the amicus brief in Figure 1 attack the argument in the petitioner's brief. The respondent's brief concludes that the First Circuit's judgment should be "vacated," which is a rebuttal of the petitioner's brief's conclusion: the judgment should be "reversed."

Respondent brief's Section I: "*Havens Realty*[10] forecloses Acheson's challenge to Ms. Laufer's standing" is an undermining attack of the petitioner brief's Section II.C: "*Havens Realty* does not assist Laufer." Because Section II.C is modeled as a premise in the argument of the petitioner's brief. The *Havens Realty* case is a key precedent in the *Acheson Hotels* case. "*Havens Realty* forecloses Acheson's challenge to Ms. Laufer's standing" directly against the petitioner's point on *Havens Realty does not assist Laufer*.

In contrast, Section II in the respondent's brief ("*Havens Realty* was correctly decided") is an undercutting attack of the petitioner's Section II.C. Because it does not

[7] Brief of petitioner Acheson Hotels, LLC
[8] Brief of respondent Deborah Laufer
[9] Brief amici curiae of *Disability Rights Education & Defense Fund*
[10] *Havens Realty Corp. v. Coleman*, 455 U.S. 363 (1982).

directly argue whether *Havens Realty* supports Laufer (respondent) or not, instead it argues whether the *Havens Realty* was correctly decided.

4 Using LLMs to Generate the Argument Model

Constructing an argument model for the briefs filed in the *Acheson Hotels* case is time-consuming. LLMs can help build up the model more efficiently.

To use AI for the argument model construction, we attempted to use LLMs to identify attacks between the sections from opposing briefs (e.g., respondent vs petitioner). This allows us to generate structured argument models composed of:

- Arguments with premises as the summary of the content in each subsection under the ARGUMENT section, and conclusions as the content of the CONCLUSION section;
- Attacks between the arguments.

For attack-relation mining, `Mixtral-8x7B` and `Mistral7B` have shown strong performance on relation-based argument mining (RbAM) tasks in non-legal datasets [6]. However, RbAM has not yet been applied to U.S. Supreme Court briefs. Given its availability as an open-weight model and its prior use in [6], we chose `Mistral7B` to evaluate Task 2, reusing the prompt from [6] with slight adaptation:

Prompt for attack relation identification

In this task, you will be given two arguments and your goal is to classify the relation between them as either "attack" or "no" based on the definitions below.
Attack: It is an argument that contradicts or opposes the parent argument.
No: It is an argument that has no relation to the parent argument. Answer with one word.

Arg1: Parent Argument(parent)
Arg2: Child Argument(child)
Relation:

Our annotation includes the petitioner's brief, the respondent's brief, the reply brief of the petitioner[11], and the amicus brief filed by *Disability Rights Education & Defense Fund, et al.* Our annotation is based on the section headings of the briefs under the ARGUMENT section, because these can be viewed as the summary of the whole sections. We compared the section headings and checked whether they attack each other. The attacks can be further characterized as undermining or rebutting, depending on whether they are premises or conclusions. Here annotations are used to show the potential of LLMs for identifying attack relations, not for evaluation purposes. We also have not identified undercutting attacks between arguments in our initial experiments.

While a lack of annotated data currently makes meaningful evaluation challenging, the initial output shows `Mistral7B`'s performance varied. As an example, `Mistral7B`

[11] Reply of petitioner Acheson Hotels, LLC

successfully identified the conflict between "*Havens Realty* forecloses Acheson's challenge to Ms. Laufer's standing" from the respondent brief and "*Havens Realty* does not assist Laufer" from the petitioner brief. However, `Mistral7B` missed other undermining attacks from the respondent's brief to "*Havens Realty* does not assist Laufer" from the petitioner brief, such as "*Havens Realty*'s reasoning squarely applies to Reservation Rule testers," where `Mistral7B` failed to connect Laufer with "Reservation Rule testers."

5 Discussion and Future Work

Based on the first author's experience annotating the briefs in the *Acheson Hotels* case, a full expert analysis of arguments in U.S. Supreme Court briefs can be very time-consuming. The structure of the U.S. Supreme Court briefs allows us to model the briefs as an argument model. It can be difficult for non-legal experts to determine whether the existing headings in the briefs conflict with headings from the opposite briefs. For example, the section heading "*This [Supreme] Court's 'information standing' cases do not support Laufer*" from the petitioner's brief does not indicate what the 'information standing' cases are, and thus make it hard for non-legal experts to determine whether the *Havens Realty* case mentioned by the respondent is one of them.

The section headings in the briefs may also be too abstract for LLMs to correctly identify all attack relations. In our initial attempt, `Mistral7B` can identify the conflict between "*Havens Realty* forecloses Acheson's challenge to Ms. Laufer's standing" from the respondent and "*Havens Realty* does not assist Laufer" from the petitioner, but failed to connect Laufer with "Reservation Rule testers". Therefore, more context will be helpful for LLMs to conduct relation-based argument mining, such as a more detailed summarization by LLMs of the subsections under the ARGUMENT section of the briefs.

How to use LLM to identify undercutting attacks between arguments still needs further investigation. Based on similar tasks performed on a dataset of social media [16], identifying undercutting attacks can be a challenge.

Future work is needed to properly evaluate LLMs' performance. To this end, we will create a legal expert-annotated corpus of argument relations in Supreme Court briefs for the LLM-based attack identification, then use the dataset to evaluate the performance of different LLMs for attack relation mining on Supreme Court briefs and compare the performance of different prompts. We will experiment with different LLMs and see if incorporating other techniques can help identify the attacks, such as factor extraction [7]. We will evaluate whether the relation mining performance depends on the level of detail in the argument summarization. We also plan to explore methods to present insights to the public based on the argument models. While our initial model was inspired by ASPIC+ [10], subsequent work will develop a more formal, structured model of Supreme Court briefs. This will allow us to analyze the briefs using the computational semantics of (structured) argumentation frameworks.

6 Summary and Conclusions

Through a preliminary study, we have investigated how to combine the strengths of LLMs and computational argumentation to assist non-legal experts in analyzing the U.S. Supreme Court briefs. We have modeled the briefs as argument models: The sections in a brief can form the premises and conclusions of an argument, and attacks can be derived between the briefs that do not favor the same party. We have explored the use of LLMs to identify the attacks between the arguments in order to generate argument models more efficiently. While a lack of annotated data currently makes meaningful evaluation challenging, we find the initial results promising. Future work is needed to evaluate LLMs' performance on attack relation identification systematically.

References

1. Al-Abdulkarim, L., Atkinson, K., Bench-Capon, T.: From Oral Hearing to Opinion in the U.S. Supreme Court. In: Legal Knowledge and Information Systems. pp. 1–10. IOS Press (2013). https://doi.org/10.3233/978-1-61499-359-9-1
2. Ashley, K., Pinkwart, N., Lynch, C., Aleven, V.: Learning by diagramming Supreme Court oral arguments. In: Proceedings of the 11th international conference on Artificial intelligence and law. pp. 271–275. ICAIL '07, Association for Computing Machinery, New York, NY, USA (Jun 2007). https://doi.org/10.1145/1276318.1276370
3. Chang, R.C., Rao, A., Zhong, Q., Wojcieszak, M., Lerman, K.: #RoeOverturned: Twitter Dataset on the Abortion Rights Controversy. Proceedings of the International AAAI Conference on Web and Social Media **17**, 997–1005 (Jun 2023). https://doi.org/10.1609/icwsm.v17i1.22207
4. Corley, P.C.: The Supreme Court and Opinion Content: The Influence of Parties' Briefs. Political Research Quarterly **61**(3), 468–478 (Sep 2008). https://doi.org/10.1177/1065912907306474
5. Evans, M., McIntosh, W., Lin, J., Cates, C.: Recounting the Courts? Applying Automated Content Analysis to Enhance Empirical Legal Research. Journal of Empirical Legal Studies **4**(4), 1007–1039 (2007). https://doi.org/10.1111/j.1740-1461.2007.00113.x
6. Gorur, D., Rago, A., Toni, F.: Can Large Language Models perform Relation-based Argument Mining? In: Rambow, O., Wanner, L., Apidianaki, M., Al-Khalifa, H., Eugenio, B.D., Schockaert, S. (eds.) Proceedings of the 31st International Conference on Computational Linguistics. pp. 8518–8534. Association for Computational Linguistics, Abu Dhabi, UAE (Jan 2025), https://aclanthology.org/2025.coling-main.569/
7. Gray, M., Savelka, J., Oliver, W., Ashley, K.: Using LLMs to Discover Legal Factors. In: Legal Knowledge and Information Systems. pp. 60–71. IOS Press (2024). https://doi.org/10.3233/FAIA241234
8. Habernal, I., Faber, D., Recchia, N., Bretthauer, S., Gurevych, I., Spiecker genannt Döhmann, I., Burchard, C.: Mining Legal Arguments in Court Decisions. Artificial Intelligence and Law **32**(3), 1–38 (Sep 2024). https://doi.org/10.1007/s10506-023-09361-y
9. Henderson, J., Bench-Capon, T.: Describing the Development of Case Law. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law. pp. 32–41. ICAIL '19, Association for Computing Machinery, New York, NY, USA (Jun 2019). https://doi.org/10.1145/3322640.3326697
10. Modgil, S., Prakken, H.: The ASPIC+ Framework for Structured Argumentation: A Tutorial. Argument & Computation **5**(1), 31–62 (Jan 2014). https://doi.org/10.1080/19462166.2013.869766

11. Park, S., Choi, A., Park, R.: Objection, Your Honor!: An LLM-Driven Approach for Generating Korean Criminal Case Counterarguments. Artificial Intelligence and Law (Feb 2025). https://doi.org/10.1007/s10506-025-09432-2
12. Schweitzer, D.: U.S. Supreme Court Brief Writing Style-Guide Developments. Journal of Appellate Practice and Process **19**(1), 129–156 (2018), https://heinonline.org/HOL/P?h=hein.journals/jappp19&i=137
13. Sullivan, B., Feldbrin, R.: The Supreme Court and the People: Communicating Decisions to the Public. University of Pennsylvania Journal of Constitutional Law **24**(1), 1–92 (2022), https://heinonline.org/HOL/P?h=hein.journals/upjcl24&i=2
14. Supreme Court of the United States: Rules of the Supreme Court of the United States. Tech. rep., Supreme Court of the United States (2023), https://www.supremecourt.gov/filingandrules/2023RulesoftheCourt.pdf
15. Woo, J., Hashemi Chaleshtori, F., Marasovic, A., Marino, K.: BriefMe: A Legal NLP Benchmark for Assisting with Legal Briefs. In: Che, W., Nabende, J., Shutova, E., Pilehvar, M.T. (eds.) Findings of the Association for Computational Linguistics: ACL 2025. pp. 13139–13190. Association for Computational Linguistics, Vienna, Austria (Jul 2025). https://doi.org/10.18653/v1/2025.findings-acl.681
16. Ye, Y., Teufel, S.: Computational modelling of undercuts in real-world arguments. In: Ajjour, Y., Bar-Haim, R., El Baff, R., Liu, Z., Skitalinskaya, G. (eds.) Proceedings of the 11th Workshop on Argument Mining (ArgMining 2024). pp. 59–68. Association for Computational Linguistics, Bangkok, Thailand (Aug 2024). https://doi.org/10.18653/v1/2024.argmining-1.6

Legal Texts to Legal Data: LLM-Based Attribute Extraction from Court Verdicts

Ivana Kvapilíková[2 [0000−0003−1479−3294]], Jan Černý[1,2 [0009−0006−4873−8233]], Vojtěch Pour[1 [0009−0005−4075−6579]], Tomáš Knap[1 [0009−0009−2085−9182]], Klára Bendová[2 [0000−0001−8002−6566]], Jaromir Savelka[3 [0000−0002−3674−5456]], and Jakub Drápal[1 [0000−0001−9455−9013]]

[1] Faculty of Law, Charles University, Prague, Czechia
[2] Faculty of Mathematics and Physics, Charles University, Prague, Czechia
[3] School of Computer Science, Carnegie Mellon University, Pittsburgh PA, USA

Abstract. Court verdicts are a valuable legal source of information about parties' behavior, especially in criminal cases. The unstructured format of presenting the behavior is, however, unsuitable for systematic analysis. We overcome this obstacle by building a structured dataset of key attributes derived from the texts, namely from the factual statement – description of the criminal behavior – contained in most criminal verdicts across continental Europe. The attribute list is partly defined by legal experts and partly expanded from the data itself in a LLM-based pipeline. We investigate whether expert-provided attributes are necessary or if data-driven discovery is sufficient. Our findings highlight both the promise and current limitations of LLMs in legal text processing and suggest how structured representations of criminal behaviors could enable downstream applications in legal analytics.

Keywords: Large Language Models (LLMs)· Attribute Extraction· Legal Text Processing· Traffic-related Crimes· Court Verdicts

1 Introduction

Court criminal verdicts represent a rich source of information about criminal behavior, which in turn can tell us much about the functioning of the justice system, especially about the factors impacting imposition of sentences. However, descriptions of criminal behaviors are typically available only in an unstructured form, which makes systematic analysis difficult. For researchers and policymakers who wish to conduct statistical studies of crimes and punishments, a structured representation of criminal behaviors is essential. When enriched with administrative data, such representations allow not only for the identification of broad patterns, but also for detailed investigations of how specific attributes (e.g. criminal history, circumstance of the conduct) interact with each other and influence judicial decisions.

A key challenge in transforming factual statements into structured datasets is preserving as much original information as possible while defining a meaningful set of attributes. Expert knowledge can guide this process by specifying theoretically relevant factors, while data-driven methods may reveal unexpected but valuable dimensions.

Our work combines these perspectives and assesses whether expert-defined attributes are indispensable or if data-driven discovery alone can suffice. Our attribute extraction process relies on large language models (LLMs), which have shown remarkable progress in natural language processing tasks in the legal domain [5, 9, 11].

The contributions of this paper are the following:

1. We construct a dataset of Slovak court verdicts concerning traffic-related offenses, structured into key attributes, and make it publicly available.[4]
2. We develop an attribute-extraction pipeline that can be adapted to new crime categories and applied across different countries.
3. We compare expert-based and data-driven approaches to attribute definition, assessing their respective benefits and limitations.

We view our work as an incremental step toward evidence-based legal analytics, enabling the analysis of sentencing practices with unprecedented depth and breadth across all common offenses. This approach has the potential to transform the study of sentencing by providing systematic insights into how sentences are imposed. Beyond national contexts, it also opens the door to cross-country comparisons: By aligning cases on the basis of factual similarities, rather than relying on differing legal definitions, we can meaningfully compare sentencing practices across jurisdictions.

2 Related Work

Prior approaches working with judicial texts have largely focused on end-to-end legal judgment prediction (LJP), where researchers model the problem as a classification task and use text features [2], text embeddings [4] or in-context learning capability of LLMs [13] to predict the court's outcome. In [1], the authors extract attributes from criminal case proceedings of the Supreme Court of India and show that including the extracted spans in the LLM prompt increases accuracy in the LJP task. In this work, our aim is not the prediction of the sentencing imposed, but rather its explainability. We extract structured attributes in the form of a flat table to be able to run a statistical regression model and quantify the impact of individual variables.

Although we construct a dataset with a simple tabular structure, we take inspiration from scholars who approach information extraction from the direction of legal ontologies and construct knowledge graphs (KGs) [12, 6] – a semantically rich representations of actors, events, and circumstances. D'Amato et al. [6] follow a bottom-up strategy, first extracting entities and relations from legal texts and then inducing an ontology and populating a KG from these data. In contrast, a top-down approach—where the ontology is designed in advance—requires substantial legal expertise and is difficult to apply uniformly in criminal law due to the heterogeneity of case facts and the level of abstraction needed to model them ontologically. Sovrano et al. [12] combine the two approaches as they bottom-up extract KGs using a dependency parser, top-down create an ontology and map the two together. Although the task of attribute extraction is slightly different, we also divide our approaches into expert-based and data-driven, depending

[4] https://github.com/kvapili/legal-attribute-extraction

Category	Dev set	Test set	Precision	Recall	Final dataset
Violation of a driving ban	38	10	100%	97.4%	369
Drug-impaired driving	156	40	97.0%	99.0%	1,775
Traffic-related injury	92	23	97.8%	96.7%	983
Other traffic-related offense	1	1	n/a	n/a	103
Non-traffic-related offense	895	224	98.5%	99.5%	6,740
Unclear	18	2	n/a	n/a	30
Total	**1,200**	**300**			**10,000**

Table 1. Overview of offenses by category with dev/test gold counts, LLM categorization performance (precision and recall) against human annotators measured on the test set, and final dataset distribution.

whether we start from expert instructions or we generate these instructions based on data. We plan to add hierarchical relations to our data in future work, especially if we encounter crime categories too complex to be captured by a flat table.

The flexibility of LLMs makes them appealing for information extraction, but concerns remain about hallucinations, reliability and the need for domain adaptation. The key aspect is a precise and knowledge-rich design of the prompts which however must be tuned to each model specifically [8]. Gray et al. [7, 8] successfully use LLMs for identifying factors in judicial texts. Their factors are more abstract and broad than our attributes (e.g. nervous behavior or criminal history), but they show that a semi-automated LLM pipeline with a human in the loop is effective and correlates with human-identified factors. Adhikary et al. [1] model the extraction as a sequence-labeling task and use LLMs to produce weakly supervised samples to train their sequence-labeling model. Zin et al. [14] extract attributes from contracts in a two-step summarize-and-extract method and demonstrate strong performance even with the now-outdates GPT-3.5 model.

3 Data and Categorization

In this work, we focus on Slovak court verdicts, which have the advantage of being publicly available. We pilot our approach on verdicts dealing with traffic-related crimes, both because of their prevalence and because they form a relatively well-defined domain. While this paper concentrates on traffic cases, our long-term objective is to apply the same methodology to other categories of crimes, gradually building toward a general framework for large-scale legal analytics.

We work with a dataset of 11,500 court verdicts obtained from the Ministry of Justice of the Slovak Republic[5]. This represents approximately 10% of the full collection available for download, which we intend to process in future work. For our purposes, we focus solely on the sections of the verdicts that describe the facts and circumstances of the case, as these are the basis for attribute extraction. The factual statements were extracted using a combination of regular expressions and prompted LLMs as described in [3].

[5] https://web.archive.org/web/20230205033430/https://obcan.justice.sk/opendata

Attribute	Description	Validation Rule
substanceUsed	Drug or substance detected	alcohol, cannabis, methamphetamine, amphetamine, cocaine, mdma, hashish, morphine, barbiturate, benzodiazepine, methadon, heroin, other
amountUsed	The amount of drug or substance detected	`float` mg/l, `float` g/kg
measuringDevice	Measuring device used for testing	–
dateOfOffense	Date when the offense occurred	`DD.MM.YYYY`
timeOfOffense	Time when the offense occurred	`HH:MM`
placeOfOffense	Place where the offense occurred	–
typeOfRoad	Type of road where the offense occurred	urban, motorway, class_I, class_II, class_III, other
checkedBy	Who conducted the inspection	–
reasonForCheck	Reason why the inspection was carried out	suspected_intoxication, traffic_check, speeding, traffic_violation, traffic_accident, other
vehicleType	Type of vehicle driven	car, van, truck, bus, motorcycle, other
measuringMethod	Method used for testing	breath, blood, urine
measuringTime	Time of measurement	`HH:SS`
refusedTesting	Whether testing was refused	yes, no
licenseStatus	Status of driver's license	valid, banned, none

Table 2. Attribute schema with corresponding descriptions and validation rules for drug-impaired driving.

To select the relevant cases from our dataset, we first applied an **LLM-based classification pipeline**, which identified traffic-related crimes and assigned them to specific subcategories we predefined. We used the GPT-5-mini model configured in the medium-reasoning mode with low verbosity. To support future large-scale data processing, we opted for the mini version of OpenAI's most capable model, which delivers high accuracy at substantially lower cost.

The subcategories of traffic-related crimes are presented in Table 1 along with their distribution across the data splits. We obtained human labels for 1,500 verdict statements to tune and evaluate our pipeline. Each statement was annotated by at least two law students who agreed on 91.3% of the cases; in the remaining 131 instances, an expert label was assigned to resolve the disagreement. The LLM instructions were tuned on a portion of the dev set, and the test set was reserved for a blind evaluation. Our LLM-based categorization pipeline achieved precision and recall values exceeding $\sim$97% in the categories assessed against human annotators.

We illustrate our attribute-extraction approach on the category of *Drug-impaired driving*, comprising 1,971 court cases. The extraction pipeline was developed using a subset of 196 cases and subsequently applied to the remaining 1,775 cases. For evaluation, we drew a random sample of 200 cases from the final dataset.

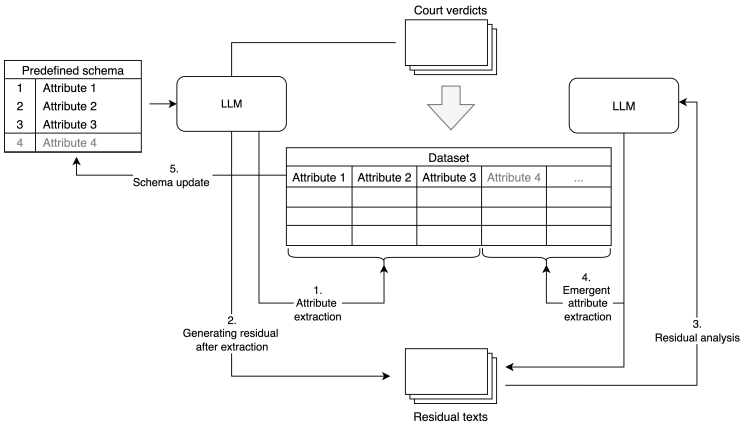


Fig. 1. Attribute extraction pipeline for the development stage.

4 Methodology

Our extraction pipeline employs an iterative sequence of LLM calls to the Gemini 2.5 Flash model by Google DeepMind, where attribute values are extracted and the attribute schema is dynamically expanded with newly identified candidates. This design, depicted in Figure 1, enables schema evolution in parallel with value extraction, ensuring that both predefined and emergent attributes are systematically captured.

The first input to our pipeline is the **predefined attribute schema** (the first two columns of Table 2) compiled by experts (lawyers), drawing on penal code commentaries, textbooks, and prior research on traffic offenses and sentencing. In addition, experts read through multiple factual statements and identified potential attributes to be extracted.

The second input are the factual statements that are processed in batches of N samples in I iterations. Each iteration consists of the following steps:

- **Attribute extraction.** In the initial step, the LLM is prompted with the attribute schema and tasked with extracting corresponding values from the input text. The results are returned as a structured JSON object in which the relevant attributes are populated if the information is available.
- **Generating residual after extraction.** The second LLM call identifies residual information in the court verdict that is not yet covered by the attribute list. This information is stored in a placeholder attribute called *"residual"* and accumulated across batches and iterations.
- **Residual analysis & Emergent attribute extraction.** After processing the batch of N samples, the accumulated *residuals* are provided to an LLM, which is prompted to propose new attribute names given the context of the existing attributes. Candidate attributes occurring in more than r fraction of cases are retained for extraction and the *residuals* are updated accordingly.
- **Schema update.** We add the new attribute names to the original attribute list and the next batch of verdicts is processed with the updated attribute schema.

You extract values for attributes from text. Return only information from the text, never deduce or make anything up. If the value for attribute is not provided, return 'n/a'. The date format is `DD.MM.YYYY`. The time format is `HH:MM`. Decimal numbers are rounded to two digits after the decimal point: #.##. Format your output as JSON. The values mut be in the original language - Slovak. Extract the following attributes: *[attribute_schema]*
What other information was there in the statement but is not covered by the attributes? Write one sentence containing the extra information. Output a JSON with one attribute: 'residual' where the value is the sentence with extra information. Only output information from the original text. The sentence mut be in the original language - Slovak.
These are the residual texts from court verdicts after extracting the following attributes *[attribute_schema]* What other pieces of information do you see repeatedly in the residual texts? Return a JSON with the emergent attribute names as keys and descriptions as values. The attributes should be as granular as possible, but do not repeat the attributes that were already defined.

Table 3. System prompt for attribute extraction (top), user prompt for generating residuals (middle) and system prompt for residual analysis (bottom)

When instructing the LLM to extract attributes, we explicitly emphasize the need for literal text excerpts rather than inferring knowledge. For subsequent data analysis, however, it is beneficial to **standardize** these values and cluster them into categories where appropriate. Therefore, we run a second stage extraction with the complete set of attributes and expert-based validation rules (2) which constrain attribute formats and map values to predefined categories. We also instruct the model to separate values by a semicolon in the case of multiple values per attribute (e.g. multiple substances used or measurements of the same substance).

The prompts are detailed in Table 3. Although the prompts were further tuned for the specific model used in our experiments, the overall extraction method remains model-agnostic and can be applied with any sufficiently capable LLM. To support replicability, the code is available online.[6]

5 Results

5.1 Discovering Attribute Names

We first applied the expert-based approach with the predefined attributes listed in the left part of Table 4. The right part of the same table lists the attributes that emerged from the data after the expert-defined attributes were accounted for. We tuned this selection on batches of 20 cases ($N = 20$) over 8 iterations ($I = 8$) with the requirement that the emergent attributes are populated in at least 15% of the dataset ($r = 0.15$).

In a second experiment, we gave up our expert knowledge and started the extraction procedure with a single predefined attribute, `dateofoffense`. Even under this minimal guidance, the data-driven approach produced an attribute schema that was nearly

[6] https://github.com/kvapili/legal-attribute-extraction

Expert-defined	Populated %	Validated %	Data-driven	Populated %	Validated %
timeOfOffense	100.00	92.10	placeDetails	99.49	n/a
dateOfOffense	99.49	93.02	vehicleType	98.92	100.00
placeOfOffense	98.47	n/a	measuringMethod	91.84	100.00
substanceUsed	95.92	89.60	measuringTime	61.73	92.70
checkedBy	88.27	n/a	lawBroken	15.10	n/a
amountUsed	87.24	97.35	refusedTesting	8.61	100.00
typeOfRoad	82.65	100.00			
measuringDevice	58.16	n/a			
reasonForCheck	30.61	100.00			

Table 4. Expert-defined vs. data-driven attributes with percentages of populated and validated cases out of the evaluation set of 200 verdicts.

equivalent in scope, resulting in 18 attributes compared to the 15 identified in the earlier approach. Without being restricted by an initial predefined schema, the LLM advanced in a systematic manner, anticipating the underlying structure of the dataset and constructing a slightly more granular and coherent attribute list. For instance, it differentiated three distinct attributes to capture location (`offenseLocation_Municipality`, `offenseLocation_Streetroad`, `offenseLocation_DistrictRegion`) and disaggregated the notion of substance of impairment into `impairmentValue`, `impairmentUnit`, and `impairmentMedium`. The candidate attributes often comprised redundant and overly granular proposals (e.g. `driverConsumedSubstancePreDriving`, `alcoholMeasurementTestPurpose`) but these got discarded upon verification of the population rate r.

The success of the data-driven approach is greatly attributable to the fact that we are dealing with a very consistent category of offenses where the factual statements of the verdicts have a similar structure in all cases of drug-impaired driving. We suspect that the expert knowledge will be more valuable in more complex categories, e.g. traffic-related offenses causing injuries.

5.2 Validation of Attribute Extraction

In the final run of our extraction pipeline, we worked with a full schema comprising both expert-defined and emergent attributes. We distinguish between attributes that are free-text strings and those that take standardized values (see Table 2).

We first applied a set of attribute-specific validation rules, defined in Table 2, to the extracted values in order to verify the LLM's compliance with instructions and obtain a preliminary indication of where the extraction was problematic. The results are summarized in Table 4. We see that the constraint validation was 100% successful for all standardized attributes except for `substanceUsed`, where the LLM often retained Slovak wording instead of the standardized English values (e.g. *alkohol* instead of *alcohol*). We manually fixed these errors before further evaluation and we believe that in the future this issue can be mitigated by prompt refinement. For other attributes, most cases of validation breach were due to verdict anonymization where sometimes even

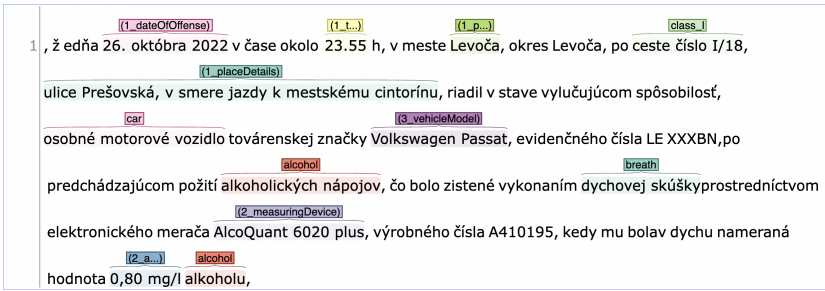


Fig. 2. Screenshot from the Inception annotation tool with labeled sequences on a part of the factual statement.

date/time values or vehicle types are replaced by `XXX` values and consequently fail our format check.

5.3 Manual Evaluation of Attribute Extraction

We conducted a manual evaluation of the extracted attributes on 200 randomly selected verdicts, asking law students to follow the same extraction instructions that were given to the LLM. The annotation was done in the Inception software [10]. The annotators were shown the factual statements of the court verdicts and they were asked to identify spans corresponding to the values of requested attributes. For standardized attributes, they were also asked to select a value from a predefined list. It was not compulsory to identify all attributes. The annotation was carried out by five law students in their 3rd to 5th year of study, with each of the 200 verdicts annotated independently by two students.

The main challenge of the manual annotation was the cognitive load imposed on the annotators by the great number of attributes we needed them to identify and the ambiguity of instructions for certain categories. This resulted in a low inter-annotator agreement (IAA) often due to missed attributes and a necessity for an expert decision to establish the ground truth.

We report both the inter-annotator agreement and the agreement between the LLM and the ground truth for most attributes in Table 5. We measure accuracy and Cohen's Kappa for attributes whose values are coded as elements of a list. For span attributes with numerals (time, date, amount used), we measure accuracy based on exact match in numerical characters.

For several attributes, the LLM agreement is high (accuracy above 90%) and even surpasses the IAA of human annotators, as the LLM (unlike humans) never misses a requested attribute due to fatigue or lack of attention. Furthermore, the LLM can use its internal knowledge to assign an appropriate category (e.g. recognizing that Fiat Ducato is a van rather than a passenger car) which human annotators might not posses.

After analyzing the results, we conclude that lower accuracies are mostly due to unclear instructions where nor humans neither the LLM can reliably fulfill the task. We further observed that LLMs tend to make deductions beyond the given instructions.

Attribute name	Annot 1 vs. Annot 2		LLM vs. Ground Truth	
	Accuracy	Cohen's κ	Accuracy	Cohen's κ
timeOfOffense	83.66	n/a	94.55	n/a
dateOfOffense	85.64	n/a	99.50	n/a
measuringTime	75.25	n/a	81.19	n/a
amountUsed	76.73	n/a	90.10	n/a
reasonForCheck	53.96	25.11	36.63	22.66
vehicleType	74.75	33.07	91.09	63.27
substanceUsed	66.34	34.54	72.28	36.06
licenseStatus	93.07	58.44	92.08	61.45
typeOfRoad	53.96	35.96	47.03	32.05
refusedTesting	98.02	84.73	98.02	85.64
measuringmethods	66.83	35.79	74.26	39.35

Table 5. Inter-annotator agreement between two annotators (left) and between an expert human annotator and LLM (right)

For example, in the absence of a clearly stated reason for the check (85% of cases), the LLM often inferred the value of `suspected_intoxication`, even though it could have been a standard `traffic_check`. Similarly, for the attribute `typeOfRoad`, the LLM frequently deduced that the offense occurred on an urban road, despite this not being explicitly mentioned. While in the latter case such inference was mostly correct (and potentially valuable for enhancing dataset completeness) in the former case it represents a risk of hallucination that we aim to mitigate through further prompt refinement.

Overall, we find that agreement levels are high for attributes that are explicitly stated in the text or defined with complete clarity. In contrast, attributes with ambiguous instructions (such as the distinction between a routine traffic violation and suspected intoxication as the reason for a traffic check) showed lower agreement, both among human annotators and between the LLM and the ground truth. In future iterations, we plan to refine the annotation guidelines to enhance inter-annotator consistency and improve overall reliability. Nevertheless, we consider these findings a solid first step toward developing a fully automated solution.

6 Conclusion and Future Work

We introduced a pipeline for extracting structured information from court verdicts by combining expert knowledge with data-driven methods. Our contributions include (i) the design of a multi-stage LLM-based framework for attribute extraction, (ii) the integration of expert-defined schemas and validation rules to guide and constrain extraction, (iii) an approach for discovering emergent attributes through iterative refinement of residual information and (iv) an evaluation of LLM ability to follow instructions for attribute extraction.

Looking ahead, we plan to extend this work in several directions. First, we aim to scale the approach to additional categories of criminal offenses (not only traffic-related). Second, we want to experiment with open-source LLMs deployed locally to

be able to use them on sensitive data. Finally, we will employ extracted attributes combined with administrative datasets to examine consistency and proportionality of imposed sentences using regression analysis.

Acknowledgments. This study was funded by the Czech Grant Foundation (grant number 25-16848M entitled "Just Sentences: Analyzing and Enhancing Proportionality and Consistency Using Typical Crimes"). The authors have no competing interests.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Adhikary, S., Sen, P., Roy, D., Ghosh, K.: A case study for automated attribute extraction from legal documents using large language models. Artificial Intelligence and Law pp. 1–22 (11 2024). https://doi.org/10.1007/s10506-024-09425-7
2. Aletras, N., Tsarapatsanis, D., Preoţiuc-Pietro, D., Lampos, V.: Predicting judicial decisions of the european court of human rights: a natural language processing perspective. PeerJ Computer Science **2**, e93 (2016). https://doi.org/10.7717/peerj-cs.93, https://doi.org/10.7717/peerj-cs.93
3. Bendová, K., Knap, T., Černý, J., Pour, V., Šavelka, J., Kvapilíková, I., Drápal, J.: What are the facts? automated extraction of court-established facts from criminal-court opinions. In: Proceedings of the ASAIL 2025 Workshop at ICAIL. Northwestern Pritzker School of Law, Chicago, IL, USA (2025), http://arxiv.org/abs/2511.05320
4. Chalkidis, I., Androutsopoulos, I., Aletras, N.: Neural legal judgment prediction in English. In: Korhonen, A., Traum, D., Màrquez, L. (eds.) Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 4317–4323. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-1424, https://aclanthology.org/P19-1424/
5. Chalkidis*, I., Garneau*, N., Goanta, C., Katz, D.M., Søgaard, A.: LeXFiles and LegalLAMA: Facilitating English Multinational Legal Language Model Development. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Toronto, Canada (2023), https://arxiv.org/abs/2305.07507
6. dAmato, C., Rubini, G., Didio, F., Francioso, D., Amara, F.Z., Fanizzi, N.: Automated creation of the legal knowledge graph addressing legislation on violence against women: Resource, methodology and lessons learned (2025), https://arxiv.org/abs/2508.06368
7. Gray, M.A., Savelka, J., Oliver, W.M., Ashley, K.D.: Can GPT alleviate the burden of annotation? In: Sileno, G., Spanakis, J., van Dijck, G. (eds.) Legal Knowledge and Information Systems - JURIX 2023: The Thirty-sixth Annual Conference, Maastricht, The Netherlands, 18-20 December 2023. Frontiers in Artificial Intelligence and Applications, vol. 379, pp. 157–166. IOS Press (2023). https://doi.org/10.3233/FAIA230961, https://doi.org/10.3233/FAIA230961
8. Gray, M.A., Savelka, J., Oliver, W.M., Ashley, K.D.: Using llms to discover legal factors. In: Savelka, J., Harasta, J., Novotná, T., Mísek, J. (eds.) Legal Knowledge and Information Systems - JURIX 2024: The Thirty-seventh Annual Conference, Brno, Czech Republic, 11-13 December 2024. Frontiers in Artificial Intelligence and Applications, vol. 395, pp. 60–71. IOS Press (2024). https://doi.org/10.3233/FAIA241234, https://doi.org/10.3233/FAIA241234

9. Katz, D.M., Bommarito, M.J., Gao, S., Arredondo, P.: Gpt-4 passes the bar exam. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **382**(2270) (apr 2024). https://doi.org/10.1098/rsta.2023.0254, https://doi.org/10.1098/rsta.2023.0254
10. Klie, J.C., Bugert, M., Boullosa, B., Eckart de Castilho, R., Gurevych, I.: The INCEpTION platform: Machine-assisted and knowledge-oriented interactive annotation. In: Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations. pp. 5–9. Santa Fe, New Mexico (2018), https://www.aclweb.org/anthology/C18-2002
11. Savelka, J., Ashley, K.D., Gray, M.A., Westermann, H., Xu, H.: Can gpt-4 support analysis of textual data in tasks requiring highly specialized domain expertise? In: Proceedings of the Sixth Workshop on Automated Semantic Analysis of Information in Legal Text (ASAIL 2023) (2023)
12. Sovrano, F., Palmirani, M., Vitali, F.: Legal Knowledge Extraction for Knowledge Graph Based Question-Answering (12 2020). https://doi.org/10.3233/FAIA200858
13. Wu, Y., Zhou, S., Liu, Y., Lu, W., Liu, X., Zhang, Y., Sun, C., Wu, F., Kuang, K.: Precedent-enhanced legal judgment prediction with LLM and domain-model collaboration. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 12060–12075. Association for Computational Linguistics, Singapore (Dec 2023). https://doi.org/10.18653/v1/2023.emnlp-main.740, https://aclanthology.org/2023.emnlp-main.740/
14. Zin, M.M., Nguyen, H., Satoh, K., Sugawara, S., Nishino, F.: Information extraction from lengthy legal contracts: Leveraging query-based summarization and GPT-3.5. In: Sileno, G., Spanakis, J., van Dijck, G. (eds.) Legal Knowledge and Information Systems - JURIX 2023: The Thirty-sixth Annual Conference, Maastricht, The Netherlands, 18-20 December 2023. Frontiers in Artificial Intelligence and Applications, vol. 379, pp. 177–186. IOS Press (2023). https://doi.org/10.3233/FAIA230963, https://doi.org/10.3233/FAIA230963

Can Legislation Be Made Machine-Readable in PROLEG?

An Investigation of GDPR Article 6

May Myo Zin[1][0000−0003−1315−7704], Sabine Wehnert[2][0000−0002−5290−0321], Yuntao Kong[1][0009−0001−2089−2363], Ha Thanh Nguyen[1,3][0000−0003−2794−7010], Wachara Fungwacharakorn[1][0000−0001−9294−3118], Jieying Xue[1][0009−0000−8070−6609], Michał Araszkiewicz[4][0000−0003−2524−3976], Randy Goebel[5][0000−0002−0739−2946], Ken Satoh[1][0000−0002−9309−4602], and Nguyen Le Minh[6][0000−0002−2265−1010]

[1] Center for Juris-Informatics, ROIS-DS, Tokyo, Japan
[2] Ruhr-University Bochum, RC-Trust, Bochum, Germany
[3] Research and Development Center for Large Language Models, NII, Tokyo, Japan
[4] Uniwersytet Jagielloński w Krakowie: Kraków, Poland
[5] University of Alberta: Edmonton, Alberta, Canada
[6] Japan Advanced Institute of Science and Technology, Ishikawa, Japan

Abstract. The anticipated positive social impact of regulatory processes requires both the accuracy and efficiency of their application. Modern artificial intelligence technologies, including natural language processing and machine-assisted reasoning, hold great promise for addressing this challenge. We present a framework to address the challenge of tools for regulatory application, based on current state-of-the-art (SOTA) methods for natural language processing (large language models or LLMs) and formalization of legal reasoning (the legal representation system PROLEG). As an example, we focus on Article 6 of the European General Data Protection Regulation (GDPR). In our framework, a single LLM prompt simultaneously transforms legal text into if-then rules and a corresponding PROLEG encoding, which are then validated and refined by legal domain experts. The final output is an executable PROLEG program that can produce human-readable explanations for instances of GDPR decisions. We describe processes to support the end-to-end transformation of a segment of a regulatory document (Article 6 from GDPR), including the prompting frame to guide an LLM to "compile" natural language text to if-then rules, then to further "compile" the vetted if-then rules to PROLEG. Finally, we produce an instance that shows the PROLEG execution. We conclude by summarizing the value of this approach and note observed limitations with suggestions to further develop such technologies for capturing and deploying regulatory frameworks.

Keywords: Machine-readable legislation · GDPR · PROLEG · Legal reasoning · Large language models · Human-in-the-loop workflow

1 Introduction

Modern legal and regulatory texts, such as the EU General Data Protection Regulation (GDPR), are written for human interpretation. Their open-textured language, cross-references, and exceptions make them difficult to operationalize in formal models, let alone in software [1]. As organizations seek to automate compliance at scale, this human-centric drafting becomes a bottleneck: it impedes consistent interpretation, slows audits, and raises the cost of demonstrating conformity. Converting law into a structured, machine-readable form is therefore a prerequisite for AI-assisted legal analysis, explainable decision support, and robust, auditable compliance automation.

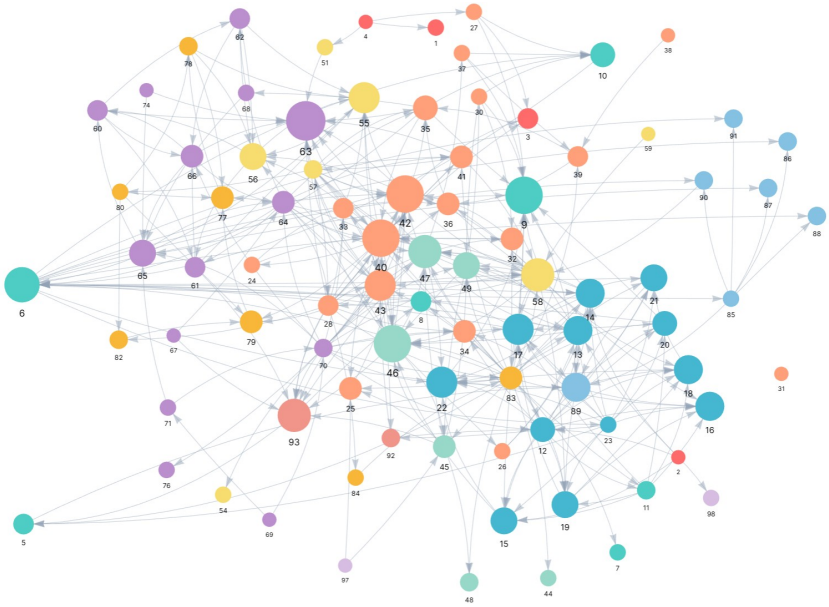


Fig. 1. Network of explicit cross-references among GDPR Articles.

This paper presents a human-in-the-loop workflow that transforms natural-language provisions into executable rules suitable for automatic reasoning and compliance checking. We focus on GDPR Article 6 (lawfulness of processing) as a representative, high-impact provision whose nuanced conditions and exceptions exemplify the challenges of formalization. Our approach combines large language models (LLMs) for scalable drafting with expert validation to ensure legal fidelity, and targets a logic-based formalism (PROLEG) to support sound, *inspectable* inference. As illustrated by Figure 1, the GDPR exhibits a dense network of inter-article references. This figure visualizes only explicit connections, yet this interconnectedness implies that analyzing Article 6 will also require consideration of related provisions.

The PROLEG knowledge representation language [14] was developed to facilitate interactions between lawyers and legal reasoning systems. While it does not address all challenges—such as limited expressiveness for certain legal concepts and the ambiguity of legal texts—it does provide a minimal yet sufficient language for reasoning, thus enabling lawyers to understand system behavior.

Our approach pairs a single, fixed composite prompt—designed to produce both if–then rules and an initial PROLEG encoding from the article text—with expert review that ensures doctrinal fidelity and preserves the semantic structure of the provision. By grounding LLM-generated rule candidates in expert validation and an executable formalism, the framework supports transparent reasoning and traceability from formal rules back to the authoritative text.

The contributions of this work are threefold:

- A practical workflow that couples LLM-based rule generation with expert review to produce PROLEG-executable legal rules;
- A qualitative evaluation of the doctrinal fidelity of if–then and PROLEG representations produced by a fixed composite prompt; and
- A curated rule set and test cases for GDPR Article 6, together with a demonstration of end-to-end reasoning behavior, including failure modes.

We conclude by examining the limitations of the initial composite prompt approach, including residual ambiguity and challenges associated with transferring the method to new legal domains, and by outlining how the pipeline may generalize to other GDPR provisions and regulatory frameworks.

2 Background

Here we present background on the PROLEG logical framework and collect related work on formal representations of legal knowledge, spanning logic- and rule-based systems, ontology and knowledge-graph approaches, and machine-readable standards for normative texts.

2.1 PROLEG

PROLEG (short for PROlog-based LEGal reasoning support system) [15] is a logic programming framework developed to model and support legal reasoning. It is based on the Prolog language but is specifically designed to represent legal rules, exceptions, and facts in a structured and executable form.

The system was originally proposed to formalize reasoning under the Presupposed Ultimate Fact Theory (JUF theory) in Japanese jurisprudence. Unlike standard logic programming, PROLEG introduces constructs that capture the nuances of legal argumentation, such as exceptions, burden of proof, and rule hierarchies, which are central to legal decision-making.

In PROLEG, laws are encoded as logical rules (similar to Horn clauses), while the facts of a case are represented separately. The reasoning process then derives conclusions by applying the rules to the facts, taking into account exceptions

and conflicting interpretations. This allows PROLEG to simulate the reasoning process of courts and legal practitioners in a transparent and explainable way.

PROLEG has been used primarily in research on computational legal reasoning and AI & Law, particularly for analyzing statutory interpretation and case reasoning in civil law systems. It provides a visual reasoning trace, ensuring that the resulting conclusions are both interpretable and reproducible, thereby enhancing the transparency of the reasoning process. Although it is not a general-purpose programming language, it serves as an important bridge between symbolic AI and legal knowledge representation, contributing to the broader field of legal informatics. For illustration purposes, we consider the following use case for formalization.

A financial institution collected an extensive set of personal data from individuals on the basis of their consent. The institution processed the data, inter alia, for the purpose of marketing. Later, one of these individuals withdrew the consent and asked the institution to stop using the data. Despite this withdrawal, the institution intended to continue processing the data.

This use case has been transformed into PROLEG and concerns GDPR Article 6. The output result is shown in the form of a block diagram in Figure 2. A bottom item of each block expresses the result of evaluation of conclusions or conditions (o: success, x: fail). A solid arrow between blocks expresses the conclusion-condition relation for a general rule, while a dotted arrow shows the exception relation of the conclusion of a general rule. The reasoning concludes with an overall failure mark (x), indicating that continued processing after consent withdrawal is unlawful.

2.2 Logic-Based and Rule-Based Approaches

Early efforts focused on using logic-based systems to model legal rules and reasoning. Logic programming, particularly Prolog, has been used as a foundation for these systems. A prominent example is the PROLEG system (PROlog-based LEGal reasoning support system), developed to implement the Presupposed Ultimate Fact Theory of Japanese civil law [15]. This system demonstrates how legal rules can be represented as Horn clauses and then processed using Prolog technology to support legal reasoning, particularly in contexts with incomplete information.

The field has also explored non-monotonic logics such as defeasible logic and deontic logic to capture further nuances of legal reasoning, such as exceptions to rules and concepts of obligation, permission, and prohibition. The pioneering work of Jones et al. [5] explored the use of deontic logic to represent legal norms. Hybrid approaches have been proposed, such as that of Dragoni et al. [4], which combines different Natural Language Processing (NLP) techniques, including frame-based, dependency-based, and logic-based extraction, to extract machine-readable rules from legal texts. This hybrid approach demonstrates the necessity of integrating multiple methods to address the complexity of legal language. Similarly, work on normative texts within the Norme in Rete (NIR) project introduces machine-learning-based provision classifiers and rule-based

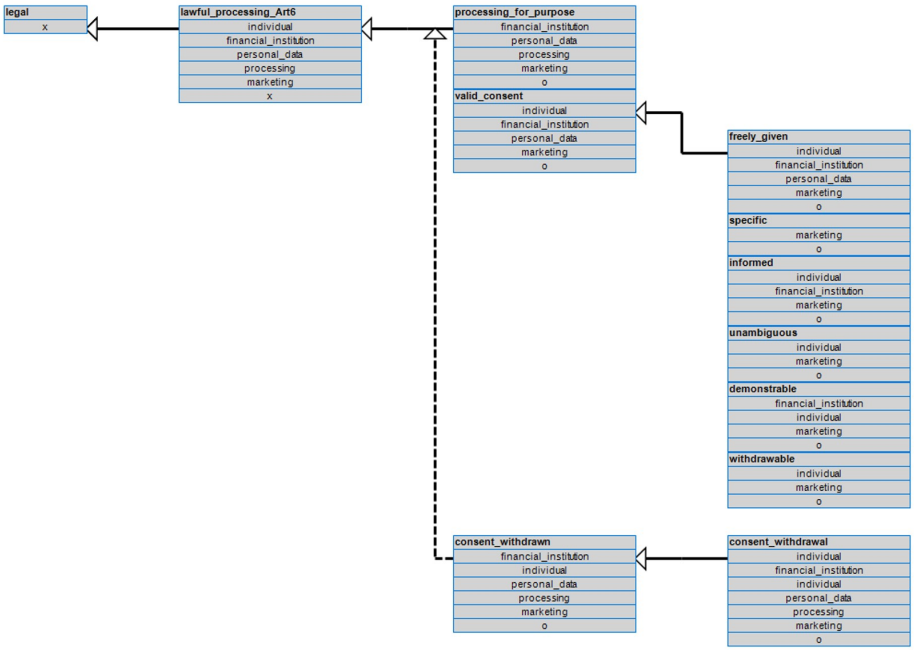


Fig. 2. Example of PROLEG block diagram.

argument extractors that operate at the level of individual provisions, automatically identifying provision types and their arguments to enrich legal documents with structured semantic metadata [2]. In our work, we likewise focus on the provision level, but our intention is to directly work on the provision text itself as the primary input to the reasoning pipeline, rather than assuming a separate, pre-annotated semantic layer.

Building upon PROLEG, Nguyen et al. [10] developed an interactive natural language interface to make the system more accessible to legal practitioners unfamiliar with PROLEG. The system consists of three main modules: a natural language perceiver, a PROLEG reasoner, and an inference explainer, thereby bridging the gap between formal logic representations and natural language input from lawyers. Subsequently, Nguyen et al. [11] proposed a pipeline for a Deep PROLEG system, which addressed scalability issues when synthesizing artificial data of legal cases in new domain adaptations.

2.3 Ontology-Based and Knowledge Graph Approaches

Another important research direction involves using ontologies to formally represent legal knowledge. Ontologies provide a shareable vocabulary and structure to describe concepts and relationships in the legal domain. Legal ontology engineering methodologies, as detailed by Casellas et al. [3], provide systematic

approaches to building and maintaining these knowledge representations. Relevant in our context is the work by Palmirani et al. [13] who introduce PrOnto, a modular GDPR-oriented privacy ontology developed with the MeLOn methodology that formally models data, actors, processing workflows, purposes, legal bases, and deontic norms to support automated legal reasoning and compliance checking using semantic web technologies. Leone et al. [7] systematically compare and classify a set of existing legal ontologies across general, modeling, and semantic dimensions. They highlight strengths, weaknesses, and reuse potential in order to guide users in selecting and extending suitable ontologies for legal knowledge representation. More recently, knowledge graphs have emerged as an extension of ontology-based methods, which are claimed to enable more flexible and scalable representation of legal entities and relationships. Servantez et al. [16] further advance this line of work by introducing a graph-based representation of contract obligations (Obligation Logic Graphs) that supports automated conversion of natural-language contracts into code; this further bridges the gap between legal ontologies and executable contract logic. In contrast, our work focuses on natural-language legal texts rather than formal ontologies, to leverage their linguistic richness and contextual depth to better capture meaning and reasoning in real-world legal documents.

2.4 Machine-Readable Standards

To ensure interoperability and widespread adoption, standardization of machine-readable formats for legal texts is important. As this standardization develops, several approaches have emerged. For example, **Akoma Ntoso** (Architecture for Knowledge-Oriented Management of African Normative Texts using Open Standards and Ontologies) is a prominent XML standard for representing parliamentary, legislative, and judicial documents [12]. It provides a rich vocabulary for marking up the structure and semantics of legal texts.

Building on this foundation, **LegalRuleML** has been developed as another standard for representing legal rules in a machine-readable format. Lam et al. [6] demonstrated how LegalRuleML can be used to enable reasoning by transforming represented rules into modal defeasible logic, thereby bridging the gap between rule representation and automated reasoning. More recently, new semantic formats such as **X2RL** (eXplainable, eXtractable, Rule-like Language) have been proposed by McLaughlin et al. [9] to augment regulatory documents with rich metadata fields, and focus not only on structure but also on content and meaning, which aims to reduce regulatory management and compliance costs.

Logic-based systems make norms executable but need manual rule crafting and expert use; ontology and graph methods model structure but not executable reasoning; and machine-readable standards ensure interoperability yet stop before inference. Our work bridges these by turning natural-language provisions directly into validated, PROLEG-executable rules through an LLM-expert workflow. This yields traceable, auditable, and explainable reasoning while remaining lightweight and compatible with existing ontologies and standards.

3 Methodology

In this section, we describe our methodology and detail the prompt design.

3.1 Overview of the Approach

Our goal is to examine how a single composite LLM prompt can support the transformation of legal text into PROLEG-executable rules. To ensure a controlled evaluation, the study uses one fixed prompt, applied once to each target provision. The outputs of this prompt, namely a set of if–then rules and an initial PROLEG encoding, form the basis for expert analysis.

Figure 3 presents the conceptual workflow, which consists of four stages involving both automated generation and expert review. Although the figure separates these stages for clarity, in this study both the if–then rules and the initial PROLEG encoding are produced by a single composite prompt.

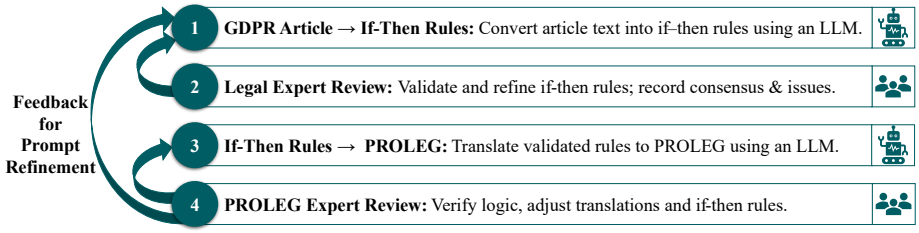


Fig. 3. Overview of the GDPR to PROLEG conversion process.

The workflow proceeds as follows:

1. **Generation of If–Then Rules and initial PROLEG Encoding:** The composite prompt instructs the LLM to interpret the selected GDPR provision, extract its conditions and exceptions, express them as structured if–then rules, and generate a corresponding PROLEG-style representation. These two representations serve as preliminary formalizations of the provision.
2. **Legal Expert Review:** Legal experts examine the if–then rules to assess whether the extracted structure accurately reflects the provision's normative content. Their review focuses on the allocation of information across main rules and sub-rules, the handling of exceptions, and the preservation of legally significant distinctions.
3. **PROLEG Expert Review:** Specialists in PROLEG evaluate the LLM-generated formalization for syntactic correctness and intended semantics. They identify issues such as predicate mismatches, logical gaps, or distortions introduced by the model's interpretation of the text.

4. **Case Creation and Execution:** A case-creation team constructs representative factual scenarios, defines the necessary fact schema, and encodes these facts. The validated rule set and encoded facts are then executed in a PROLEG environment, producing reasoning traces that reveal how the formalized rules behave in concrete cases.

This design allows us to isolate and analyze the characteristic patterns of a fixed-prompt approach, both its strengths and its limitations, while keeping LLM generation, expert evaluation, and PROLEG execution clearly separated. We systematically record recurring issues in the structure or wording of the rules and in their PROLOG counterparts as part of our evaluation.

3.2 Prompt Design

The interpretation of the GDPR is inherently complex due to the interdependencies among its provisions, which frequently span multiple Articles and their corresponding Recitals; therefore, a comprehensive and systematic interpretative approach must consider these cross-references and contextual relationships. This study employs a custom version of ChatGPT-5 in which the Chain-of-Instructions (COI) prompt [17] is embedded as an internal reasoning framework; this guides the model through each stage of legal interpretation and rule formalization. The prompt operates through five internal instruction stages. The first two stages instruct the model to identify all relevant Recitals and cross-referenced Articles that provide interpretive or procedural context for the selected GDPR provision. This ensures that the subsequent interpretation and formalization processes are contextually grounded and legally coherent. The third stage directs the synthesis of these interpretive materials into a unified, detailed legal rule that explicitly incorporates all dependencies and interpretive nuances. This stage is critical for maintaining normative fidelity, as omissions or simplifications could compromise the representational validity of the machine-readable rule. In the fourth stage, the synthesized rule is transformed into a normalized if-then structure. This logical framing not only enforces syntactic clarity but also aligns the rule with the representational requirements of logic-based systems. It explicitly separates conclusions, conditions, and exceptions, thereby supporting transparency in both human and machine reasoning. The final stage involves generating a complete PROLOG program that includes entities, rules, exceptions, facts, and queries. Although our ultimate goal is to produce a PROLEG program, we initially instruct the model to generate the logic in PROLOG, since current LLMs are more familiar with it. This approach helps achieve higher accuracy, and we can then systematically convert the resulting PROLOG program into PROLEG. The complete prompt and the corresponding initial outputs generated for GDPR Article 6 (Lawfulness of Processing) are available in the GitHub repository[7].

[7] `https://github.com/JurisInformaticsCenter/GDPR-PROLEG-data`

4 Results and Analysis

The generated if-then rules were subject to qualitative evaluation performed by legal experts. This evaluation was based on the criterion of the adequacy of the generated rules to the structure and content of legal norms that lawyers derive from the GDPR provisions expressed in natural language. Importantly, our approach relied on the structural resemblance assumption, meaning that a representation of a rule based on one Article (or a part thereof) is preferred over complex representations derived from multiple Articles. If other parts of the normative material are relevant, e.g., for the interpretation of the generated rule, they should be presented in the form of accompanying sub-rules rather than as components of the general rule. This semantic layering of the generated rules was verified against professional doctrinal material [8]. Different types of infidelity may arise between the if-then rule and the original normative material, including confused or otherwise altered structure, as well as semantic modifications that may lead either to restriction or broadening of the target rule. One research question of the project is to investigate what types of inadequacies or other issues occur in the generated content, assuming the fixed prompting system defined above. We did not attempt to develop a complete classification of possible inadequacy issues. At the outset, we adopted only a general distinction between structural and semantic issues. We decided to develop a typology of such issues using a bottom-up method, that is, by analysing specific outputs and clarifying the nature of the identified issues. The resulting catalogue of issue examples is presented in the following two subsections: structural issues (Section 4.1) and semantic issues (Section 4.2).

4.1 Structural Issues

Questionable allocation of information to sub-rules: ChatGPT generated catalogues of sub-rules for the main rules, but in certain instances, it was unclear why one piece of information was included in the set of sub-rules while another, equally relevant, was not. For example, in the context of Art. 6.1(a) (consent), ChatGPT generated sub-rules explaining that consent is freely given (Art. 7.4) and that consent should be withdrawable (Art. 7.3), but then omitted the requirement that consent should be distinguishable from other statements and expressed in clear language (Art. 7.2). Similarly, it created exception-like sub-rules indicating parental authorization of a child's consent (Art. 8) and explicit consent for the processing of special categories of data (Art. 9.1(a)), but not other instances of explicit consent (Art. 22(c), Art. 49.1(a)). In another result, ChatGPT generated a sub-rule related to Arts. 13 and 14 (in representing Art. 6.1(c)), while (correctly) refraining from doing so in other cases.

Questionable allocation of information to main rules: In some instances, ChatGPT enriched the catalogue of conditions for the main rule extracted from a specific Article by resorting to other Articles. For example, in the output for Art.

6.1(a), it included the demonstrability requirement (Art. 7.1) in the body of the rule, as opposed to the withdrawability requirement (Art. 7.3), which was placed in the sub-rules. Similarly, in the generated main rule representing Art. 6.1(b) (contractual necessity), ChatGPT added a condition that processing should be compliant with the general principles outlined in Art. 5. However, this condition applies to any instance of processing under the GDPR and should therefore be included in the condition set of every rule concerning data processing. This was not done consistently. Our conclusion was that such results should be avoided for the sake of structural resemblance.

Generating presupposed clauses: In some instances concerning the reconstruction of rules regarding data processing, ChatGPT reconstructed presupposed information ("data is processed"), as in the case of Art. 6.1(c), whereas in other instances it did not. Generally, this should be avoided because, if a rule specifies the conditions for the legality of data processing, its applicability already presupposes that data processing is occurring; thus, the information "data is processed" is already implicit in any condition that mentions processing.

4.2 Semantic Issues

Predicate simplification: The generated sub-rule explaining the term "being freely given" simplified the condition excluding compliance with this requirement. The generated rule reads: "If consent is conditional upon services not necessary for contract performance, then it is not freely given." However, such conditionality is not, in fact, a sufficient reason to conclude that consent is not freely given. In some cases, despite the presence of a conditional mechanism, consent may still be freely given. This is because the existence of such a mechanism is gradual rather than binary: the more intensively it is present in the analysed case, the stronger, ceteris paribus, the argument that consent might not have been freely given. Additionally, the presence of a conditional mechanism is only one among many reasons that may lead to non-compliance with the requirement: ChatGPT omitted the "inter alia" clause present in Art. 7.4.

Questionable paraphrases or inference results as sub-rules: In some instances, instead of providing explanatory or definitional information in sub-rules, ChatGPT attempted to paraphrase the main rule or to draw an inference from it. This occurred in the context of Art. 6.1(b). The natural-language expression of the rule reads: "processing is necessary for the performance of a contract to which the data subject is party or in order to take steps at the request of the data subject prior to entering into a contract," whereas the generated sub-rule stated: "If the purpose of processing is not necessary for performing or entering a contract, then processing is not lawful under Article 6(1)(b)." Although this is, strictly speaking, a valid inference, it omits important information: processing of data prior to entering the contract must occur at the request of the data subject, not the controller. Consequently, if we intend to draw an inference from the

main rule to a negative conclusion (i.e., specify when there is non-compliance with the Article), we should also state that even if such processing were necessary prior to entering the contract, it would still be non-compliant if initiated by the controller.

Adding information not present in the relevant source text: In subrule 3 generated for Art. 6.1(c), ChatGPT stated: "If the legal obligation is derived from non-EU/non-Member State law without an EU legal mandate." The phrase "without an EU legal mandate" does not appear in the source text. Art. 6.3, which clarifies the sources of relevant legal obligations, mentions only Union law or Member State law. Moreover, the term "mandate" is used in the GDPR in different contexts, referring to authorizing an entity to act on behalf of someone.

Restricting the semantic scope of expressions: While generating a rule representing Art. 6.1(d), ChatGPT restricted the crucial expression "vital interest" to "vital interest essential for the life or physical integrity." Although the resulting scope includes instructive examples, it omits a significant part of the original scope; for example, private property may also be classified as a "vital interest."

5 Methodological Limitations

Although our goal was to explore an end-to-end transformation pipeline for converting natural-language legal provisions into executable PROLEG rules, we deliberately limited the study to a single custom LLM configuration and to GDPR Article 6. This constrained scope enabled us to isolate the effects of prompt design, model behavior, and expert feedback without introducing additional variability from cross-model differences or broader regulatory contexts. Accordingly, the present analysis is best interpreted as a proof-of-concept rather than a comprehensive empirical evaluation. Future work will extend the methodology to multiple LLMs, additional GDPR provisions, and larger, more heterogeneous test sets to systematically assess robustness and generalizability.

Within this constrained setting, the methodology demonstrates the feasibility of linking legal interpretation and logical formalization within a single prompt, but several limitations must be acknowledged. The current approach remains sensitive to model behavior, even when using the same prompt and identical input text. When applied repeatedly to the same Article, the number and ordering of extracted Recitals and cross-referenced Articles occasionally differ slightly across runs. Although the LLM consistently identifies the most important and directly relevant provisions, it occasionally omits provisions that are not explicitly connected but remain contextually essential for a complete legal interpretation. For example, when applied to Article 6(1)(a) (lawfulness of processing based on consent), the model successfully retrieved the directly related and most relevant

provisions, such as Articles 4(11), 7, 8, and 9(2)(a), as well as key Recitals 32, 33, 42, and 43. However, it failed to include two important Articles, namely Article 49(1)(a) and (f), and Article 22(1) and (2), which provide necessary contextual and substantive links regarding data transfers based on explicit consent and the prohibition of automated individual decision-making, including profiling. These omissions suggest that, while the model performs effectively in identifying core definitional and consent-related provisions, it struggles to capture cross-contextual dependencies that are less syntactically but more semantically connected, thereby highlighting a limitation in its ability to model the broader legal relationships inherent in the GDPR framework.

Moreover, the methodology was intentionally confined to a single prompt with an internal Chain-of-Instructions (CoI) executed by a single-model configuration. Although this design enabled an assessment of the model's capacity for integrated, end-to-end legal text formalization, future work could explore multi-agent or modular architectures in which specialized sub-models independently address interpretive, logical, and representational tasks. Such distributed systems could enhance interpretability, mitigate reasoning drift, and support feedback loops between human experts and computational agents.

6 Discussion

The results of our qualitative evaluation demonstrate both the promise and the current limitations of using LLMs for the structured reconstruction of legal norms. On the one hand, the models are capable of producing rule-like representations that reflect, at least superficially, the architecture of statutory provisions. On the other hand, our analysis reveals systematic deviations from the expected structural and semantic fidelity. These deviations arise not merely at the level of technical inaccuracies but often concern deeper issues related to the allocation of information, the preservation of essential normative distinctions, and the interpretation of legal terminology. The structural issues show that the model lacks a stable internal criterion for distinguishing between the general rule and its subrules, leading to an inconsistent distribution of content across representational layers. Similarly, the semantic issues indicate a tendency toward overgeneralisation, oversimplification, or unwarranted inferences, which can narrow or distort the scope of the reconstructed norm. Taken together, these findings suggest that while current prompting methods can elicit rules resembling expert-generated representations, they do not guarantee their doctrinal adequacy. The observed patterns of inadequacy underscore the need for comparing different prompting strategies, model calibration, and perhaps hybrid systems that combine linguistic generation with domain-specific legal constraints.

7 Conclusion and Future Work

We have examined whether GDPR Article 6 can be rendered machine-readable in PROLEG through a human-in-the-loop workflow that combines large language

models with expert review. We showed how a single chain of instructions prompt can guide the model from identifying relevant Articles and Recitals, through drafting if-then rules, to compiling executable PROLEG code and test cases in a demo system for compliance checks. Our qualitative evaluation indicates that this LLM-based compilation process can recover much of the structure of legal norms, while also revealing recurring structural and semantic issues, such as unstable allocation of content across main rules and sub-rules, reconstruction of presupposed clauses, and simplifications or additions that alter the scope of key predicates. We deliberately assumed this prompting basis as a static approach and observed what followed from that choice, but future work will improve the prompts, for example, by explicitly instructing the model not to paraphrase the wording of GDPR Articles and Recitals, because such paraphrasing introduced additional ambiguity. These findings confirm the value of PROLEG as an inspectable target formalism, highlighting the suitability of LLMs as generators of rule candidates. It further demonstrates that expert validation is essential for doctrinally reliable machine-readable legislation. Future work will refine prompt design in light of these insights, strengthen guidance for LLM-based rule and code generation, and extend the general approach beyond Article 6 to other parts of the GDPR and to further bodies of legislation.

Acknowledgments. This research was supported by ROIS-DS-JOINT (023RP2025), the "Strategic Research Projects" grant from ROIS (Research Organization of Information and Systems), the "R&D Hub Aimed at Ensuring Transparency and Reliability of Generative AI Models" project of the Ministry of Education, Culture, Sports, Science and Technology, and JSPS KAKENHI Grant Numbers JP22H00543, JP25H00522, JP25H01112, and JP25H01152.

References

1. Araszkiewicz, M., Płeszka, K. (eds.): Logic in the Theory and Practice of Lawmaking. Springer (2015)
2. Biagioli, C., Francesconi, E., Passerini, A., Montemagni, S., Soria, C.: Automatic semantics extraction in law documents. In: Proceedings of the 10th international conference on Artificial intelligence and law. pp. 133–140 (2005)
3. Casellas, N.: Legal ontology engineering: Methodologies, modelling trends, and the ontology of professional judicial knowledge, vol. 3. Springer Science & Business Media (2011)
4. Dragoni, M., Villata, S., Rizzi, W., Governatori, G.: Combining nlp approaches for rule extraction from legal documents. In: 1st Workshop on MIning and REasoning with Legal texts (MIREL 2016) (2016)
5. Jones, A.J., Sergot, M.: Deontic logic in the representation of law: Towards a methodology. Artificial Intelligence and Law **1**(1), 45–64 (1992)
6. Lam, H.P., Hashmi, M.: Enabling reasoning with legalruleml. Theory and Practice of Logic Programming **19**(1), 1–26 (2019)

7. Leone, V., Di Caro, L., Villata, S.: Taking stock of legal ontologies: a feature-based comparative analysis. Artificial Intelligence and Law **28**(2), 207–235 (2020)
8. Litwiński, P. (ed.): ODO. Compliance. Praktyczny komentarz z przykładami i orzecznictwem. (Data Protection. Compliance. Practical commentary with examples and case law). C.H. Beck (2025)
9. McLaughlin, P.A., Stover, W.: Drafting x2rl: A semantic regulatory machine-readable format. MIT Computational Law Report (2021)
10. Nguyen, H.T., Nishino, F., Fujita, M., Satoh, K.: An interactive natural language interface for proleg. In: Legal Knowledge and Information Systems, pp. 294–297. IOS Press (2022)
11. Nguyen, P.M., Nguyen, T.H., Zin, M., Satoh, K.: Data augmented pipeline for legal information extraction and reasoning. In: Proceedings of the 20th International Conference on Artificial Intelligence and Law (ICAIL 2025). p. 2. ACM, Chicago, IL, USA (June 16–20 2025). https://doi.org/10.1145/3769126.3769200
12. Palmirani, M.: Legislative change management with akoma-ntoso. In: Legislative XML for the Semantic Web: Principles, Models, Standards for Document Management, pp. 101–130. Springer (2011)
13. Palmirani, M., Martoni, M., Rossi, A., Bartolini, C., Robaldo, L.: Pronto: Privacy ontology for legal reasoning. In: International Conference on Electronic Government and the Information Systems Perspective. pp. 139–152. Springer (2018)
14. Satoh, K.: PROLEG: Practical Legal Reasoning System, pp. 277–283. Springer Nature Switzerland, Cham (2023), https://doi.org/10.1007/978-3-031-35254-6_23
15. Satoh, K., Asai, K., Kogawa, T., Kubota, M., Nakamura, M., Nishigai, Y., Shirakawa, K., Takano, C.: Proleg: an implementation of the presupposed ultimate fact theory of japanese civil code by prolog technology. In: JSAI international symposium on artificial intelligence. pp. 153–164. Springer (2010)
16. Servantez, S., Lipka, N., Siu, A., Aggarwal, M., Krishnamurthy, B., Garimella, A., Hammond, K., Jain, R.: Computable contracts by extracting obligation logic graphs. In: Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law. pp. 267–276 (2023)
17. Zin, M.M., Satoh, K., Borges, G.: Leveraging llm for identification and extraction of normative statements. In: Legal Knowledge and Information Systems, pp. 215–225. IOS Press (2024)

Using LLMs to Create Legal Ontologies for Traffic Rule Compliance

Galileo Sartor[1][0000−0001−6355−851X], Thiago Raulino Dal Pont[2][0000−0001−7837−5505], Enrico Francesconi[3][0000−0001−8397−5820], and Adam Wyner[1][0000−0002−2958−3428]

[1] Swansea University, Department of Computer Science, Swansea, UK
sartor,wyner@swansea.ac.uk
[2] University of Bologna, Alma-AI Centre, Bologna, Italy thiago.raulino@unibo.it
[3] IGSG-CNR, Institute of Legal Informatics and Judicial Systems, National Research Council, Italy enrico.francesconi@cnr.it

Abstract. This paper introduces a pipeline leveraging Large Language Models (LLMs) to automate the creation and maintenance of rule-based legal ontologies. We also explore the use of the generated legal ontologies to perform legal reasoning in autonomous agents, focusing on traffic rules for Autonomous Vehicles (AVs). The proposed ontology model encodes compliant and non-compliant instances through property restrictions. We present a prototype of the pipeline and its use, discussing the different components and how they can be adapted.
Additionally, we discuss the possible use of the system to keep legal ontologies up to date with changes in legislation and case law.

Keywords: Legal reasoning · Semantic Web · OWL · Norm compliance · Large Language Models.

1 Introduction

Ontologies have become a foundational tool in the legal domain, enabling structured representation, management, and retrieval of complex legal knowledge. Early work emphasized the need for explicit domain conceptualizations to support legal information systems, highlighting ontologies as a means to clarify commonalities and differences across legal approaches and to facilitate the creation of reusable legal knowledge libraries [2].

However, the development and maintenance of legal ontologies is a complex and resource-intensive task, often requiring significant expertise in both law and ontology engineering. This has led to a knowledge acquisition bottleneck, limiting the scalability and adaptability of legal ontologies.

To address this bottleneck, we use Large Language Models (LLMs).These models can assist in creating and updating legal ontologies, thereby improving the efficiency of ontology development. This research applies this approach to create an ontology that enables legal reasoning and compliance evaluation in the

context of Autonomous Vehicles (AVs). This paper explores how LLMs can facilitate knowledge modelling by partially automating the definition and population of ontology models from existing natural language sources.

2 Autonomous Driving

Currently, most AV implementations rely predominantly on machine learning (ML) models to govern vehicle behaviour and ensure legal compliance. However, these systems lack an explicit knowledge representation of legal rules, meaning they do not possess a structured understanding of traffic laws, nor a way to evaluate their compliance in accordance to the existing traffic norms. This lack of explicit specification is a central challenge, as decision-making algorithms must also provide legally defensible explanations for their actions. Instead, compliance is mostly derived through data-driven approaches, where the predicted behaviour is evaluated in a testing phase. Lawful behaviour can be induced by tweaking the loss function, but due to the *black-box nature of neural networks*, there is no guarantee that the actual behaviour is compliant with the relevant legal norms [24, 15].

Conversely, most research in decision-making for AVs often considers only collision free driving to be compliant, removing the complexity of the environment entirely and constructing controlled models that consider traffic laws [14]. While the use of ML models is effective and an important piece of the driving puzzle, an approach solely based on inferred behaviour lacks explainability and robustness in complex factual, legal, and ethical scenarios. This is potentially even more relevant if we consider highly automated vehicles (SAE Levels 4 and 5, with minimal to no human interaction required [7]) driving in environments where the available data for training is lacking.

Another consideration is the range of legal jurisdictions which may be in parallel (e.g., crossing national boundaries) or hierarchical (e.g., the Vienna Convention on Road Traffic in conjunction with national and local regulations). Here too, ontologies could help in developing a tiered knowledge base, where the different legal (and societal) norms can be merged together.

3 Related Work

To deal with the above-mentioned issue of legal compliance a number of different approaches are being evaluated, broadly grouped in two categories: scenario testing, which deals with specific complex situations, and more general rule representation. Of the two, we are more interested in the second, with the goal of developing a comprehensive rulebase. The different proposed systems leverage different logics or programming languages, such as Prolog [6], Linear Temporal Logic [19], or Defeasible Deontic Logic [3].

The related work can be divided into two main areas: a) the use of ontologies in the considered legal domains, and b) the use of LLMs to create and update

ontologies. This work will not go into the use of LLMs directly for legal reasoning, as it is out of scope for this paper. Still, it is worth mentioning that the increase in the capability of LLMs is a critical piece in the proposed pipeline, and understanding the usefulness and the limitations of LLMs for legal reasoning is an important area of research.

Ontologies have a long history of application in the legal domain, serving various purposes such as *knowledge acquisition, sharing, reuse, verification and validation, and domain theory development* [2].

Subsequent research introduced core legal ontologies such as FOLaw and LRI-Core, which model dependencies in legal reasoning and capture abstract, commonsense concepts underlying legal knowledge. These ontologies have been applied in various European ICT projects, demonstrating practical value in legal information processing and reasoning tasks [22].

The evolution of legal ontologies has paralleled advances in the Semantic Web, with a shift toward scalable, reusable, and interoperable models. Modern legal ontologies are often built using W3C standards (e.g., OWL), and their development is guided by principles of knowledge reuse and modularity [13]. Comparative analyses have catalogued a wide range of legal ontologies, focusing on their features, implementation details, and suitability for reuse across different legal frameworks.

Ontologies have also been leveraged to improve legal information retrieval, addressing challenges such as synonymy and ambiguity in legal texts. Ontology-based search frameworks and knowledge graphs have demonstrated superior performance over traditional keyword-based methods, enabling more accurate and semantically rich retrieval of legal documents [1]. Additionally, ontologies support advanced applications such as legal argumentation, statutory reasoning, and the semantic annotation of legal texts, further enhancing the capabilities of legal information systems.

Ontologies can be used to model deontic norms as classes and restrictions [9], an approach that can handle defeasible norms and leverage the OWL reasoners to make decisions and perform legal compliance checking.

Recent work has explored the integration of foundational ontologies (e.g., UFO) to foster interoperability and conceptual rigour, as well as the use of ontology design patterns to streamline the modelling of recurring legal knowledge [10]. The field continues to address challenges related to the dynamic and heterogeneous nature of legal concepts, the need for domain expert involvement, and the adaptation of ontologies to evolving legal systems and technologies [8].

In the Autonomous Vehicle (AV) domain, ontologies have been employed to model traffic regulations, road environments, and vehicle behaviours, as well as to define scenarios and situations. These ontologies facilitate the interpretation and application of traffic rules by AV systems, supporting decision-making processes in complex driving scenarios [17, 18, 21]. For instance, ontologies have been used to represent traffic signs, signals, and road layouts, enabling AVs to understand and comply with legal requirements while navigating diverse road conditions [11, 12].

With the increasing ability of Large Language Models to manipulate text and to assist with coding tasks, it becomes possible to use these tools to generate a structured version of a source in natural language. Recent work has explored the use of LLMs to assist in ontology engineering tasks, such as ontology learning, population, and alignment. For example, LLMs have been employed to extract concepts and relationships from unstructured text, facilitating the semi-automated construction of ontologies [5]. Additionally, LLMs have been used to generate ontology axioms and definitions, leveraging their language understanding capabilities to produce coherent and contextually relevant representations.

While the focus has often been on ontology population on the basis of an existing schema, there is also work on the use of LLMs to create an ontology from scratch, or to extend an existing ontology with new concepts and relationships [4]. This involves prompting the LLM with relevant domain knowledge and examples, and iteratively refining the generated ontology through human-in-the-loop validation and correction.

4 Ontology model and norm representation

The ontology model used in this work is based on a rule-based representation of legal norms, capturing the essential components of legal rules such as subjects, actions, conditions, and modalities such as in [23]. The ontology is designed to be expressive enough to represent complex legal constructs while remaining accessible for automated reasoning and compliance checking.

Norms, expressing obligations, permissions, and prohibitions, are represented as restrictions on the properties of legal agents (e.g., vehicles, drivers) within specific contexts (e.g., road environments). Each agent is modelled as a class in the ontology, with properties corresponding to actions they can perform and conditions under which these actions are regulated. Other entities in the environment (e.g., traffic signs, road types) are also modelled as classes with relevant properties. Determining the compliance status then consists on the evaluation of these properties against the defined norms.

Let us consider, as an example, Rule 171 from the UK Highway Code section "Using the Road".

> ***Rule 171***
> *You **MUST** stop behind the line at a junction with a 'Stop' sign and a solid white line across the road. Wait for a safe gap in the traffic before you move off.*
> *[Laws RTA 1988 sect 36 & TSRGD schedule 9 parts 7 and 8]*

In Figure 1 we show a possible representation of Rule 171 in the ontology, where the "Vehicle" class has subclass "VehicleR171", the subclass of vehicles pertaining R171, namely vehicles "at a junction with a 'Stop' sign and a solid white line across the road.

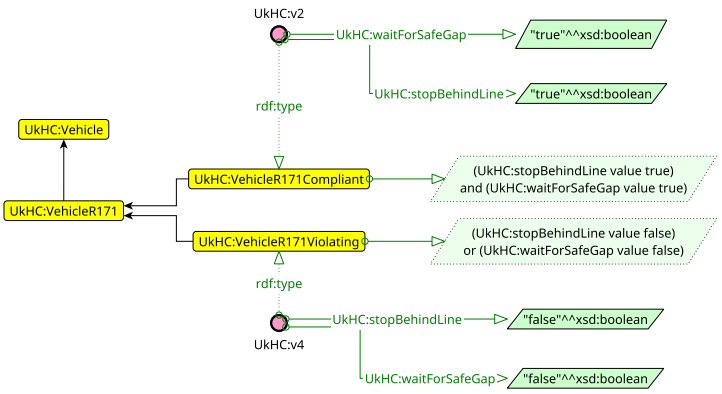


Fig. 1. Ontology representation of Rule 171 from the UK Highway Code.

Two subclasses represent the violating and compliant entities directly. The norm modelled is represented as a restriction on the properties of the "VehicleR171" class, specifying that vehicles of this class must stop at junctions with specific characteristics (i.e., a 'Stop' sign and a solid white line).

The "Violating" and "Compliant" classes are two disjoint subclasses of the *situation specific* class "VehicleR171", and differ in the values assigned to the properties of the class, stopBehindLine and waitForSafeGap. A vehicle is considered to be compliant if it both stops behind the line AND waits for a safe gap to restart. If one of the two properties is false, the vehicle instead violates the rule.

In the same graph, we have two individuals represented by pink dots, v2 and v4, that belong respectively to the Compliant and Violating classes. Vehicle v2 is compliant with rule 171 as both properties stopBehindLine and waitForSafeGap are true. Conversely, v4 fails to stop behind the line, thus violating the rule.

The represented ontology is a description of the norm, which can be extended with additional information, such as the vehicle type (e.g., an emergency vehicle has different requirements and obligations) or more complex scenarios.

5 Methodology

The proposed methodology aims to create a structured legal ontology from natural language legal texts using Large Language Models (LLMs). The process involves several steps to ensure that the generated ontology is consistent with the source text and the desired output is suitable for automated reasoning. It is useful to break down the task into smaller, manageable components, guiding the LLM through a structured prompt that focuses on specific aspects of the ontology creation process. Multiple LLMs are evaluated without changing the default parameters (e.g., temperature, penalty) and relying solely on the prompt. This fairly evaluates the portability of the instructions.

5.1 Proposed Pipeline

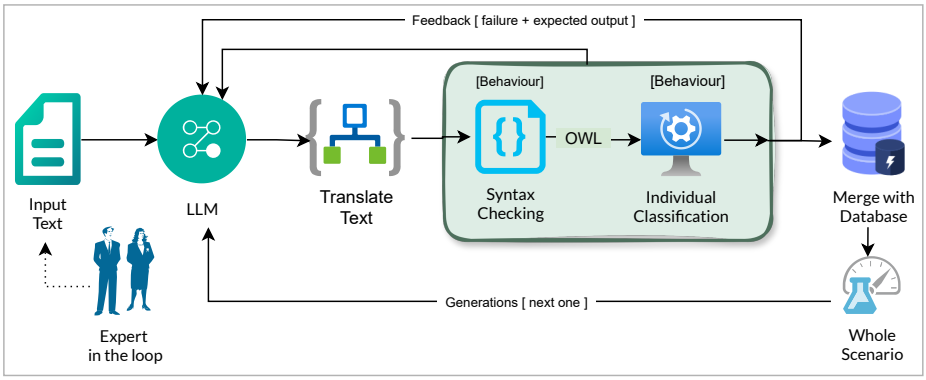


Fig. 2. Proposed pipeline for the creation of legal ontologies using LLMs.

The proposed translation pipeline is shown in Figure 2 and leverages the strengths of LLMs while attempting to constrain them from issues such as "hallucinations". The process is iterative, with human oversight and validation to ensure the accuracy and relevance of the generated ontology.

The input text is converted to the desired output by the LLM. The resulting ontology of this conversion is then initially loaded to identify any parsing errors in the "Syntax Checking" phase. Then, if any individuals (scenarios) are made available for testing purposes, the output is also evaluated against those. If any of these fail, we return to the LLM with an indication of the error and attempt a new conversion. If the process is successful, the generated rule is added to the existing ontology and can be tested again.

To guide the LLM, we use a combination of 1-shot and Chain-of-Thought (CoT) prompting, where we break down the task into smaller, manageable steps, providing examples and explanations of each stage. This approach helps to guide the LLM and reduce ambiguities or inconsistencies in the model's output.

Listing 1.1. System Prompt structure

```
# Prompt: Transform Legal Text into OWL
## Objective
Convert the provided legal text into OWL for clarity and precision.

## Step by Step
1. Split the paragraph into sentences.
2. For each sentence identify the following relevant information:
- Subject: Entity (vehicle, person, etc.) that performs the action.
- Conclusion: The action that is present in the main sentence.
- Deontic Modality: Optional. The importance of the action indicated in the
    conclusion. Either, should or must.
```

```
- Conditions: Things that have to be true, in order for the conclusion to be
    applicable.
3. Present the JSON with the extracted elements for each sentence.
4. For each JSON object with conclusion and conditions, *encode the OWL
    representation*.

- **Conclusion**:
* Predicate name is the deontic modality [...]. If not explicit infer from the
    context and add a comment stating so.
* Arguments are the subject, action and specifications, like a location, a time,
     a characteristic.

* Only create new classes or properties, if and only if, the existing ones *do
    not fit*.
* present the conclusion together with the corresponding added class or property

- **Conditions**
* Find the subject, the verb, and the specification.
* Use the verb as the predicate name.
* The arguments are the subject and specifications.

5. *Build the rules* using the identified structure.

## Few Shot examples
[...]

## Final Remarks
[A summary of the instructions above]
```

The translation phase is itself structured, as can be seen in the System prompt in Listing 1.1 The prompt starts by extracting the input text (e.g., a norm from a legal code) and breaking it down into smaller segments, such as sentences or clauses. Each segment is then processed individually, with the LLM being prompted to identify key concepts, such as subjects, actions, conditions, and deontic modalities (e.g., must, may, should). The extracted concepts are then mapped when possible to existing ontology templates (i.e., the example in the prompt). Finally, the structured components are recombined using logical operators (e.g., AND, OR) to reconstruct the legal rule in a formalised, machine-readable format.

To address issues of consistency in the output of an LLM, we include a single example from the domain in question (e.g., the UK Highway Code) to illustrate the expected output format and structure. This example serves as a reference for the model, helping to align its output with the desired ontology representation. Specifically, it provides the model with a concrete reference it can replicate when generating the new output, reducing unwanted linguistic variations and ensuring greater consistency in the logical transformation process. The inclusion of additional information is thus used to explicitly limit the “autonomy” of the model

in both phases (data extraction and generation) in order to minimise ambiguity and hallucination while enforcing strict adherence to the desired structure.

The more structured information is given to the LLM, the better the results. This is true not only of examples (in the 1/multi shot prompting), but also of external information. Currently in the prompt we include one example of a modelled rule, but it might be useful to include additional information, especially for more complex rules.

6 Running Example

6.1 Traffic Rules: The UK Highway Code

As a first example, we consider the UK Highway Code (HC), which contains the rules and guidelines for road users in the United Kingdom. The Highway Code is a comprehensive document that covers various aspects of road safety, including traffic signs, road markings, and rules for different types of vehicles and road users. The Highway Code is regularly updated to reflect changes in traffic laws and regulations, making it a dynamic source of information for road users.

In particular, we consider the section on *Using the Road: Road Junctions* (Rules 170-183), which provides guidance on how to navigate different types of road junctions safely and legally. In our experiment we use Rule 171 (above) is the structured example, which is then used to guide the LLM in forming output of Rules 172 and 175, which are our running examples.

> ***Rule 172***
> *The approach to a junction may have a 'Give Way' sign or a triangle marked on the road. You **MUST** give way to traffic on the main road when emerging from a junction with broken white lines across the road.*
> ***Rule 175***
> *You MUST stop behind the white 'Stop' line across your side of the road unless the light is green. If the amber light appears you may go on only if you have already crossed the stop line or are so close to it that to stop might cause a collision.*

Rules in the HC can describe strict legal requirements or indications for "responsible drivers". Rule 172 is of the first type, as can be evinced from the explicit reference to associated legal provisions. These rules are characterised by specific wording (e.g., must).

6.2 Automated Extraction

With the 1-shot example provided to the LLM, we define a structure for the entire ontology. This is enforced in the multistep process, where we identify the components of the rule, and then map them to the ontology template.

In the example, the key components in rule 171 (in Listing 1.2) are passed as a structured example in the prompt (cf. "Few Shot examples" in Listing 1.1), and the LLM is asked to extract the same components from rule 172.

Listing 1.2. Extracted information from Rule 171

```
"subject":"ego",
"conclusion":"stop behind the line at a junction",
"deontic_modality":"must",
"conditions":[
  "the junction has a 'Stop' sign",
  "the junction has a solid white line across the road"
]
```

The ontology itself is encoded in the Turtle syntax[4], and presented as follows as in Listing 1.3.

Listing 1.3. Representation of Rule 171

```
:VehicleAtJunctionWithStopAndSolidWhiteLineR171CompliantDefault rdf:type
    owl:Class;
  owl:equivalentClass [
    owl:intersectionOf ( [
      rdf:type owl:Restriction;
      owl:onProperty :stopBehindLine;
      owl:hasValue "true"^^xsd:boolean;
    ] [
      rdf:type owl:Restriction;
      owl:onProperty :waitForSafeGap;
      owl:hasValue "true"^^xsd:boolean;
    ] );
    rdf:type owl:Class;
  ];
  rdfs:subClassOf :VehicleAtJunctionWithStopAndSolidWhiteLineR171Compliant.
```

In the ontology structure, there is a main class that represents our agents, "Vehicle", which is then split in subclasses for each rule (e.g., "VehicleR171"). Each subclass contains the restrictions that define the (non)compliant instances for that specific rule. In general terms, the modelled rules are expressed by a main class for that rule, and two disjoint subclasses for the compliant and noncompliant restrictions.

Listing 1.4. Output rule using Llama 4 Maverick

```
:VehicleEmergingFromJunctionWithBrokenWhiteLinesR172CompliantDefault rdf:type
    owl:Class;
  owl:equivalentClass [
    owl:intersectionOf ( [
      rdf:type owl:Restriction;
      owl:onProperty :givesWayToTrafficOnMainRoad;
      owl:hasValue "true"^^xsd:boolean;
    ] );
    rdf:type owl:Class;
  ];
  rdfs:subClassOf :VehicleEmergingFromJunctionWithBrokenWhiteLinesR172Compliant.
```

[4] The specification is available at https://www.w3.org/TR/turtle/

Listing 1.4 shows the output of the LLM for Rule 172, with the new rule classes and *Datatype properties* that describe the actions of the vehicle (equivalent to "stopBehindLine" used in Rule 171).

In this case the differences between the LLMs are minor, and mainly show differences in the Class or Property names (e.g., stop or stops).

If we consider Rule 175, where there is an additional entity (the traffic light) that can assume different colors, we see that certain LLMs make an additional distinction with *Object properties*, related to road infrastructure (e.g., signs, lines, etc), that was not in the single provided example. If this is not the desired behaviour, adding additional examples would improve the output. The output Compliant/Violating classes are modelled in a consistent way in the different models referring to the newly created classes and object properties instead.

Comparing LLMs, this is particularly the case with Gemini 2.5 Flash, where the relation between Junction, Traffic Light, Traffic Light status is encoded through Object Properties, and Classes are created to represent the different situations, as in Listing 1.5. The output from the Llama models is more in line with the structure presented for Rule 172, relying on properties such as "lightIsNotGreen", as in Listing 1.6.

Listing 1.5. Gemini output for rule 175

```
:hasTrafficLight rdf:type owl:ObjectProperty ;
    rdfs:domain :JunctionSituation ;
    rdfs:range :TrafficLight .
:hasStatusValue rdf:type owl:ObjectProperty ;
    rdfs:domain :TrafficLight ;
    rdfs:range :TrafficLightStatus .
:AmberLightSituation rdf:type owl:Class ;
    owl:equivalentClass [
        owl:intersectionOf (
            :JunctionSituation ;
            [
                rdf:type owl:Restriction ;
                owl:onProperty :hasTrafficLight ;
                owl:someValuesFrom [
                    rdf:type owl:Restriction ;
                    owl:onProperty :hasStatusValue ;
                    owl:hasValue :Amber ;
                ] ;
            ]
        ) ;
        rdf:type owl:Class ;
    ] .
```

Listing 1.6. Llama output for rule 175

```
:VehicleAtStopLineR175Compliant rdf:type owl:Class;
  owl:equivalentClass [
```

```
    owl:intersectionOf ( [
      rdf:type owl:Restriction;
      owl:onProperty :mustStopBehindStopLine;
      owl:hasValue "true"^^xsd:boolean;
    ] [
      rdf:type owl:Restriction;
      owl:onProperty :lightIsNotGreen;
      owl:hasValue "true"^^xsd:boolean;
    ] );
    rdf:type owl:Class;
  ];
  rdfs:subClassOf :VehicleAtStopLine.
```

7 Evaluation Method

To validate the output of the LLM, we constructed a gold standard by manual population of an ontology in OWL. We then took the first rule and used it in the 1- shot prompt. The LLM output is compared with the manually created gold standard. This comparison currently is done manually, by checking for the presence of the same classes and properties, and their relationships. In future work we plan to automate this process further, by using ontology matching tools to compare the generated ontology with the gold standard. Tools are available to highlight differences between ontologies, which can be used to assist in identifying missing or incorrect elements in the generated ontology.

In the manual evaluation, we focus on the correctness of the generated ontology, checking for the presence of the same classes and properties, and their relationships. Given the nature of the task we consider a generated ontology to be correct if it captures the same legal norm as the gold standard, even if the exact representation contains syntactic differences. This is important, as there may be minor terminology variants. We hypothesize that the use of multiple examples in the prompt may help in reducing these variations, by providing a more comprehensive view of the desired terms, and that by expanding the ontology with additional background knowledge (e.g., traffic signs) we may be able to further constrain the output.

When evaluating the LLM generated portions of the ontology, especially in comparison with the human generated ones, it is important to highlight the distinction between semantic errors, where the meaning of the modelled rule is different and gives rise to different legal implications, and minor syntactic issues that do not alter the overall meaning. The generated classes could be superfluous or in line with the structure provided. [5]

We plan to expand and formalise the evaluation strategy, focusing on two main factors: a comparison between the output of different LLMs with the same

[5] In the future, this part of the validation could be partially automated through the OOPS! validator [16], with the output fed back in the pipeline triggering another run of the LLM generation.

prompt; the consistency of the LLM output with the human generated rule. In particular, the LLM to LLM comparison would highlight the differences mentioned in Section 6, that derive from the model, not the prompt.

Another tool for evaluation we intend to explore is the generation of specific individuals that act as "scenarios", to validate their assignment to the correct Compliant or Violating class.

8 Conclusion and Future Work

In this paper, we presented a general pipeline for the creation of rule-based legal ontologies with Large Language Models (LLMs). We discussed the use of these ontologies with LLMs to perform legal reasoning and applied it in the context of Autonomous Vehicles (AVs) to ensure compliance with traffic rules. We described the ontology model and how compliant and non-compliant classes are modelled through restrictions on properties.

We plan to improve the pipeline, including a more comprehensive evaluation step-by-step. We also plan to explore the use of the generated ontologies in legal reasoning tasks and to assess their effectiveness in supporting compliance checking and decision-making processes in different legal domains.

Currently, the representation is limited to a subset of norms. Expanding the representation, it will become more important to consider the interaction between different rules, considering potential conflicts of norms, and the treatment of mitigating circumstances [20]. Such cases could be modelled in the system as Disjoint Union subclasses of the "Compliant" Class [9].

Future research avenues include enriching the input for the LLM with an ontology of traffic signs and their meaning, or a list of entities (e.g., vehicles, pedestrians, road types) and their properties (i.e., the sensor abstraction).

Finally, we believe this approach is easily generalizable to other legal domains, so further work will be done in expanding and formalizing the methodology and toolset, ensuring the structure of the proposed pipeline is not tied to a single domain. Changing the domain would mean tweaking the domain specific sections of the prompt, with the data extraction and examples.

References

[1] Kevin D. Ashley. *Artificial Intelligence and Legal Analytics: New Tools for Law Practice in the Digital Age*. Cambridge: Cambridge University Press, 2017. ISBN: 9781107171503. DOI: 10.1017/9781316761380.

[2] Trevor J. M. Bench-Capon and Pepijn R. S. Visser. "Ontologies in legal information systems; the need for explicit specifications of domain conceptualisations". In: *Proceedings of the 6th international conference on Artificial intelligence and law*. ICAIL '97. New York, NY, USA: Association for Computing Machinery, June 30, 1997, pp. 132–141. ISBN: 9780897919241. DOI: 10.1145/261618.261646.

[3] Hanif Bhuiyan et al. “Driving Decision Making of Autonomous Vehicle According to Queensland Overtaking Traffic Rules”. In: *The Review of Socionetwork Strategies* 17.2 (Oct. 2023), pp. 233–254. ISSN: 1867-3236. DOI: 10.1007/s12626-023-00147-x.

[4] Maria Assunta Cappelli and Giovanna Di Marzo Serugendo. “Methodological Exploration of Ontology Generation with a Dedicated Large Language Model”. In: *Electronics* 14.14 (Jan. 2025), p. 2863. ISSN: 2079-9292. DOI: 10.3390/electronics14142863.

[5] Giovanni Ciatto et al. “Large language models as oracles for instantiating ontologies with domain-specific knowledge”. In: *Knowledge-Based Systems* 310 (Feb. 15, 2025), p. 112940. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2024.112940.

[6] Joe Collenette, Louise A. Dennis, and Michael Fisher. “Advising Autonomous Cars about the Rules of the Road”. In: *Electronic Proceedings in Theoretical Computer Science* 371 (Sept. 2022), pp. 62–76. ISSN: 2075-2180. DOI: 10.4204/EPTCS.371.5. arXiv: 2209.14035[cs].

[7] On-Road Automated Driving (ORAD) Committee. *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles*. SAE international, 2021.

[8] Dung V. Dang et al. “Information Retrieval from Legal Documents with Ontology and Graph Embeddings Approach”. In: *Advances and Trends in Artificial Intelligence. Theory and Applications*. Ed. by Hamido Fujita et al. Cham: Springer Nature Switzerland, 2023, pp. 300–312. ISBN: 9783031368196. DOI: 10.1007/978-3-031-36819-6_27.

[9] Enrico Francesconi and Guido Governatori. “Patterns for legal compliance checking in a decidable framework of linked open data”. In: *Artificial Intelligence and Law* 31.3 (Sept. 1, 2023), pp. 445–464. ISSN: 1572-8382. DOI: 10.1007/s10506-022-09317-8.

[10] Giancarlo Guizzardi et al. “UFO: Unified Foundational Ontology”. In: *Applied Ontology* 17.1 (Mar. 2022). Ed. by Stefano Borgo, Antony Galton, and Oliver Kutz, pp. 167–210. ISSN: 1570-5838. DOI: 10.3233/ao-210256.

[11] Lu Huang et al. “Ontology-Based Driving Scene Modeling, Situation Assessment and Decision Making for Autonomous Vehicles”. In: 2019 4th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS). July 2019, pp. 57–62. DOI: 10.1109/ACIRS.2019.8935984.

[12] Michael Hülsen, J. Marius Zöllner, and Christian Weiss. “Traffic intersection situation description ontology for advanced driver assistance”. In: 2011 IEEE Intelligent Vehicles Symposium (IV). ISSN: 1931-0587. June 2011, pp. 993–999. DOI: 10.1109/IVS.2011.5940415.

[13] Valentina Leone, Luigi Di Caro, and Serena Villata. “Taking stock of legal ontologies: a feature-based comparative analysis”. In: *Artificial Intelligence and Law* 28.2 (June 1, 2020), pp. 207–235. ISSN: 1572-8382. DOI: 10.1007/s10506-019-09252-1.

[14] Xiaohan Ma et al. “Law compliance decision making for autonomous vehicles on highways”. In: *Accident Analysis & Prevention* 204 (Sept. 1, 2024), p. 107620. ISSN: 0001-4575. DOI: 10.1016/j.aap.2024.107620.

[15] Kumar Manas, Mert Keser, and Alois Knoll. *Integrating Legal and Logical Specifications in Perception, Prediction, and Planning for Automated Driving: A Survey of Methods*. 2025. DOI: 10.48550/ARXIV.2510.25386.

[16] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. “OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation”. In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 10.2 (2014), pp. 7–34.

[17] Ron Provine et al. “Ontology-based methods for enhancing autonomous vehicle path planning”. In: *Robotics and Autonomous Systems*. Knowledge Engineering and Ontologies for Autonomous Systems 2004 AAAI Spring Symposium 49.1 (Nov. 2004), pp. 123–133. ISSN: 0921-8890. DOI: 10.1016/j.robot.2004.07.020.

[18] Ralf Regele. “Using Ontology-Based Traffic Models for More Efficient Decision Making of Autonomous Vehicles”. In: Fourth International Conference on Autonomic and Autonomous Systems (ICAS’08). ISSN: 2168-1872. Mar. 2008, pp. 94–99. DOI: 10.1109/ICAS.2008.10.

[19] Albert Rizaldi et al. “Formalising and Monitoring Traffic Rules for Autonomous Vehicles in Isabelle/HOL”. In: *Integrated Formal Methods*. Ed. by Nadia Polikarpova and Steve Schneider. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 50–66. ISBN: 9783319668451. DOI: 10.1007/978-3-319-66845-1_4.

[20] Galileo Sartor and Adam Wyner. “A Legal Logic Programming Framework for Autonomous Vehicles”. In: *Joint Proceedings of the Workshops and Doctoral Consortium of the 41st International Conference on Logic Programming (ICLP-WS-DC 2025)*. Workshop on Logic Programming and Legal Reasoning (LPLR 2025). Sept. 12, 2025.

[21] Simon Ulbrich et al. “Graph-based context representation, environment modeling and information aggregation for automated driving”. In: 2014 IEEE Intelligent Vehicles Symposium. ISSN: 1931-0587. June 2014, pp. 541–547. DOI: 10.1109/IVS.2014.6856556.

[22] Tom Van Engers et al. “Ontologies in the Legal Domain”. In: ed. by Hsinchun Chen et al. Boston, MA: Springer US, 2008, pp. 233–261. ISBN: 9780387716114. DOI: 10.1007/978-0-387-71611-4_13.

[23] Adam Z. Wyner and Wim Peters. “On Rule Extraction from Regulations”. In: *Legal Knowledge and Information Systems - JURIX 2011: The Twenty-Fourth Annual Conference, University of Vienna, Austria, 14th-16th December 2011*. Ed. by Katie Atkinson. Vol. 235. Frontiers in Artificial Intelligence and Applications. IOS Press, 2011, pp. 113–122. DOI: 10.3233/978-1-60750-981-3-113.

[24] Jingyuan Zhao et al. “Autonomous driving system: A comprehensive survey”. In: *Expert Systems with Applications* 242 (May 2024), p. 122836. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2023.122836.

Plans and Diversions

Galileo Sartor[1 [0000−0001−6355−851X]], Guido Governatori[2 [0000−0002−9878−2762]], Giuseppe Pisano[3 [0000−0003−0230−8212]], Antonino Rotolo[2 [0000−0001−5265−0660]], and Adam Wyner[1 [0000−0002−2958−3428]]

[1] Swansea University, UK galileo.sartor,a.z.wyner@swansea.ac.uk
[2] University of Bologna, Italy g.pisano,antonino.rotolo@unibo.it
[3] Central Queensland University, Australia ggovernatori@csu.edu.au

Abstract. An agent in a context acts to bring about goals, formulating a path from the agent's current state to a goal state via intermediate states. For our purposes, some of an agent's actions may be legally constrained. For example, in the transportation domain, autonomous vehicles (AVs) must abide by the Highway Code. Thus, plans ought to incorporate legal reasoning. An optimal, deterministic, law abiding plan could be found, e.g., that plan with the fewest stoplights in a predetermined state (red or green). However, such a plan does not account for contingencies, i.e., non-deterministic states, where the stoplight's value is only determined in the state at a time. In the state, the agent may be faced with a choice between violating the law or finding and choosing an alternative plan which would not induce a violation - a diversion. Defeasible Deontic Logic (DDL) has been used for legal reasoning, and planning has been implemented in Answer Set Programming (ASP); an ASP encoding of DDL has been proposed and used for planning with respect to optimal plans. The novel contribution of this paper is to introduce diversions of plans for AVs which address contingencies with respect to legal reasoning.

Keywords: defeasible deontic logic · answer set programming · planning · autonomous vehicles

1 Introduction

Planning is critical in many domains in order to attain goals, e.g., project, business, or operations management. We can make plans prior to execution of the plan - *static plans*. However, things may not go according to plan in the course of executing it, as unaccounted for contingencies may arise. To address them, mitigating strategies may be deployed, whether arranged in advance or on the fly. Such strategies may introduce a plan, which is a variant on the original one; in effect, we have *dynamic plans*. We refer to such variant plans as *diversions*, which accommodate contingencies.

In the legal setting, static and dynamic planning are also critical, e.g., in litigation strategy, contracts, financial transactions, and others. What is critical in *legal planning* is that the action executed in a given state also abides by legal norms - what is obligated, prohibited, or permitted. The norms specify which actions violate the norms and (depending on mitigations) lead to sanctions and reparations.

Given an environment, an agent's plan and behaviour may vary depending on its *attitude*. For instance, one agent type may not tolerate violation of any legal norm while tolerating longest (or slowest) plans - *agens legalis*. Another agent type may want the shortest (or fastest) plan while tolerating violations - *agens economicus*. Other agents may have different trade-offs between violations, time, other costs. As it is complex to define the attitude itself, we only consider the preference for one type of plan or another.

As a use case in the analysis, formalisation, and implementation of legal plans with diversions, we consider *autonomous vehicles* (AVs) in an environment which can represent core features. Autonomous vehicles are particularly attractive as we can explicitly represent: plans as a series of states from a start state to a goal state; actions from state to state; agents that execute actions; legal norms; and contingencies such as whether a light in a state is red or green.

Planning has been a long-standing topic in Artificial Intelligence with wide application. There is recent research on using LLMs in planning [28]. Yet, for the legal domain, precision, logical correctness, controllability, and explainability are paramount virtues that are not available in LLMs but are in logic-based approaches. Research in Artificial Intelligence and Law on legal planning has been limited to date. [19] mentions legal planning and that prior work, e.g., [26], which concerned constraint satisfaction of building plans (static plans). Beyond that, relevant literature is hard to come by (see Section 5 for additional discussion).

Taking a logic-based approach, we have well-developed and powerful tools of Answer Set Programming (ASP) [27], which generate plans, and Defeasible Deontic Logic (DDL), which represents and reasons with legal knowledge. DDL has been encoded in ASP [6], and integrated with ASP planners, DDL-ASP planner, as outlined below.

This paper contributes to the DDL-ASP planner by enabling it to reason with contingencies in order to dynamically propose diversions to a pre-existing global plan. [4]. The system is a step towards richer, more complex legal planning systems. The frame of the research is scoped to focus on fundamental aspects: one type of legal signal - the stoplight; one prohibition (on crossing a red light); a choice of balance between *agens legalis* and *agens economicus*; no reasoning behind the type of agent; limited cost/benefit analysis; a small environment; and no mitigations. While scoped, the system is the basis for future work.

The paper first provides background on DDL-ASP (Section 2) and planning with DDL (Section 3). In Section 4, autonomous vehicles with DDL-ASP and planning is presented, giving code snippets and worked examples. Related work is outlined in 5. Section 6 concludes and outlines future work.

2 Defeasible Deontic Logic

Defeasible Deontic Logic (DDL) is an efficient rule-based non-monotonic formalism for legal reasoning. It combines features of Defeasible Logic for the natural and efficient modelling of exceptions with deontic logic, including deontic operators (obligation and permission), and the ability to reason with violations and compensatory obligations.

[4] The ASP and Python scripts are available at https://github.com/Gilbocc/asp-av-planning.

The language of DDL is built from a set of atomic propositions ($\{p_1, p_2, \dots\}$). The logic distinguishes between plain literals (which are either atomic propositions or the negation of atomic propositions) and deontic literals. A deontic literal is a plain literal that falls within the scope of either a deontic operator or a negated deontic operator. The deontic operators are O for obligation and P for permission.

A theory in DDL is a triple

$$(F, R, >)$$

where F is a set of facts (a set of literals), R is a set of rules, and $>$ is a binary relation called *superiority relation* over the set of rules. The rules are partitioned in *constitutive rules*, *prescriptive rules* and *permissive rules*. A rule is an expression

$$r : a_1, \dots, a_n \Rightarrow_X c$$

where r is the label or name of the rule, and it is unique for each rule; $a_1, \dots, a_n$ are the premises or the antecedent of the rule (the set of the premises of a rule can be empty); each a_i is either a literal or a deontic literal; X is the mode of the rule. The idea behind DDL is that the conclusion of a rule is the effect of the rule, and the type of the effect depends on the mode of the rule, which is that the mode specifies the type of conclusion produced by the rule. In addition, rules are defeasible in the sense that we can assert the conclusion of a rule unless there are other rules against it. For a constitutive rule $X = \mathsf{C}$, the conclusion is a factual statement, a statement that can be evaluated as true or false in the given scenario. For a permissive rule $X = \mathsf{P}$, the rule establishes that the conclusion is permitted. Finally, for prescriptive rules $X = \mathsf{O}$, when a rule is applicable and not defeated, the conclusion is an obligation in force in the given state. In addition, the conclusion of a prescriptive rule can be an expression $b_1 \otimes b_2 \otimes \cdots \otimes b_m$. The meaning of such expressions is, if the rule is applicable and there are no rules against it (the various elements of the sequence), that b_1 is obligatory (i.e., $\mathsf{O}b_1$ holds), and if it is violated, meaning that $\sim b_1$ holds as well, then b_2 is obligatory, and doing it compensates for the violation of b_1. Moreover, it is recursive. Thus, if $\mathsf{O}b_2$ holds and if it is violated, $\mathsf{O}b_3$ holds, and complying with it compensates for the violation of b_2, and so on. Thus, b_m is the last chance to comply with the norm/rule producing the sequence $b_1 \otimes b_2 \otimes \cdots \otimes b_m$.

The superiority relation allows us to establish which rule overrides another rule, when the two rules are both applicable and for conflicting conclusions.

In this paper we adopt the implementation of DDL in ASP presented in [6] ASP and DDL are two approaches to rule-based non-monotonic reasoning. However, they have different semantics, thus, it is not possible to represent DDL in ASP since they have different interpretations of negation. Moreover, [7] proves that Stable Semantics, the underlying semantics of ASP, is not suitable to model certain aspects of legal reasoning. This means that we cannot model directly norms in ASP, but we can use a meta-programming approach to capture DDL in ASP based on the framework proposed in [3] and extended to the case of Defeasible Logic with modal operator [9]. Accordingly, we define constructs to treat atoms, negation, and defeasible rules in ASP. In addition, we have to define the ASP clause to capture the DDL reasoning mechanism.

Here we recall how to encode a theory in DDL in ASP following the implementation of DDL in ASP proposed in [6]. The first step is to declare the language, which means

that for each atomic proposition p, we have to include the clause

$$\mathrm{atom}(p).$$

A rule

$$r\colon l_1,\ldots,l_m,\mathsf{O}o_1,\ldots\mathsf{O}o_n,\neg\mathsf{O}no_1,\ldots\neg\mathsf{O}no_k,\mathsf{P}p_1,\ldots\mathsf{P}p_w,\neg\mathsf{P}np_1,\ldots\neg\mathsf{P}np_z \Rightarrow_X c$$

is encoded as

```
<mode>Rule(r,c).
applicable(r,c) :-
   defeasible(l_1), ... , defeasible(l_m),
   obligation(o_1), ... , obligation(o_n),
   not obligation(no_1), ... , not obligation(no_k),
   permission(p_1), ... , permission(p_w),
   not permission(np_1), ... , not permission(np_z).
```

where <mode> is a placeholder that stands for constitutive, prescriptive or permissive according to the value of X.

When the conclusion of a prescriptive rule is an expression $c_1 \otimes c_2 \otimes \cdots \otimes c_m \otimes c_n$. The rule is rendered as

```
prescriptiveRule(r,c_1).
...
compensate(r,c_1,c_2,1).
compensate(r,c_2,c_3,2).
...
compensate(r,c_m,c_n,n-1).
```

The ASP implementation of the DDL reasoning defines clauses for defeasible/1, obligation/1, permission/1. The clauses capture in ASP the proof conditions for $+\partial_{\mathsf{C}}$, $+\partial_{\mathsf{O}}$ and ∂_{P}.

Notice that the implementation allows for literals built from predicates. However, ASP requires specifying the domain of quantification to ensure safe grounding; for such a case, we extend the clauses with a predicate that allows us to specify the domain. Thus we can have clauses like (see the next section for concrete examples):

$$\mathrm{atom(p(X,Y))}\ \text{:-}\ \mathrm{domainX(X)},\ \mathrm{domainY(Y)}.$$

3 Planning with DDL

A planning problem in ASP is defined as a tuple:

$$\mathcal{P} = \langle S, A, T, s_0, G\rangle$$

where:

- S is the set of possible world states, represented as collections of fluents.
- A is the set of actions, each with preconditions and postconditions.
- $T \subseteq S \times A \times S$ is the transition relation defined by the effects of actions.
- $s_0 \in S$ is the initial state.
- $G \subseteq S$ is the set of goal states.

In ASP, this is encoded over a bounded time horizon $t = 0 \ldots H$ as following. The **initial state** is encoded by specifying which fluents hold at time $t = 0$. This is done by asserting that the fluents in s_0 hold at time 0:

$$\text{state}(0, f) \leftarrow \text{ for each fluent } f \in s_0$$

The **executability of an action** at time t is determined by the state of the world at that time. An action a is executable at time t if all its preconditions hold in the state at time t. This is captured by the following rule:

$$\text{trace}(a, t) \leftarrow \text{state}(t, f_1), \ldots, \text{state}(t, f_n)$$

where $\text{trace}(a, t)$ indicates that action a is executed at time t.

The **effects of an action** on the state are encoded by the postconditions of the action. If an action a is executed at time t, it updates the state at time $t + 1$. Some fluents are added (i.e., the action causes a fluent to hold), while others are deleted (i.e., the action causes a fluent to no longer hold). This is represented as:

$$\text{state}(t + 1, f) \leftarrow \text{trace}(t, a), \text{causes}(a, f)$$
$$\text{-state}(t + 1, f') \leftarrow \text{trace}(t, a), \text{deletes}(a, f')$$

Frame axiom encode world inertia, meaning that fluents that are not affected by an action remain unchanged. If a fluent f holds at time t, and no action deletes it, it will also hold at time $t + 1$:

$$\text{state}(t + 1, f) \leftarrow \text{state}(t, f), \text{ not } \text{-state}(t + 1, f)$$

The **goal condition** specifies fluents that must hold at the end of the planning process at time $t <= H$:

$$\leftarrow \text{not } (\text{state}(t, f_1), \ldots, \text{state}(t, f_k))$$

The **action choice rule** specifies that, at each time step, at most one action must be chosen to be executed:

$$\{\text{trace}(t, a) : a \in A\}1 \leftarrow \text{time}(t)$$

Every stable model produced by the ASP reasoner corresponds to a valid plan. This set of plans can be refined using an optimization strategy. In our methodology, legal norms are encoded in ASP via Defeasible Deontic Logic, so the resulting normative concepts, such as obligations, violations, and permissions, are also included in the stable model. As a result, planning can be easily constrained to reflect desired behaviours by constraining the propositions present in the model. For example, we may choose to discard all plans that contain violations, thereby enforcing strict normative compliance.

4 Autonomous Vehicles

The complexity of autonomous driving motivates a hierarchical planning architecture across global, strategic route selection and local, context-aware decision making. A global planner cannot account for contingencies, e.g., status of traffic lights, other road

users, etc.; a local planner has no strategic reasoning. Note that we do not consider a sanitized environment, where the vehicles know ahead of time what they will encounter.

The *global* and *local* planner share the basic world model. The distinction between the planners lies in their different access to the states.

- The **long-term planner** operates under uncertainty, as the agent is not aware about the contingencies. It reasons over a static world model, aware of the *existence* of dynamic elements (like traffic lights), but ignorant of their state at a specific future time. The long-term planner identifies the optimal path minimising the total distance and an additional weight added by the existence of traffic lights, which are interpreted as a potential delay in time. Given this, the planner can only approximate the time to complete a path.
- The **short-term planner** has access to the perceived state of the world (e.g., traffic light state). Its knowledge is more complete, but is localized to a short window. Crucially, the short-term planner finds the local plan optimised by norm compliance, which means that the plan minimises the number of norm violations which would arise were the actions of the plan to be executed.

For this paper, minimisation is a constraint on the short-term planner. An optimal short term plan may still contain a violation, leaving the vehicle to decide if such a plan is acceptable. In the example, the vehicle avoids violations and recalculates a long-term plan. In future work, we will explore issues related to mitigating circumstances.

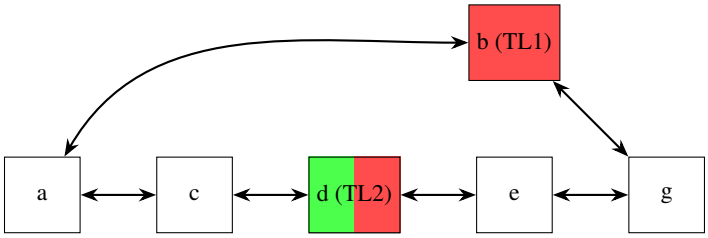


Fig. 1. Graph of the World model

The program divides labour between legal plans and the actions performed by the agent. In particular, the ASP code encodes the traffic rules and plans, while the Python program manages the dynamics changing in the world. It uses the *clingo* Python library, calling the Clingo ASP solver to generate the plans when needed. The global planner generates a proposed plan that is used as input for the local planner in a shorter window.

1. If the local plan would be compliant, i.e., no violations occur were the agent to execute actions from the current state, then the actions of the local plan are executed, the window is shifted, and the local planner is called on the next actions.
2. If the plan is deemed not acceptable on the basis of the local planner's preference, the global planner is invoked to find alternatives.

```
% Preconditions
action(T, move(Y)) :- fact(state(T, position(X))), next(T, X, Y)
    , time(T+1).
action(T, invert(X)) :- fact(state(T, position(X))), time(T+1).

% Postconditions
fact(state(T+1, position(X))) :- trace(T, move(X)).
fact(state(T+1, direction(ND))) :- trace(T, invert(X)), fact(
    state(T, direction(D))), invert(D, ND).

-fact(state(T+1, position(D))) :- trace(T, move(_)), fact(state(
    T, position(D))).
-fact(state(T+1, direction(D))) :- trace(T, invert(_)), fact(
    state(T, direction(D))).

% Inertia
fact(state(T+1, X)) :- fact(state(T, X)), time(T+1), not -fact(
    state(T+1, X)).
```

Listing 1.1. Vehicle actions

The environment is modelled as a directed graph consisting of states, some of which contain traffic lights (Figure 1). The agent can move forward or backwards between adjacent nodes (Listing 1.1). The *available actions* are modelled in terms of preconditions and postconditions. If necessary the vehicle can also skip an action, stopping in the current node. Where the action is taken, the agent's new position and direction are updated. In the world description, traffic lights are modelled as either static or dynamic. A permanent red light is present at location *b*, while the light at location *d* alternates between green and red (indicated as half green to red), depending on the simulation time. The connection between locations and current direction of the agent determines which transitions are possible at any given step. The graph is sufficient to show the core aspects of the system. Future work will include more complex environments and simulations.

```
% Maximum 1 action X per time T
0 { trace(T, X) : action(T, X) } 1 :- time(T).

% Must reach position g
goal :- fact(state(_, position(g))).
:- not goal.

% Weight assigned to a traffic light
weight(X, 2) :- fact(traffic_light(_, X)).

#minimize { T@1, T, S : trace(T, S) }.
#minimize { W@1, T, S, W : trace(T, move(S)), weight(S, W) }.
```

Listing 1.2. Global Planner

The global planner (Listing 1.2) reads the world state and generates a plan which minimises the time needed to reach the final goal *g* and the potential delay added by the

obstacles (e.g., traffic lights). For this, it uses the graph representing the world and the weight that on obstacles (Listing 1.2).

```
% Rule r1/r2: Obligation to stop at traffic lights if red and
    not emergency
prescriptiveRule(r1, stop(T, X)) :- tf(T, X).
applicable(r1, stop(T, X)) :- defeasible(traffic_light(ID, X)),
    defeasible(state(T, traffic_light(ID, red))), defeasible(
    state(T, direction(forward))).
compensate(r1, stop(T, X), pay_fine, 1) :- tf(T, X).

constitutiveRule(r2, non(stop(T, X))) :- tf(T, X).
applicable(r2, non(stop(T, X))) :- obligation(stop(T, X)),
    defeasible(state(T, position(X))), defeasible(trace(T, move(
    _))).
convertPermission(r2, non(stop(T, X))) :- defeasible(traffic_
    light(ID, X)), defeasible(state(T, traffic_light(ID, red))),
     defeasible(emergency).

fact(trace(T, move(X))) :- trace(T, move(X)).

atom(pay_fine).
atom(stop(T, X)) :- tf(T, X).

tf(T, X) :- fact(traffic_light(ID, X)), fact(state(T, traffic_
    light(ID, red))).

% Planning Problem
0 { trace(T, X) : action(T, X) } 1 :- time(T).

#minimize { 10@2 : obligation(pay_fine) }.
#minimize { T@1, T, S : trace(T, S) }.
```

Listing 1.3. Local Planner

The short term planner has access to the traffic regulation as modelled in Listing 1.3. The agent operates under a set of norms that regulate its behaviour. A prescriptive rule (r1) obligates the agent to stop at red traffic lights, with a monetary fine serving as compensation in cases of non-compliance. A constitutive rule (r2) identifies the conditions under which we have a failure to stop, namely, when the agent is obligated to stop yet acts. Exceptions are modelled using a permissive rule (r2) that allows the agent to disregard red lights during emergency situations. The short-term planner evaluates the states and actions *prospectively* and *hypothetically* with respect to the norms; that is, where the execution of an action in a state were to lead to a violation, we say that the plan contains a violation. In such a case, a new long term plan will be calculated.

The short term planner considers dynamics, represented in context dependent information at runtime by the Python script. This information is included in the ASP theory before the grounding phase. This includes: the updated position, the current direction, the window size, the goal, and any mitigating circumstances such as being in an emergency

We use the ASP optimization mechanisms so the local planner can optimize in different ways, depending on the explicit optimization criteria: minimising the number of violations; minimising the time to reach the goal; or some combination. In the Python code, the optimization is set when calling the local planner. While in the current state the reasoning with violations and optimisation is limited, in future developments it will be made more context dependent.

We consider two potential scenarios in relation to Figure 1, varying the number of time steps the local planner can see ahead and the total time the vehicle allows for the short plan. In both cases the vehicle starts at node *a*, point forward, and aims to reach node *g*. The vehicle is not in an emergency, so it must comply with the traffic rules.

In Listing 1.4, we set the visible window to 2 time steps, and the total time (for the short planner) to 4. The global planner proposes passing through nodes *b* and *g*. The local planner then checks the possible routes and finds that there is no way to avoid a violation. This is because of the traffic light in b, fixed on red. The global planner is invoked again, avoiding the previous intersection that caused the violation, and proposes a new plan: *c, d, e, g*. The local planner analyses the first leg (the first two moves, according to the window size), and accepts the move *c, d*, as there are no violations, and repeats the evaluation for the next leg, accepting *e, g*, reaching its destination. The vehicle does not perform any action at time 2, as the optimal plan includes waiting at the red light.

```
trace(0,move(c))
trace(1,move(d))
trace(3,move(e))
trace(4,move(g))
```

Listing 1.4. Path example 1

```
trace(0,move(b))
trace(1,invert(b))
trace(2,move(a))
trace(3,invert(a))
trace(4,move(c))
trace(5,move(d))
trace(7,move(e))
trace(7,move(g))
```

Listing 1.5. Path example 2

In Listing 1.5, we obtain a different routing where the window is set to 1, and the time to 6. In this case the local planner is only looking at one action at a time. The first proposed global plan is *b, g*, but now only the first action *b* is evaluated by the local planner, and no violation is detected. The action is executed, and the vehicle moves to *b*. The local planner is called again, and now the only action in the window is *g*. Here the traffic light violation is detected, and the global planner is invoked this time with a different starting point, *b*. The new plan is *invert(b), a, invert(a), c, d, e, g*. The local planner evaluates the first move action, to *a*, which is accepted and executed. The vehicle moves to *a*, and the local planner is called again. This continues, with no further violations, and the vehicle reaches its destination.

The analysis illustrates the benefits, in terms of simplicity and consistency, of a shared DDL-ASP logical representation which integrates the environment, actions, planning, and with reasoning about norms.

5 Related Work

We outline the few recent contributions on normative planning.

5.1 DDL

[22] addresses planning in normative environments using the PDDL planning framework; and [21] formalise it in Linear Temporal Logic. However, the expressiveness of both approaches is limited, as the planner will never produce a plan in which a norm instantiation is violated and remains uncompensated for. Additionally, they do not include constitutive rules in their formalisation.

Answer Set Programming (ASP) has been established as a powerful formalism for declarative problem-solving. ASP-based planning systems have benefited from continuous performance improvements of ASP solvers as well as support for heuristics and constraint-solving techniques to further enhance efficiency [27].

However, there are still few contributions in the area of normative planning. For instance, [11] introduces an ASP-based framework that encodes legal norms using weak constraints, enabling the resolution of well-known deontic paradoxes. Their framework does not account for compensatory obligations. Real-time norm-aware planning is presented in [1], where ASP is used in run-time to generate emergency plans to repair policies that violate norms. While the planner minimises the number of violations, the formalism does not have constitutive and compensatory rules as in DDL.

DDL has already been used for planning in [5, 14, 20]. [5] represents the mental attitudes of agents; it selects from amongst pre-computed plans. Similarly, [14] has an agent with attitudes and plans routes to avoid collisions. In [20], a normative supervisor (implemented in SPINdle [13]) uses reinforcement learning to remove actions that do not comply with the norms encoded.

Notably, these developments do not represent diversions with respect to norms.

5.2 AVs

We briefly survey relevant AV literature. As mentioned in [23, 15], traffic rules may be more context dependent recommendations rather than absolute constraints. For situational reasoning, AVs may require background knowledge or commonsense reasoning, as suggested in [12]. [24] considers drivers violating rules without consequential penalties. Drivers alter their behaviour in view of the entire socio-technical system.

Planning for AVs is generally limited to trajectory prediction, with systems integrating formal methods with Reinforcement Learning or LLMs [18]. AV evaluation of rule compliance and violations appears in [16]. A reinforcement learning appoach is taken, which triggers, re-planning when encountering potential violations. The encoding is based on norms as constraints on trajectories.

There are various representations of traffic rules. [4] models rules and actions in a Belief-Desire-Intention framework in Prolog. The BDI model is often used to trace the behaviour of the AV, as in [2]. [17] focusses on rules for trajectory monitoring. The encoding can be more or less isomorphic with regard to the source text, sometimes using controlled natural languages as in [25]. The goal is to model all traffic laws, e.g., the UK Highway Code. However, such research does not use expressive normative reasoners.

6 Conclusions

This work present a unified framework for integrating planning and normative reasoning using Answer Set Programming and Deontic Defeasible Logic to model obligations, permissions, and violations alongside the agent's operational capabilities. By embedding normative constraints into the planning model, the system reasons about goal achievement and compliance in a coherent, integrated, and transparent way. The approach flexibly compares different agent behaviours, supporting legal and context-sensitive decision-making without requiring separate modules for normative evaluation of plans.

In future work, the patterns of violation, penalty, and reparation will be incorporated as well as reasoning with respect to mitigations. Agents will consider their *tolerance towards risk* in making plans. The defeasible reasoning will incorporate the richer logics of [8, 10], which accounts for different agentive modalities, levels of goal achievement, and attitudes towards norm compliance. We intend to develop quantitative analysis of performance, with the aim of assessing the computational overhead introduced by the DDL component. This will evaluate the trade-offs between normative expressiveness and planning efficiency in larger or more dynamic environments.

References

[1] Sebastian Adam and Thomas Eiter. ''ASP-Driven Emergency Planning for Norm Violations in Reinforcement Learning''. In: *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*. Ed. by Toby Walsh, Julie Shah, and Zico Kolter. AAAI Press, 2025, pp. 14772–14780. DOI: 10.1609/AAAI.V39I14.33619.

[2] Gleifer Vaz Alves, Louise Dennis, and Michael Fisher. ''A Double-Level Model Checking Approach for an Agent-Based Autonomous Vehicle and Road Junction Regulations''. In: *Journal of Sensor and Actuator Networks* 10.3 (Sept. 2021), p. 41. ISSN: 2224-2708.

[3] Grigoris Antoniou et al. ''Embedding Defeasible Logic into Logic Programming''. In: *Theory and Practice of Logic Programming* 6.6 (2006), pp. 703–735. DOI: 10.1017/S1471068406002778.

[4] Joe Collenette, Louise A. Dennis, and Michael Fisher. ''Advising Autonomous Cars about the Rules of the Road''. In: *Electronic Proceedings in Theoretical Computer Science* 371 (Sept. 2022), pp. 62–76. ISSN: 2075-2180. DOI: 10.4204/EPTCS.371.5. arXiv: 2209.14035[cs].

[5] Mehdi Dastani et al. ‘‘Programming Cognitive Agents in Defeasible Logic’’. In: *12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning* (Montego Bay, Jamaica, Dec. 2–6, 2005). Ed. by Geoff Sutcliffe and Andrei Voronkov. Vol. 3835. LNAI. Heidelberg: Springer, 2005, pp. 621–636. DOI: 10.1007/11591191_43.

[6] Guido Governatori. ‘‘An ASP Implementation of Defeasible Deontic Logic’’. In: *Künstliche Intell.* 38.1-2 (2024), pp. 79–88. DOI: 10.1007/s13218-024-00854-9.

[7] Guido Governatori. ‘‘Weak Permission is not Well-Founded, Grounded and Stable’’. In: *CoRR* abs/2411.10624 (2024). DOI: 10.48550/ARXIV.2411.10624. arXiv: 2411.10624 [cs.LO].

[8] Guido Governatori and Antonino Rotolo. ‘‘BIO Logical Agents: Norms, Beliefs, Intentions in Defeasible Logic’’. In: *Journal of Autonomous Agents and Multi Agent Systems* 17.1 (2008), pp. 36–69. DOI: 10.1007/s10458-008-9030-4.

[9] Guido Governatori and Antonino Rotolo. ‘‘Defeasible Logic: Agency, Intention and Obligation. Deontic Logic in Computer Science’’. In: *7th International Workshop on Deontic Logic in Computer Science*. Ed. by Alessio Lomuscio and Donald Nute. LNAI 3065. Berlin: Springer, 2004, pp. 114–128. DOI: 10.1007/978-3-540-25927-5_8.

[10] Guido Governatori et al. ‘‘The Rational behind the Concept of Goal’’. In: *Theory and Practice of Logic Programming* 16.3 (2016), pp. 296–324. DOI: 10.1017/S1471068416000053.

[11] Christian Hatschka, Agata Ciabattoni, and Thomas Eiter. ‘‘Deontic Paradoxes in ASP with Weak Constraints’’. In: *Proceedings 39th International Conference on Logic Programming, ICLP 2023, Imperial College London, UK, 9th July 2023 - 15th July 2023*. Ed. by Enrico Pontelli et al. Vol. 385. EPTCS. 2023, pp. 367–380. DOI: 10.4204/EPTCS.385.39.

[12] Suraj Kothawade et al. ‘‘AUTO-DISCERN: Autonomous Driving Using Common Sense Reasoning’’. In: *International Conference on Logic Programming 2021 Workshops*. Ed. by Joaquin Arias et al. Vol. 2970. CEUR Workshop Proceedings. Porto, Portugal (virtual): CEUR, Sept. 2021.

[13] Ho-Pun Lam and Guido Governatori. ‘‘The Making of SPINdle’’. In: *International Symposium on Rule Interchange and Applications* (Las Vegas, Nevada, USA, Nov. 5–7, 2009). Ed. by Guido Governatori, John Hall, and Adrian Paschke. LNCS 5858. Heidelberg: Springer, 2009, pp. 315–322. DOI: 10.1007/978-3-642-04985-9_29. eprint: papers/2009/ruleml09spindle.pdf.

[14] Ho-Pun Lam and Guido Governatori. ‘‘Towards a Model of UAVs Navigation in Urban Canyon through Defeasible Logic’’. In: *Journal of Logic and Computation* 23.2 (2013), pp. 373–395. DOI: 10.1093/logcom/exr028.

[15] Ronald E. Leenes and Federica Lucivero. ‘‘Laws on Robots, Laws by Robots, Laws in Robots: Regulating Robot Behaviour by Design’’. In: *Law, Innovation and Technology* 6(2).2546759 (Nov. 2014).

[16] Jiaxin Lin et al. ‘‘Road Traffic Law Adaptive Decision-making for Self-Driving Vehicles’’. In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. 2022 IEEE 25th International Conference on Intel-

ligent Transportation Systems (ITSC). 2022, pp. 2034–2041. DOI: 10.1109/ITSC55140.2022.9922208.

[17] Sebastian Maierhofer, Paul Moosbrugger, and Matthias Althoff. ''Formalization of Intersection Traffic Rules in Temporal Logic''. In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. 2022 IEEE Intelligent Vehicles Symposium (IV). June 2022, pp. 1135–1144. DOI: 10.1109/IV51971.2022.9827153.

[18] Kumar Manas and Adrian Paschke. ''Knowledge Integration Strategies in Autonomous Vehicle Prediction and Planning: A Comprehensive Survey''. In: *CoRR* abs/2502.10477 (2025). DOI: 10.48550/ARXIV.2502.10477. arXiv: 2502.10477.

[19] L. Thorne McCarty. ''Artificial Intelligence and Law: How to Get There from Here''. In: *Ratio Juris* 3 (1990), pp. 189–200.

[20] Emery Neufeld et al. ''Enforcing Ethical Goals over Reinforcement-Learning Policies''. In: *Ethics and Information Technology* 24 (2022). DOI: 10.1007/s10676-022-09665-8.

[21] Sofia Panagiotidi, Sergio Álvarez-Napagao, and Javier Vázquez-Salceda. ''Towards the Norm-Aware Agent: Bridging the Gap Between Deontic Specifications and Practical Mechanisms for Norm Monitoring and Norm-Aware Planning''. In: *Coordination, Organizations, Institutions, and Norms in Agent Systems IX - COIN 2013 International Workshops, COIN@AAMAS, St. Paul, MN, USA, May 6, 2013, COIN@PRIMA, Dunedin, New Zealand, December 3, 2013, Revised Selected Papers*. Ed. by Tina Balke et al. Vol. 8386. Lecture Notes in Computer Science. Springer, 2013, pp. 346–363. DOI: 10.1007/978-3-319-07314-9_19.

[22] Sofia Panagiotidi and Javier Vázquez-Salceda. ''Towards Practical Normative Agents: A Framework and an Implementation for Norm-Aware Planning''. In: *Coordination, Organizations, Institutions, and Norms in Agent System VII, COIN 2011 International Workshops, COIN@AAMAS 2011, Taipei, Taiwan, May 3, 2011, COIN@WI-IAT 2011, Lyon, France, August 22, 2011, Revised Selected Papers*. Ed. by Stephen Cranefield et al. Lecture Notes in Computer Science. Springer, 2011, pp. 93–109. DOI: 10.1007/978-3-642-35545-5_6.

[23] Henry Prakken. ''On the problem of making autonomous vehicles conform to traffic law''. In: *Artif. Intell. Law* 25.3 (2017), pp. 341–363.

[24] Astrid Rakow and Maike Schwammberger. ''Brake or Drive: On the Relation Between Morality and Traffic Rules when Driving Autonomously''. de. In: *Software Engineering 2023 Workshops*. 2023, p. 104. DOI: 10.18420/se2023-ws-12.

[25] Galileo Sartor et al. ''Mind the Gap - The Rules of the Road for Humans and Machines''. In: *Proceedings of the 18th International Workshop on Juris-Informatics (JURISIN 2024)*. May 2024, pp. 55–70. ISBN: 978-4-915905-96-4.

[26] D. A. Schlobohm and D. A. Waterman. ''Explanation for an Expert System that Performs Estate Planning''. In: *ICAIL-1987*. ACM, 1987, pp. 18–27.

[27] Tran Cao Son et al. ''Answer Set Planning: A Survey''. In: *Theory Pract. Log. Program.* 23.1 (2023), pp. 226–298. DOI: 10.1017/S1471068422000072.

[28] Hui Wei et al. ''PlanGenLLMs: A Modern Survey of LLM Planning Capabilities''. In: *CoRR* abs/2502.11221 (2025). DOI: 10.48550/ARXIV.2502.11221. arXiv: 2502.11221.

Towards Translating Natural Language Normative Text into a Digital Twin of Administrative Law

Florian Schnitzhofer[0009−0005−3338−0366] and
Christoph G. Schuetz[0000−0002−0955−8647]

Institute of Business Informatics – Data & Knowledge Engineering, Johannes Kepler University Linz, Altenberger Str. 69, 4040 Linz, Austria
{schnitzhofer,schuetz}@dke.uni-linz.ac.at
https://www.dke.uni-linz.ac.at/

Abstract. Automating public-sector decision-making promises efficiency gains in administration. This short paper proposes a research agenda for translating natural-language normative text into a Digital Twin of Administrative Law (DTAL), which we envision as a layered, executable representation of statutes that preserves traceability to the authoritative legal text while enabling accountable automation of decision-making in the public sector. To obtain a DTAL from normative text, a stepwise translation pipeline must be followed, which we demonstrated on the Upper Austrian Tourism Contribution Levy Act. The resulting DTAL yields deterministic and explainable outcomes. In this short paper, we outline open questions on standardizing legal ontologies, integrating LLMs as assistant tools, and embedding DTALs into public-sector engineering practices to realize transparent auditable automation of decision-making aligned with the rule of law.

Keywords: Legal ontologies · law as code · digital twin of legislation · automated decision-making · administrative law

1 Introduction

Public administrations explore the use of *automated decision-making* (ADM) to improve the efficiency and consistency of tasks in areas such as taxation, social welfare, licensing, and law enforcement. However, algorithmic decisions raise rule-of-law concerns: Transparency, due process, and accountability can be compromised if systems are opaque or inconsistently implemented. To align ADM with the required legal guaranties, systems must be explainable, auditable, and grounded in law [2].

In practice, government agencies and software vendors often re-code portions of legislation on a case-by-case basis into software for supporting administrative tasks. However, rather than relying on ad-hoc software implementations, we propose translating administrative laws into structured, executable representations [5]—essentially a *Digital Twin of Law*. A Digital Twin of Law is a

synchronized digital counterpart of a legal text: Whenever the law is updated, the digital twin is updated accordingly, and can be used to automate decisions or provide explanations in a legally faithful manner.

This short paper, building on our previous work [12] that introduces the concept of a *Digital Twin for Administrative Law* (DTAL), focuses on the translation process from natural-language normative text to formalized DTAL representation. In this regard, a central research question is the following: *How can natural-language normative text be systematically translated into a Digital Twin of Administrative Law to support accountable automated decision-making?* We already identified necessary steps in a pipeline for transforming legislative text into a DTAL, derived from iterative design cycles, a grounded-theory analysis of expert insights, and a literature review. We already conducted a first demonstration of the approach using the *Upper Austrian Tourism Contribution Levy Act* as a use case, highlighting how the proposed pipeline works in practice. Nevertheless, key design decisions regarding the proposed pipeline remain open. In this paper, we describe some of these open decisions and propose next steps for research towards a systematic translation pipeline for building a DTAL from natural-language normative text.

The remainder of this paper is structured as follows. Section 2 describes relevant background on the legal formalization process and digital twins. Section 3 outlines the proposed research methodology based on the echeloned Design Science Research framework. Section 4 presents the DTAL translation pipeline, describes the tourism contribution levy use case and reports our research agenda. Section 5 concludes the paper.

2 Background

Encoding legal rules in machine-executable form has a long tradition in the *AI and Law* research community. Multiple streams of research inform our approach. Early systems formalized statutes and regulations as logical rules to derive transparent outcomes [8]. Satoh et al. have explored the translation of complex legal theories into Prolog code to build the PROLEG legal reasoning system [9]. More recently, researchers have been investigating how large language models (LLMs) can assist in the formalization process [15,10,4]. Legal ontologies capture core entities, relations, and constraints, providing semantic clarity and allowing for reuse across applications [14].

The concept of *digital twin*, established in engineering as a synchronized virtual counterpart of a real-world physical system [3], can be adapted to legislation to keep computable models aligned with promulgated texts [5]. For administrative law, which is often formulaic and data-driven, we adapt the digital-twin concept through a layered architecture of a *Digital Twin of Administrative Law* [12]. The combination of existing but distinct approaches (such as law-as-code [9], computable legislation [8,1], ontologies [14], and parameterization [5]), together with a systematic investigation of the translation process itself, constitutes the path we seek to investigate further.

3 Research Methodology

To develop our translation approach, our research follows the *echeloned Design Science Research* (eDSR) framework [13] to iteratively develop and evaluate the DTAL translation pipeline. Therefore, our research project is structured into multiple build-and-evaluate cycles (*design echelons*) to manage the complexity of translating legal texts into code. We first identified the problem and research objectives through literature review and exploratory interviews with domain experts, including legislative drafters, legal practitioners, and e-government specialists. Literature and interviews confirmed the need for automation of translation that remains faithful to the law, which then led to a formulation of key requirements for the DTAL construction process: Partial transformation of functionality, semantic accuracy, adapting drafting process to support ADM, and adaptability to legal changes.

Using these requirements, we designed an initial DTAL architecture and translation pipeline, which we prototyped for a specific use case: the *Upper Austrian Tourism Contribution Levy*. Through an iterative build-evaluate process, we converged on a generalized DTAL translation pipeline and a reference architecture, which we present in the next section.

4 From Normative Text to a Digital Twin

In this section, we describe a step-wise pipeline for translating natural-language law into a Digital Twin of Administrative Law. We first describe the general DTAL architecture that underpins the proposed approach. We then walk through the steps of the translation pipeline, from selecting the legal scope to evaluating the obtained DTAL.

4.1 Structure of a Digital Twin for Administrative Law

A core premise of our approach is to preserve a tight coupling between the legal text and the executable model. To this end, we structure the DTAL into the following four linked layers.

1. **Statutory text:** The promulgated wording remains the authoritative source, and all artifacts reference the source for provenance.
2. **Ontology:** The ontology serves as a shared vocabulary of legal concepts and relations, disambiguating terms and anchoring inputs/outputs of rules.
3. **Configuration:** Jurisdiction-specific parameters (e.g., rates, thresholds, categorical thresholds, lists of qualifying entities, and references to external standards) are maintained as data, enabling non-invasive updates.
4. **Executable logic:** Deterministic rules implement applicability, calculations, and decisions, returning decisions and computation results as well as explanations (logging of decision parameter and calculation pathways) tracing back to the legal sources.

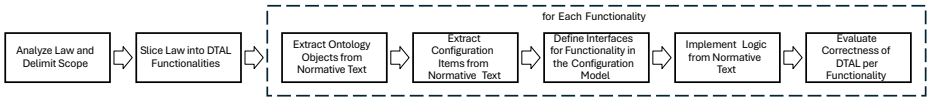


Fig. 1. Administrative law to DTAL pipeline steps

The logic consumes ontology terms and configuration values, while every rule cites the originating provisions. Exposed via a simple API and the Model Context Protocol (MCP), the DTAL becomes a runtime service that LLM-based agents can query for reproducible, computation-supporting, explainable, and auditable ADM without completely replacing human judgment for some decisions.

4.2 Translation Pipeline

Given the above architecture, we propose an iterative pipeline for translating a natural-language normative text into a DTAL, which is illustrated in Fig. 1. In the following, we briefly describe the steps of the translation pipeline.

1. **Analyze Law and Delimit Scope.** We begin by selecting the target normative text and determining which parts of the text are in scope for automation. An ongoing grounded-theory study with expert interviews shows that often, normative texts in administrative law contain some sections that are well-suited to automation and others that are not.
2. **Slice Law into DTAL Functionalities.** Next, we decompose the law's in-scope portions into discrete functional modules. Each functionality corresponds to a coherent decision[11] or computation that the law requires. Each functionality will be implemented and evaluated end-to-end, enabling modular development and targeted review.
3. **Extract Ontology Objects from Normative Text.** We identify and name the core legal concepts, roles, and relations that recur in the text (e.g., `ContributionRate`, `Municipality`, `LevyRate`) to create the ontology layer, capture them in a formal ontology (e.g., OWL class/property schema) with subclass hierarchies, and formulate constraints to disambiguate terms and standardize inputs/outputs across functionalities. To explore the possibility of the reuse of standardized legal ontologies, we currently conduct a systematic literature review following the mapping-study methodology proposed by Oliveira Rodrigues et al. [7], covering the period of 2017–2025.
4. **Extract Configuration Items from Normative Text.** We scan the law for any explicit values, external references, or parameters that should be treated as configuration items. We store configuration items in a separate configuration layer so that policy changes (e.g., updating a rate or threshold) are data updates rather than logic or ontology changes, preserving stability of the executable rules and the ontology model.
5. **Define Interfaces for Each Functionality in the Configuration Model.** For each digitally provided functionality we specify the external interface,

i.e., its required inputs, outputs, and validation constraints, as part of the configuration layer. This API-centric approach aligns the design with the principle of service-oriented architecture, i.e., each legal function can be seen as a microservice accessible via an API call.

6. **Implement Logic from Normative Text.** In the logic layer, we translate operative provisions, i.e., conditions, obligations, calculations, and decision rules, into executable, deterministic program code. For our use case, we used a high-level programming approach (Python), emphasizing readability and auditability in line with *Better Rules* principles [1]. For this current study, we conducted the extraction of the information from the natural language text manually even though this remains costly. The work by Zin et al. [15] and colleagues addresses data scarcity in legal information extraction with modern methods. LLMs can assist in moving from text to intermediate structures when coupled with retrieval or expert curation [10,4]. Future work will adopt a conservative stance: LLMs support drafting and hypothesis generation, while validation remains formal and/or expert-driven.
7. **Evaluate Correctness of DTAL per Functionality.** Once a functionality is implemented, it must be rigorously evaluated before it can be considered a faithful digital twin of the law. Whether this evaluation should be done via expert peer review, formal proofs, or extensive simulation (or a combination) is an open area for exploration. For this study we performed verification using scenario-based testing with expert-provided ground truth.

Executing these stages yields a layered DTAL (text, ontology, configuration, logic) whose rules cite their legal sources and whose parameters are externally governed. The steps are repeatable whenever the law changes (keeping the digital twin current) and transferable across statutes, enabling a growing repository of reusable, auditable legal models.

4.3 Use Case: Upper Austrian Tourism Contribution Levy

To demonstrate the proposed approach, we applied the DTAL translation pipeline to a real statute—the Upper Austrian Tourism Act 2018 [6]—focusing on the provision on the tourism contribution levy. This law is a suitable case study because it combines numeric calculations (the levy formula) with categorical rules (exemptions and applicability conditions), mirroring common ADM scenarios. Following our approach, we restricted the scope to the calculation of the tax, mapped key definitions and cross-references into the ontology and configuration layers, and manually implemented the logic of the law, i.e., applicability conditions, exemptions, and rate calculations, as Python functions, annotated with links to the corresponding legal provisions.

For testing the correctness and robustness, we assembled 100 realistic test scenarios reflecting diverse cases with different types of businesses, revenue levels, and municipalities. Each scenario's expected outcome was determined as ground truth by a legal tax expert. We then queried the DTAL service via its API for

each scenario and compared the results to the ground truth. The DTAL produced correct and consistent outcomes across all scenarios for this use case[1].

In contrast, a state-of-the-art LLM—we used GPT-5 Pro with deep search as an example—given the same task showed mixed performance. The LLM's answers were incomplete or incorrect on several edge cases and its outputs varied across multiple runs of the same query, indicating a lack of determinism. For example, the LLM often overlooked a specific exemption for certain business types. While the LLM produced executable code, the method of producing the code only allowed for weak traceability between the underlying legal norms and the generated implementation.

Throughout the work on the use case, we encountered several unresolved design decisions that highlight opportunities for future research. One opportunity relates to the question of how to design and conduct the evaluation process of the DTAL. Another open question is which legal ontology format should be adopted and whether it ought to be standardized. Although the case study is a relatively formulaic tax law, future work should apply the DTAL pipeline to other types of administrative law, including those with more textual or discretionary provisions, to evaluate the generalizability of the DTAL concept. We anticipate that certain legal norms, e.g., open-textured concepts or broad discretionary clauses, may not fully translate into code. Identifying how to handle such cases, e.g., via human-in-the-loop or hybrid approaches, and ensuring traceability of the decision-making process within the DTAL is an important area for further research. Finally, the application of LLMs to extract information from natural language law and therefore semi-automate this pipeline will also be an important focus of future work.

5 Conclusion

In this work, we presented an approach for bridging the gap between natural language normative text and automated decision-making systems through the concept of a *Digital Twin of Administrative Law*. We developed a layered DTAL architecture and an accompanying methodology to systematically translate normative legal content into a formal, executable model. We demonstrated the approach on a practical use case and showed that the resulting DTAL outperforms a purely LLM-based approach in terms of accuracy, consistency, and explainability[2].

By continuing to refine the proposed approach and embedding it in real-world legislative and administrative processes, we move closer to a future where digital government systems are as trustworthy and understandable as the legal texts that govern them.

[1] Scenario testing results are available online: https://doi.org/10.5281/zenodo.17571983.

[2] The DTAL use case implementation is open source: `https://github.com/FlorianSchnitzhofer/digital-twins-administrative-law-tourism-levy`.

References

1. Barraclough, T., Fraser, H., Barnes, C.: Legislation as code for New Zealand. Report: https://nzlii.org/nz/journals/NZLFRRp/2021/3.pdf (2021)
2. European Union: Regulation (eu) 2024/1689 of 13 june 2024 laying down harmonised rules on artificial intelligence (artificial intelligence act). Official Journal of the European Union, L Series (12 July 2024). ELI: http://data.europa.eu/eli/reg/2024/1689/oj (2024)
3. Grieves, M.: Digital twin: Manufacturing excellence through virtual factory replication. Tech. rep., Florida Institute of Technology (2014), White Paper
4. Janatian, S., Westermann, H., Tan, J., Savelka, J., Benyekhlef, K.: From Text to Structure: Using Large Language Models to Support the Development of Legal Expert Systems, pp. 167–176. IOS Press (12 2023). https://doi.org/10.3233/FAIA230962
5. Lamprecht, A.: Digital twins of law: Embracing complexity. In: Law for Professionals, pp. 115–135. Springer (2025). https://doi.org/10.1007/978-3-031-78596-2_6
6. Land Oberösterreich: Oö. Tourismusgesetz 2018 (LGBl. Nr. 3/2018). https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=LrOO&Gesetzesnummer=20000953 (2018), retrieved Feb 1, 2025
7. de Oliveira Rodrigues, C.M., de Freitas, F.L.G., Barreiros, E.F.S., de Azevedo, R.R., de Almeida Filho, A.T.: Legal ontologies over time: A systematic mapping study. Expert Systems with Applications **130**, 12–30 (2019). https://doi.org/https://doi.org/10.1016/j.eswa.2019.04.009
8. Prakken, H., Sartor, G.: Logical models of legal argumentation. In: Legal Knowledge Based Systems: JURIX 1997, pp. 23–33. IOS Press (1997)
9. Satoh, K., Asai, K., Kogawa, T., Kubota, M., Nakamura, M., Nishigai, Y., Shirakawa, K., Takano, C.: PROLEG: An implementation of the presupposed ultimate fact theory of japanese civil code by PROLOG technology. In: Onada, T., Bekki, D., McCready, E. (eds.) New Frontiers in Artificial Intelligence. pp. 153–164. Springer (2011). https://doi.org/10.1007/978-3-642-25655-4_14
10. Savelka, J., Ashley, K.D., Gray, M.A., Westermann, H., Xu, H.: Explaining legal concepts with augmented large language models (GPT-4). arXiv:2306.09525 (2023). https://doi.org/10.48550/arXiv.2306.09525
11. Schnitzhofer, F., Pils, P., Seper-Ambros, P.: Der selbstfahrende Staat. Springer Gabler (2024)
12. Schnitzhofer, F., Schütz, C.: Towards a scalable architecture for legal-ontologies integrated into digital twins of administrative law. In: SEMANTICS 2025 Developers Workshop (2025), https://ceur-ws.org/Vol-4064/SEMDEV-paper5.pdf
13. Tuunanen, T., Winter, R., vom Brocke, J.: Dealing with complexity in design science research: A methodology using design echelons. MIS Quarterly **48**(2), 427–458 (2024). https://doi.org/10.25300/MISQ/2023/16700
14. Valente, A.: Types and roles of legal ontologies. In: Law and the Semantic Web: Legal ontologies, methodologies, legal information retrieval, and applications, pp. 65–76. Springer (2005). https://doi.org/10.1007/978-3-540-32253-5_5
15. Zin, M.M., Nguyen, H.T., Satoh, K., Nishino, F.: Addressing annotated data scarcity in legal information extraction. In: Suzumura, T., Bono, M. (eds.) New Frontiers in Artificial Intelligence. pp. 77–92. Springer (2024). https://doi.org/10.1007/978-981-97-3076-6_6

Testing Modelling Fitness of Normative Specification Languages for LLMs

Giovanni Sileno[1] and Andrea Marino[1]

Informatics Institute, University of Amsterdam, The Netherlands
{g.sileno, a.marino}@uva.nl

Abstract. Several works propose to leverage LLMs to facilitate the translation from natural language into formal notations. The number of distinct formal notations proposed in the literature for normative specifications calls for dedicated comparative evaluations. In this paper, we elaborate on the notion of "modelling fitness" as a way to assess a given notation, which depends on the modeller's interpretation capacity and proficiency with the notation. We draft a methodology to experimentally test dimensions of modelling fitness for LLMs, introducing a "mirroring" pipeline for notations centred primarily on informational models, and reporting the results of preliminary experiments.

Keywords: Normative specifications · Representation of Norms · Modelling fitness · LLMs · Natural language · Formal notations

1 Problem setting

The number of domain-specific languages (DSLs), programming languages, and informational models for normative specifications continues to grow in the literature.[1] Comparing these proposals, and, even more, formally evaluating their expressivity, is a challenging task.[2] Yet, all these solutions share a fundamental common ground: normative requirements originate from human sources and as such they are initially formulated in natural language.

With the rise of initiatives aimed at integrating normative specifications into LLM-enhanced modelling pipelines, a significant, complementary question has to be addressed: *Which formal notation provides the most appropriate mapping for natural language sources?* For this aim, we introduce the notion of "modelling fitness". Modelling fitness describes the accessibility of a notation for a given modelling task, assessed relatively to the conceptual model held by the modeller (intertwining lexical, syntactic and semantic dimensions). Operationalizing modelling fitness for humans is difficult because its expression heavily depends on

[1] Focusing only on norm/contract representation frameworks with a stronger computational orientation represented in the last decade, we have: Defeasible Deontic Logic/SPINdle [11, 7], LegalRuleML [16, 12], NPL/NPL(s) [8, 25], PROLEG [19], InstAL [15], ODRL [9, 5], Symboleo [20], FLINT/eFLINT [23, 22], Logical English [10], Catala [13], Blawx [14], Stipula [4], and DCPL [21].

[2] For a wider socio-technical view see [1], in the main JURIX conference.

the modeller's background. Consider an analogy with programming languages: a Java programmer finds Scala or Kotlin much more familiar than Rust, while a C programmer experiences the opposite.[3] Similarly, an Italian speaker would find French easier to learn than Dutch, but the reverse would be true for a German speaker. Extending these examples from language learning to modelling tasks, we find that modelling fitness primarily depends on two components:

(i) *the capacity of forming a correct interpretation of the scenario at stake*, relying on the modeller's natural language proficiency and domain knowledge;
(ii) *the proficiency with the modelling notation*, depending on how easily the notation can be leveraged by the modeller's existing conceptual resources.

In the case of LLMs performing the modelling task, their "conceptual model" results from the texts used training texts. We formed a few working hypotheses of the performance of LLMs with respect to modelling fitness. The model's interpretation accuracy is highest for common scenarios (i). The accuracy of the output in the target modelling notation is enhanced if the formal language (ii-a) and its transportation layer (ii-b) are more commonly present in the training data, and if the notation allows for a more direct mapping from word to concept (ii-c). The present paper does not aim to provide exhaustive benchmarking, nor to validate these hypotheses. Instead, its purpose is to draft a methodology for starting testing them and to offer a reflection on an initial experimental iteration.

2 Methodology

Assessing a translation The most common way in the literature (see e.g. [17]) to assess the accuracy of a translation from natural to formal language (or symbolic specifications broadly) is through question-answering (Q/A) relative to a given scenario (S). In what we will call the ***reasoning pipeline***, both the question (Q) and the scenario (S) must first be translated into a formally equivalent specification. A reasoning engine or solver then provides the final answer (A). The simplest type of questions are those that require a Yes or No answer. For LLM-enhanced modelling processes, directly querying the LLM with the question (Q) establishes a baseline performance against which to measure whether the symbolic pipeline actually improves accuracy. Note that a self-refinement step may be necessary to correct syntactic errors that would prevent execution.

Since some of the target notations we are considering are primarily informational models (e.g. ODRL [9, 5], DCPL [21]), they cannot be directly connected to a reasoning engine. Consequently, these notations cannot be used with a reasoning pipeline. We introduced therefore an alternative method, inspired by the auto-encoder architecture, that we will call the ***mirroring pipeline*** (Fig. 1). This pipeline involves translating the natural language for S and Q into a formal specification, and subsequently translating it back to natural language. The core

[3] Since these are general-purpose languages, they can still create functionally equivalent programs; therefore, the issue here is not expressiveness in absolute terms.

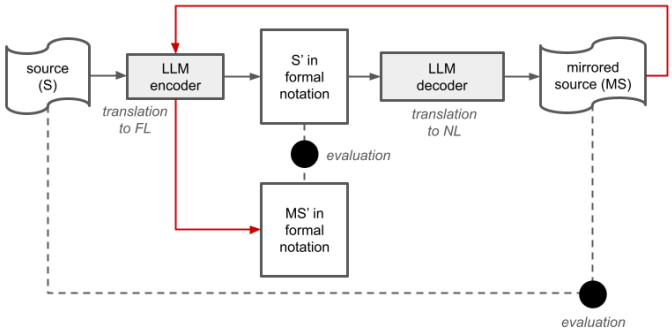


Fig. 1. The "mirroring" experimental pipeline, evaluating the system both on natural and formal language artefacts, by reconstructing the original source.

test is whether the initial and final natural linguistic artefacts maintain semantic equivalence, indicating that no significant informational loss occurred during the process. However, because the textual output may be different from the initial source, an additional step is needed to further automate the semantic equivalence test. A traditional way is to use some technique based on sentence embeddings; yet, the selection of proper embeddings adds complexity. A complementary way is to pass the mirrored text through the translation process again. The more the formal notation acts as a normalized representation, the more we should expect the formal representation to be equivalent, if the translation step is correct.

Translating via LLMs In both "reasoning" and "mirroring" pipelines, we need to settle upon a reproducible method for prompting the LLMs to perform the translation step from natural to formal language. However, as some of the target notations may not be present in the training dataset used to build the LLM, it is necessary to provide adequate relevant information about the notation syntax and semantics in the prompt. The syntax of DSLs and formal notations is generally described by BNF specifications, whereas the structure of informational models and knowledge-centred representational artifacts is described through frames, schemas, or ontologies. Some contribution may have both. These inputs are given to the LLM with the prompt. Note that the lexical choice of keywords of the language, as well as the transportation form (XML, JSON, RDF, ...) may influence the response of the LLM. Furthermore, to expose the LLMs to the notation semantics, we consider including representative examples (*few-shot learning*), possibly formal axioms, or plain-text descriptions of the language's core concepts and relationships directly within the prompt's context. A priori, it is unclear which of these components may play a more relevant role in improving the translation performance. We hypothesize that BNFs will likely improve syntactic forms of alignment [24], whereas use cases of the target language with their explanation in natural language will likely improve semantic alignment. Knowledge-centred notations, if expressed in commonly used means such as JSON, would likely already improve with just the examples.

3 Experimental setup

Testbench For our longer term experiments, we aim to collect a wide range of scenarios relevant for normative reasoning. These scenarios need to be chosen for *plausibility* (scenarios informed by statutory law, private law, contracts, agreements, technical regulations, and technical policies, including access and usage control), as well as for *specificity* (normative primitives and minimalistic compositions). Instances for the second group will be selected from literature on normative systems, as well as by generating a synthetic dataset.

Selection of target notations Before proceeding into the benchmarking, target languages for normative specifications need to be scrutinized. By examining academic and technical documentation, we will look for several different elements: (i) (for the reasoning pipeline) a reasoning engine that allows querying; (ii) formal specifications of the language (such as BNF grammars, schemas); (iii) relevant examples for few-shot learning; (iv) (optional) additional information such as formal axioms or plain-text descriptions which may improve semantic alignment. These elements should be maintained in a public repository to ensure reproducibility and allow for iterative improvements.

Evaluation The reasoning pipeline greatly simplifies the evaluation by centering on Yes/No questions; however, more complex questions can also be imagined. The mirroring pipeline evaluation involves instead two natural language texts (the source scenario and the reconstructed scenario) and two formal notation texts (the encodings of the above, generated independently by the same LLM module). Natural language texts can be compared using text-edit distances (Hamming, Levenshtein) or by comparing their embeddings, applying language models dedicated to this task. Formal language texts, however, allow for more structured comparisons. Several proposals exist for instance for schema/ontology alignment and matching (e.g. [2]). These methods typically rely on a mixture of methods (e.g. [6]), including graph-based embeddings (e.g. [3]) constructed on training data independent of the target language. However, their opacity makes them a suboptimal choice to identifying where the translation fails. On the opposite end of the spectrum, there exists methods to compare programs by means of *abstract syntactic trees* (ASTs), although these may be vulnerable to functional equivalence of distinct language compositions. As in these preliminary experiments we focus simple normative directives, this should be less relevant.

To simplify, let us assume the symbolic output is a list of associative arrays or key-value mappings (which may be nested, similarly to Python dictionaries), resulting in a JSON-like computational object. Given two such outputs, four comparison measures can be defined:

(a) *matching*: the number of dicts in the two lists that can be mutually matched with an adequate score, normalized on the total number of elements;
(b) *structural*: the degree to which two dicts share the same keys, across the overall structure;

(c) *type-matching*: the degree to which two dicts share the same data type in positions where they are structurally identical;
(d) *content*: the degree to which two dicts share the exact same content (values) in positions where their value types are the same.

The product of these four measures provides a measure of *deep equality*.

Preliminary experiments For our first experiments, we have collected from literature examples of normative primitives as the ones given in [18], education material, and simple scenarios from online sources, summing up to 48 simple scenarios. We will focus only on ODRL[4] and DCPL [5], both presented primarily as informational models, and provided with a JSON-schema. Even if no dedicated solver is available, we can still apply the mirroring pipeline.

To have a better picture of how the two languages perform in both encode/decode directions, we consider four different variations for an ablation analysis: (i) the prompt contains the JSON schema of the target language; (ii) the prompt uses a few-shot approach to provide examples of symbolic formulation in the given language; (iii) the prompt contains both schemas and few-shot examples; (iv) the prompt contains neither of the two (baseline). We generate embeddings of both natural language texts and compare their semantic similarity (using `all-MiniLM-L6-v2` from HuggingFace). The formal artefacts are instead compared based on the measures (*matching, structural, type-matching, content*) defined above. We performed the experiments on several models, both remote (`gpt-5-mini`, `gpt-4.1`, `gpt-4.1-mini`) and local (`gpt-oss`). The code, scenarios, prompts, output data, and scripts of analysis are available online.[6]

4 Preliminary results

In Table 1, we report the average performance of the different models when both schemas and few-shot examples are provided in the input. Regarding the semantic score, the performance of DCPL is consistent across the four models, whereas ODRL shows a relevant drop with both `gpt-oss` and `gpt5-mini`. The opposite trend occurs in the comparison of the symbolic artifacts: ODRL is rather consistent in its performance, whereas DCPL exhibits both the best performance (with `gpt5-mini`) and the worst performance (with `gpt4.1-mini`). This suggests that modelling fitness does vary. In Table 2, we report the average increase in `gpt5-mini`'s performance relative to the baseline (no additional input). This increase is measured in the presence or absence of the schema of the notation (as syntactic knowledge support) and few-shot examples (as semantic knowledge support) in the prompt. Interestingly, for ODRL, any additional information decreases the semantic score. Exposure to the schema has a negative impact for both ODRL and DCPL, although the few-shot examples balance it out in the

[4] `https://www.w3.org/TR/odrl-model/`
[5] `https://github.com/uva-cci/DCPLschema`
[6] `https://github.com/uva-cci/nll2fr-2025-neurosymbolic`

Model	Language	Semantic score	(a)	(b)	(c)	(d)	a · b · c	a · b · c · d
gpt5-mini	DCPL	0.82 ± 0.12	*0.92 ± 0.25*	*0.76 ± 0.26*	*0.86 ± 0.27*	**0.83 ± 0.27**	*0.68 ± 0.27*	**0.60 ± 0.28**
	ODRL	0.64 ± 0.20	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	0.49 ± 0.35	**1.0 ± 0.0**	0.49 ± 0.35
gpt4.1	DCPL	**0.83 ± 0.14**	0.74 ± 0.44	0.62 ± 0.40	0.67 ± 0.42	0.67 ± 0.42	0.56 ± 0.38	0.50 ± 0.36
	ODRL	*0.75 ± 0.17*	0.97 ± 0.16	0.94 ± 0.18	0.97 ± 0.16	*0.51 ± 0.29*	0.94 ± 0.18	*0.49 ± 0.29*
gpt4.1-mini	DCPL	0.78 ± 0.13	0.53 ± 0.49	0.47 ± 0.43	0.50 ± 0.47	0.44 ± 0.44	0.41 ± 0.40	0.34 ± 0.37
	ODRL	0.72 ± 0.19	0.96 ± 0.20	0.95 ± 0.20	0.96 ± 0.20	0.44 ± 0.24	0.95 ± 0.20	0.44 ± 0.23
gpt-oss	DCPL	0.81 ± 0.14	0.69 ± 0.45	0.63 ± 0.42	0.65 ± 0.43	0.60 ± 0.44	0.56 ± 0.41	0.47 ± 0.38
	ODRL	0.58 ± 0.25	0.96 ± 0.20	0.96 ± 0.20	0.96 ± 0.20	0.36 ± 0.29	0.96 ± 0.20	0.36 ± 0.29

Table 1. Performance across different models on the mirroring experimental pipeline, aggregated over the 48 scenarios using both schema/syntax and few-shot examples. The semantic score is calculated between two natural language texts (S and MS in Figure 1). The values a, b, c, d are ratios (defined respectively on matching *items*, *structure*, *type*, and *content*—see Section 3) computed between two JSON artifacts (S' and MS' in Figure 1). The final column represents a measure of deep equality.

Syntax	Few shot	Semantic score	(a)	(b)	(c)	(d)	a · b · c	a · b · c · d
present	absent	-0.04 ± 0.11	0.22 ± 0.54	**0.23 ± 0.44**	**0.24 ± 0.52**	**0.30 ± 0.47**	**0.25 ± 0.41**	*0.30 ± 0.35*
absent	present	**0.05 ± 0.10**	-0.56 ± 0.59	-0.43 ± 0.51	-0.49 ± 0.57	-0.44 ± 0.53	-0.38 ± 0.45	-0.30 ± 0.40
present	present	0.05 ± 0.09	**0.23 ± 0.50**	0.22 ± 0.42	0.23 ± 0.46	0.28 ± 0.44	0.20 ± 0.34	0.23 ± 0.30
present	absent	-0.20 ± 0.16	0.08 ± 0.34	0.13 ± 0.34	0.08 ± 0.34	*0.35 ± 0.50*	0.13 ± 0.34	**0.41 ± 0.46**
absent	present	*-0.01 ± 0.19*	*0.10 ± 0.30*	0.14 ± 0.34	*0.10 ± 0.30*	0.14 ± 0.40	0.14 ± 0.34	0.18 ± 0.34
present	present	-0.06 ± 0.18	*0.10 ± 0.30*	*0.16 ± 0.34*	*0.10 ± 0.30*	0.27 ± 0.48	*0.16 ± 0.34*	0.32 ± 0.40

Table 2. Performance change of `gpt5-mini` on the mirroring experimental pipeline for different configurations (syntax and few-shot examples present or absent), compared to the baseline of no added input. Data is aggregated across the 48 scenarios.

case of DCPL, and less so for ODRL. Conversely, looking at the comparison measures between the symbolic artifacts, we see an opposite trend, particularly for DCPL. Exposure to syntax greatly improves the model's performance, whereas the presence of only few-shot examples greatly degrades it. The best deep equality scores for the two languages are found when only the schema is given in the input. In short, our data suggests the existence of a trade-off between formal consistency and domain coherence: best performance on deep equality coincides with the worst performance on the semantic score, and vice-versa.

References

1. Ali, S., Sileno, G., Van Engers, T.: A systematic approach to assess languages proposed for rules as code. JURIX 2025 (2025)
2. Aumueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and ontology matching with COMA++. ACM SIGMOD 2005 pp. 906–908 (2005)
3. Chen, J., Hu, P., Jimenez-Ruiz, E., Holter, O.M., Antonyrajah, D., Horrocks, I.: OWL2Vec*: embedding of owl ontologies. Machine Learning **110**(7) (2021)
4. Crafa, S., Laneve, C., Sartor, G.: Stipula: a domain specific language for legal contracts. Workshop on Prog. Languages and the Law (ProLaLa 2022) (2022)
5. De Vos, M., Kirrane, S., Padget, J., Satoh, K.: ODRL policy modelling and compliance checking. Int. Joint Conf. on Rules and Reasoning pp. 36–51 (2019)
6. Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I.F., Couto, F.M.: The agreementmakerlight ontology matching system. OTM pp. 527–541 (2013)

7. Governatori, G.: An asp implementation of defeasible deontic logic. KI - Künstliche Intelligenz **38**(1), 79–88 (Aug 2024)
8. Hübner, J.F., Boissier, O., Bordini, R.H.: A normative programming language for multi-agent organisations. Annals of Mathematics and AI **62**(1), 27–53 (2011)
9. Iannella, R., Villata, S.: ODRL information model 2.2. W3C Recomm. (2018)
10. Kowalski, R., Datoo, A.: Logical English meets legal English for swaps and derivatives. Artificial Intelligence and Law pp. 1–35 (2021)
11. Lam, H.P., Governatori, G.: The making of spindle. Rule Interchange and Applications pp. 315–322 (2009)
12. Lam, H.P., Hashmi, M.: Enabling reasoning with LegalRuleML. Theory and Practice of Logic Programming **19**(1), 1–26 (2019)
13. Merigoux, D., Chataing, N., Protzenko, J.: Catala: A programming language for the law. Proc. ACM Program. Lang. **5**(ICFP) (aug 2021)
14. Morris, J.: Blawx: Web-based user-friendly rules as code. Workshops co-located with (ICLP 2022) **3193** (2022)
15. Padget, J., Elakehal, E.E., Li, T., Vos, M.D.: InstAL: An Institutional Action Language, Law, Governance and Technology Series, vol. 30 (2016)
16. Palmirani, M., Governatori, G., Rotolo, A., Tabet, S., Boley, H., Paschke, A.: LegalRuleML: XML-based rules and norms. Rule-Based Modeling and Computing on the Semantic Web pp. 298–312 (2011)
17. Pan, L., Albalak, A., Wang, X., Wang, W.: Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. EMNLP 2023 pp. 3806–3824 (2023)
18. Sartor, G.: Fundamental legal concepts: A formal and teleological characterisation. Artificial Intelligence and Law **14**(1), 101–142 (2006)
19. Satoh, K., Asai, K., Kogawa, T., Kubota, M., Nakamura, M., Nishigai, Y., Shirakawa, K., Takano, C.: PROLEG: An implementation of the presupposed ultimate fact theory of Japanese civil code by PROLOG technology. New Frontiers in Artificial Intelligence pp. 153–164 (2011)
20. Sharifi, S., Parvizimosaed, A., Amyot, D., Logrippo, L., Mylopoulos, J.: Symboleo: Towards a specification language for legal contracts. 2020 IEEE 28th Int. Requirements Engineering Conf. (RE) pp. 364–369 (2020)
21. Sileno, G., van Binsbergen, T., Pascucci, M., van Engers, T.: DPCL: a language template for normative specifications. Workshop on Prog. Languages and the Law (ProLaLa 2022) (2022)
22. Van Binsbergen, L.T., Liu, L.C., van Doesburg, R., van Engers, T.: eFLINT: a domain-specific language for executable norm specifications. ACM SIGPLAN Generative Programming: Concepts and Experiences pp. 124–136 (2020)
23. Van Doesburg, R., Van Der Storm, T., Van Engers, T.: Calculemus: towards a formal language for the interpretation of normative systems. AI4J **1**, 73 (2016)
24. Wang, B., Wang, Z., Wang, X., Cao, Y., Saurous, R.A., Kim, Y.: Grammar prompting for domain-specific language generation with large language models (2023)
25. Yan, E., Nardin, L.G., Hübner, J.F., Boissier, O.: An agent-centric perspective on norm enforcement and sanctions. Coordination, Organizations, Institutions, Norms, and Ethics for Governance of Multi-Agent Systems XVII pp. 79–99 (2025)

Author Index