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.

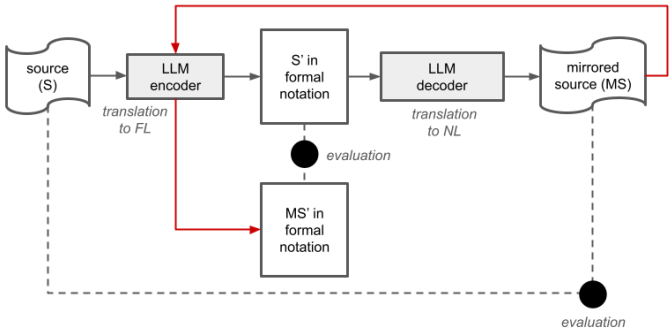

Fig. 1. The "mirroring" experimental pipeline, evaluating the system both on natural and formal language artefacts, by reconstructing the original source.

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)