

LawFOL: A Knowledge Representation Language for the Law

Robin DE VOGELAERE ^{a,b}, Bart BOGAERTS ^b Marc DENECKER ^b
Marjolein DERYCK ^b Joost VENNEKENS ^a

^a*AI Lab, Vrije Universiteit Brussel, Brussels, Belgium*

^b*Department of Computer Science, KU Leuven, Leuven, Belgium*

Abstract. This paper introduces a new legal knowledge representation formalism, based on first-order logic (FOL). Its goal is to follow the structure of the natural language legal text as closely as possible without having to rely on auxiliary symbols not found in the text. It improves on existing formalisms in this regard by incorporating various aspects typically found in legal language, specifically cross-references, hierarchical scopes and multiple types of exceptions. We evaluate the strength of our language by modelling a number of cases taken from EU legislation.

Keywords. legal reasoning, knowledge representation, non-monotonic logic, rule-based reasoning

Introduction

It is well known that legal language contains unique properties that make it hard to precisely model in a formal language. This paper introduces a new formalism for legal knowledge to address these challenges. Our aim is to stay as closely to the natural language text as possible. The advantages of such isomorphism are that it is easier to formalise legislation, verify the correctness of the resulting models, and update the knowledge base in case of legislative amendments [1]. It also enhances readability by domain experts, improving their trust in the model.

To accomplish this, we identify two main requirements that a legal knowledge representation language should meet:

- The formalism should follow the structure of the law, in the sense that information should (only) appear where it occurs in the natural language text, i.e. it should be possible to partition the resulting model using the article and subarticle numbering from the legal text.
- It should not be necessary to introduce auxiliary symbols to the knowledge base, which have no basis in the legal text and are added purely for technical reasons. For instance, expressing that a rule is defeasible, often requires the addition of a statement that no exception applies in formal languages, whereas this is implicitly understood in legal rules.

There are three important features that are typical of legal text and warrant a unique approach to modelling:

- *Exceptions*. Legal texts typically start from a main case and then carve out the situations where this default rule does not apply.
- *Cross-references*. Legal texts contain plenty of references to other rules to avoid repeating information and ensure consistency.
- *Hierarchies*. Information is often grouped in a hierarchical way. By capturing this structure, the exceptions and cross-reference mechanisms become more effective.

We propose a new representation language, called LawFOL, that natively incorporates these specificities of legal texts. We evaluate our formalism by modelling several real-world pieces of legislation, namely the GDPR [2], which deals with the processing of personal data and privacy, and the Rome I Regulation [3], which governs which law is applicable to cross-border contracts.

The paper starts with a discussion of key characteristics of legal texts in Section 1. After some formal preliminaries in Section 2, Section 3 then explores the main features and syntax of LawFOL. Next, Section 4 details its formal semantics. Section 5 analyses when legal acts modelled in LawFOL are acceptable. Section 6 provides practical examples of its use, highlighting some advantages. Finally, Section 7 discusses the related work, before the conclusion.

1. Key characteristics of legal text

This section introduces the key characteristics of legal text that inspire LawFOL.

Exceptions. We identify two distinct types of exceptions in legal texts: modifications and priorities.

In the case of *modifications*, a later article alters a preceding article, e.g. by adding or removing conditions, or by changing the outcome of applying the article. A typical example concerns Article 6(1)(a) GDPR, which states that consent by the data subject enables lawful processing of personal data. However, Article 8 modifies this rule, by stipulating that, for minors under the age of 16, authorisation from the parents is additionally required (in certain cases). LawFOL allows a faithful representation of this, in which the representation of Article 6 does not need to be adapted to allow for the subsequent modification by article 8 (see Section 6).

A special case of modifications are *cancellations*. Continuing the same example, Article 9(1) states that consent is never a valid ground for processing certain types of sensitive data. In other words, if this type of data is concerned, the normal application of Article 6(1)(a) is “cancelled”. From a logical perspective, this corresponds to a strengthening of the conditions of this article.

Chains of modifications should be applied backwards from the end of the chain. Again continuing the same example, Article 9(2) contains some exceptions to the prohibitions of Article 9(1). In other words, Article 9(2) sometimes cancels Article 9(1), which in turn cancels Article 6(1)(a). If the conditions for all three articles are met, then the Article 9(1) is cancelled by Article 9(2), and therefore Article 6(1)(a) is reinstated and will apply as usual.

In contrast to modifications, *priorities* usually deal with possible conflicts between independent rules. For instance, Article 5 Rome I contains the conflicts of laws rules for contracts of carriage, while Article 6 deals with consumer contracts. To a consumer

contract of carriage, both (potentially contradictory) rules seem to apply. To resolve this possible conflict, Rome I explicitly prioritises Article 5 over Article 6 (again, also see Section 6).

In LawFOL, priorities take the form of a fixed partial order between articles, i.e., we assume that modifications or cancellations do not change the priorities. Unlike cancellations, chains of priorities are transitive. For the Rome I example, if Article 5 overrides the general rules from Article 4, this implies that Article 6 also overrides Article 4. (as opposed to reinstating it as with cancellation). We say that the higher-priority article *overrules* the lower-priority articles.

Cross-references. In legal text, articles often refer to (parts of) other articles. LawFOL provides built-in mechanisms for referring to the conditions of other articles.

Hierarchies. Legal acts are layered, reflecting the general top-down structure of starting from the general case and moving to the specific. Conditions or exceptions may apply to an entire article, or only to a paragraph. Again, LawFOL faithfully reflects this structure in its formal syntax.

2. Formal Preliminaries

LawFOL builds on a typed variant of first-order logic (FOL). A *vocabulary* consists of a set of type symbols $\{T_1, \dots, T_n\}$, a set of predicate symbols $\{p_1, \dots, p_m\}$ and a set of function symbols $\{f_1, \dots, f_k\}$. Each symbol σ has an associated arity $n \in \mathbb{N}$, denoted σ/n ; a predicate with arity zero is a *propositional symbol* and a function with arity zero is a *constant*. Each predicate symbol p/n has an associated *signature*, which is an n -tuple of types. Each function symbol f/n also has a signature, which is an $(n+1)$ -tuple of types. *Terms*, *atoms* and *formulas* are defined as usual [4], with the additional constraint that they must all be well-typed. Quantifiers range over *typed variables* and we write $\forall x \in T$ or $\exists x \in T$ to indicate that variable x is of type T . A formula is in *Conjunctive Normal Form (CNF)* if it is a conjunction of clauses, where each clause is a disjunction of literals. All variables are implicitly universally quantified.

An *interpretation* S for a vocabulary Σ consists of a set of objects T_i^S for each type $T_i \in \Sigma$, and assigns to each other symbol $\sigma \in \Sigma$ an appropriate interpretation σ^S . If σ/n is a predicate with signature $(T_1, \dots, T_n)$, then $\sigma^S \subseteq T_1^S \times \dots \times T_n^S$. If σ/n is a function symbol with signature $(T_1, \dots, T_{n+1})$, then σ^S is a function symbol from $T_1^S \times \dots \times T_n^S$ to T_{n+1}^S . The value of a formula or term in an interpretation is defined as usual in typed first-order logic. Meaning, for instance, that a formula $\forall x \in T : \phi$ is true in S if $\phi[x/d]$ (the formula obtained from ϕ by replacing all free occurrences of x by d) holds for every $d \in T^S$. We write $S \models \phi$ if ϕ is true in S .

3. LawFOL Syntax

Normal articles are the basic building block of LawFOL. Such an article α is a triple $(\vec{x}, \phi, A)$ where $\vec{x}$ is a tuple of typed variables, ϕ a formula in CNF called the *conditions* of α , and A an atom called its *outcome*. All free variables in ϕ and A must be contained in $\vec{x}$. We write such a normal article in the following syntax, where the condition *true* can also be omitted:

```

1 Article  $\alpha$  {
2   variables:  $x_1 \in T_1, \dots, x_n \in T_n$ .
3   conditions:  $\phi$ .
4   outcome:  $A$ .
5 }

```

As discussed, legal texts often contain subsequent articles that modify earlier articles. To represent this, we introduce *modifying articles*, which are a quadruple $\alpha = (\vec{x}, \phi, \beta, \sigma)$. As before, $\vec{x}$ is a tuple of variables and ϕ the conditions. In addition, $\beta \neq \alpha$ is (the identifier of) the article being modified and σ is the actual modification itself. Formally, we require that σ is a β -modification, defined as follows.

For a normal article β , a β -modification is either an expression of the form $\beta.outcome := B$, with B an atom, or an expression of the form $\beta.conditions \sim \theta$ where θ is a formula and $\sim \in \{+=, -=, :=\}$. Intuitively, the $+=$ operator means that θ will be added to the original conditions of β , strengthening them, while $- =$ means that the condition θ will be removed from the conditions of β , weakening them, and $:=$ means that θ will replace the original conditions altogether. Whenever $\beta.conditions -= \theta$ is used, we assume that the CNF formula $\beta.conditions$ indeed contains the conjunct θ .

For a modifying article β , a β -modification can either be a modification of the conditions, as above, or an expression $\beta.modifies := \theta$. In the latter case, the original modification made by β will be replaced by a new modification θ . Here, we impose the restriction that θ must still affect the same article as the original $\beta.modifies$ did. In other words, whenever a new modification θ replaces the original modification σ of a modifying article $\beta = (\vec{x}, \phi, \gamma, \sigma)$, θ must still be a modification that affects article γ .

The variables of the modified and modifying articles do not need to be identical, but when the same variable appears in both, it is understood to refer to the same object.

As explained above, a modifying article can completely cancel an original article. We introduce an expression “cancels: α ” as syntactic sugar for “modifies: $\alpha.conditions += \text{false}$ ”.

To represent priorities, we allow an article α to include a *priority-expression* of the form: $\text{priority: } \sim \beta$, with $\beta \neq \alpha$ an article and $\sim \in \{<, >\}$, to indicate that α article has a lower, respectively higher, priority than β .

A *legal act* is a set of normal and/or modifying articles. To further enhance the correspondence to the structure of legal text, we now introduce two additional features.

3.1. Hierarchical Articles

A *hierarchical article* is an article that contains nested subarticles, which can either in turn be hierarchical articles, or normal or modifying articles. The hierarchical article itself can contain variables, priority relations and conditions, which are implicitly part of each of its subarticles. For instance, the article α_1 below has all of the x_i and y_j as its variables and its conditions are $\phi \wedge \psi$:

```

1 Article  $\alpha$  {
2   variables:  $x_1 \in T_1, \dots, x_n \in T_n$ .
3   conditions:  $\phi$ .
4   Article  $\alpha_1$  {
5     variables:  $y_1 \in U_1, \dots, y_m \in U_m$ .

```

```

6         conditions:  $\psi$ .
7         outcome:  $A$ .
8     }
9 }

```

3.2. Cross-references

The final element of LawFOL are cross-references. There are three possible types of references, which can be used in the conditions of articles as if they were simple atoms:

- A reference $\alpha.conditions(\vec{x})$ to the *original* conditions of an article α with variables $\vec{x}$, without considering any modifications to them.
- A reference $\alpha.scope(\vec{x})$ to the *modified* conditions of α . For instance, if the article α has been cancelled for a specific tuple of objects $\vec{o}$, then $\alpha.scope(\vec{o})$ will be *false* for $\vec{o}$, signifying that, when considering the entire legislation, $\vec{o}$ is outside the scope of this article.
- A reference $\alpha.applies(\vec{x})$ to whether α actually applies or not. It α applies to objects $\vec{o}$, its outcome must be the case for these $\vec{o}$. If α does not apply, this can be either because $\vec{o}$ is outside of its scope, or because it has been overruled by an article with *higher priority*.

While these three types of cross-references are important for matching the structure of legal text, to simplify the treatment of the formal semantics in the next section, we will first eliminate these cross-references by transforming the legal act.

If an article contains an expression $\alpha.conditions(\vec{x})$ and $\alpha = (\vec{x}, \phi, A)$, then we simply replace this expression by $\phi(\vec{x})$.

To handle the expression $\alpha.applies(\vec{x})$, with $\alpha = (\vec{x}, \phi, p(\vec{x}))$, we first introduce a new predicate symbol p_α and change α to have the outcome $p_\alpha(\vec{x})$ instead. We then add a new article $(\vec{x}, p_\alpha(\vec{x}), p(\vec{x}))$ to ensure that this new outcome always causes the original outcome as well. The purpose of this transformation is to have the symbol p_α which is only caused by the article α . Therefore, $p_\alpha(\vec{x})$ is true if and only if α actually applies for $\vec{x}$. We then replace $\alpha.applies(\vec{x})$ by $p_\alpha(\vec{x})$. For outcomes containing a function symbol, a similar transformation can be used.

For $\alpha.scope(\vec{x})$ we can use a similar transformation. Only, in this case, we move the priority relations from α to the new article $(\vec{x}, p_\alpha(\vec{x}), p(\vec{x}))$, so p_α remains unaffected by any priority rules that might have affected α before.

4. Formal Semantics

We now explain the formal semantics of LawFOL. For the remainder of this section, we assume a fixed legal act (i.e., set of articles). In addition, we also assume that we are given a partial order $\preceq$ on these legal articles, that represents their priorities.

In our framework, the purpose of a legal act is to define the value of some symbols that appear in it, in terms of other symbols. We define an *exogenous symbol* as one which appears only in the conditions of articles but not in outcomes. The remaining symbols are called *endogenous*. Note that an exogenous symbol can appear in a modification, but only in a modification of a condition.

Let us fix an interpretation J_0 for the exogenous symbols. In the context of J_0 , an article α with typed variables $x_1 \in T_1, \dots, x_n \in T_n$ applies to each appropriately typed tuple of objects $\vec{o} \in T_1^{J_0} \times \dots \times T_n^{J_0}$. We call the pair $(\alpha, \vec{o})$ an *instantiation* of α . For each such instantiation, we consider the article $\alpha[\vec{x}/\vec{o}]$ which is obtained by replacing each variable x_i of α by the corresponding object o_i . We call such an $\alpha[\vec{x}/\vec{o}]$ a *grounding* of α in J_0 . The *grounding* of the entire legal act $\mathcal{A}$ in J_0 is set of all possible groundings of articles $\alpha \in \mathcal{A}$ in J_0 . From now on, we continue to work with the grounding L of the legal act $\mathcal{A}$ in J_0 . Since their variables have been replaced, we identify such ground articles $\alpha \in L$ with the condition-outcome pair (ϕ, A) if they are normal and with the condition-article-modification triple (ϕ, β, σ) if they are modifying articles.

The central idea behind the LawFOL semantics is that, starting from the interpretation J_0 of the exogenous symbols, ground articles $\alpha \in L$ are successively applied to gradually build up the desired interpretation for the endogenous symbols. First, we extend the interpretation J_0 to an interpretation I_0 for the entire vocabulary. We do this by interpreting each endogenous predicate symbol p by $p^{I_0} = \emptyset$ and each endogenous function symbol by a partial function with an empty domain, so that $f^{I_0}(\vec{o})$ is undefined for each $\vec{o}$. The application process is formalised as a sequence of states of the following form:

Definition 1 A state is a pair (I, L) of an interpretation I and a ground legal act L .

Applying a normal article produces its outcome if its conditions are satisfied.

Definition 2 Let (I, L) be a state and let $\alpha \in L$ be a ground normal article of the form (ϕ, A) . The result of applying α in (I, L) , denoted as $(I, L)[\alpha]$ is defined as $(I \cup A, L')$ if $I \models \phi$ and (I, L') otherwise. In both cases, L' is $L \setminus \{\alpha\}$.

Here, the notation $I \cup A$ requires some additional explanation. If the outcome A is of the form $p(\vec{o})$, then $I' = I \cup A$ is the interpretation which is identical to I , apart from the fact that $p^{I'} = p^I \cup \{\vec{o}\}$. If A is of the form $f(\vec{o}) = o$, then $I' = I \cup A$ is again identical to I , apart from $f^{I'}(\vec{o}) = o$.

Applying a modifying article does not change our information about the state of the world I , but instead changes the set of ground articles L itself.

Definition 3 Let $\alpha = (\phi, \beta, \sigma)$ be a modifying article. The result of applying α to (I, L) , denoted again as $(I, L)[\alpha]$, is defined as (I, L') with L' determined as follows. If $I \not\models \phi$, then $L' = L \setminus \{\alpha\}$, as before. If $I \models \phi$, then $L' = L \setminus \{\alpha, \beta\} \cup \{\beta'\}$ where β' is the result of applying the modification σ to β , as explained before.

For the example of Article 8 modifying Article 6(1)(a) GDPR, this would mean that for a given data subject, we check if that person is under 16. If this is the case, we replace the grounded Article 6(1)(a) with respect to that person with a new version containing the additional condition of authorisation by a parent. If not, we retain the original version. In both cases, the modifying Article 8 with respect to that person has been applied, and can be removed from L .

The idea is now to iteratively apply articles, starting from the interpretation I_0 .

Definition 4 Let I_0 extend J_0 with default interpretations for the endogenous symbols, as defined above, and let (I_0, L) be a state. An application sequence is a sequences of states $(I_0, L), (I_1, L_1), \dots, (I_n, L_n)$ such that each $(I_{j+1}, L_{j+1}) = (I_j, L_j)[\alpha_j]$ for some $\alpha_j \in L_j$.

Obviously, articles cannot be applied in just any order. To be more precise about which orders are allowed, we first introduce the following concept.

Definition 5 A ground article α directly depends on another ground article β if:

- Article β is a normal article (ϕ, A) and the conditions of α contain A .
- Article β is a modifying article of the form $(\cdot, \alpha, \cdot)$.
- Article β has priority over article α , i.e., $\alpha \prec \beta$.

Article α (indirectly) depends on article β if either α directly depends on β , or there exists a third article γ such that α depends on γ and γ depends on β .

Now, we can define when an article can safely be applied.

Definition 6 We say that an article α is safe in state (I_i, L_i) if α does not depend on any article in $\beta \in L_i$. We call an application sequence safe if the article that is applied at each step in the sequence is safe.

The idea is now to keep applying safe articles, until no articles remain.

Definition 7 A terminal application sequence is a sequence that ends in $(I_n, \emptyset)$. We call I_n a solution and say that it is complete if, for each function symbol f , f^{I_n} is a total function, that assigns a value to all correctly typed tuples.

In the next section, we will show that, under certain conditions, such complete solutions are guaranteed to exist and, moreover, that the safeness condition ensures that they are unique.

5. Acceptable Legal Acts¹

A first consideration is that a non-terminal application sequence might get stuck if there are cyclic dependencies.

Definition 8 A legal act has a loop if an instantiated article (indirectly) depends on itself.

Note that this definition talks about the *ground* articles. It is possible for articles to refer to each other at the non-ground level, without introducing a loop at the ground level.

Lemma 1 For a finite loop-free legal act and a given interpretation of the exogenous symbols, in every state (I, L) , where $L \neq \emptyset$ there will always be at least one safe article $\alpha \in L$.

From a practical point this lemma is important, since legal acts are finite and, barring legislative mistakes, do not contain loops.

Lemma 2 If a ground legal act does not contain loops, it has at least one solution.

¹The proofs for this section, which follow mostly from the definitions in the previous section, can be found at <https://gitlab.com/EAVISE/rdv/lawfol>

We will now show that all terminal safe application sequences have the same final state, and that ground legal acts therefore have a *unique* solution.

Let $(I_i, L_i)_i$ be an application sequence. We say that an article α *changes* in step i of such a sequence if either the truth value of α 's conditions differs between I_i and I_{i+1} , or α 's conditions and/or outcome in L_i are different than in L_{i+1} , or it is overruled by the article applied in step i .

Lemma 3 *If article β changes in step i of an application sequence, then β depends on the article α that is applied in step i .*

From this result, we immediately obtain the following.

Lemma 4 *If an article α is safe in step i , then it will not change in any step $j > i$.*

Lemma 5 *Suppose that $S = (I_i, L_i)_{i=0}^n$ and $S' = (I'_j, L'_j)_{j=0}^m$ are two terminal application sequences. If an article α is applied in S with outcome o , then α is also applied in S' with the same outcome o .*

This lemma now immediately implies the following result, because the final state of an application sequence depends only on the applied articles and their outcomes.

Theorem 1 *All terminal application sequences have the same limit.*

We hypothesise that all “good” legislation consists of legal acts without loops which have a complete solution as they assign a default value to each function.

6. Examples

6.1. Modification in GDPR

In Section 1 we discussed how Article 8 GDPR adds a requirement to Article 6(1)(a) that the consent to process personal data given by a minor is authorised by a parent. The following simplified listing shows how modification in LawFOL can be used to add this condition to Article 6(1)(a), without needing to change anything to that article ².

```

1 Article 6(1)(a) {
2     variables: dp ∈ Data_Processing, c ∈ Consent.
3     conditions: valid(c).
4     outcome: lawful(dp).
5 }
6 Article 8 {
7     variables: dp ∈ Data_Processing, c ∈ Consent, p ∈ Person.
8     conditions: age(data_subject(dp)) < 16.
9     modifies: Article 6(1)(a).conditions
10    += authorised(p, c) ∧ parent(p, data_subject(dp)).
11 }
```

²The full example can be found at <https://gitlab.com/EAVISE/rdv/lawfol>

6.2. Rome I Regulation: Co-Occurrence of Exceptions and Cross-References

Rome I contains an interesting example of the interplay between references and different types of exceptions.

The base rules to determine the applicable law to a contract, in the absence of any express agreement, are laid out in Article 4. This is followed by special rules for certain types of contracts. To avoid that the same issue is governed by (possibly conflicting) articles, the Regulation states that article 4 has lower priority than the specific articles following it.

Article 6 on consumer contracts is one such specific article. Its fourth paragraph contains a number of exceptions. One such carve out is the following:

Article 6(4): Paragraphs 1 (...) shall not apply to: (...)

(e) a contract concluded within the type of system falling within the scope of Article 4(1)(h).

How should we deal with this seemingly contradiction between the priority relation and the quoted exception? First, we can simply introduce the exception from Article 6(4)(e) as a cancellation of 6(1), which corresponds to the literal text. Next, we realise that the priority between Articles 4 and 6 only relates to the parts of the article regarding the conflicts of laws contained in Articles 6(1), rather than the full Article 6. Once the priority has been pointed to the right position in the hierarchy, any cancellation of 6(1) automatically reinstates Article 4. What remains is to identify the right type of cross-reference to 4(1)(h) in the exception. Again, we can follow the literal text and use the `.scope`, resulting in the following listing:

```
1 Article 6(4)(e) {
2   variables: c ∈ Contract.
3   cancels: Article 6(1).
4   conditions: Article 4(1)(h).scope(c).}
```

The alternative references would not have worked. `.applies` would create a circularity with the priority, while `.conditions` would have lead to possibly unintended consequences if Article 4(1)(h) would have been cancelled in turn, as now neither that special rule nor the special rules from Articles 6(1) would have applied.

7. Related Work

Our work relates to the long-standing research on the application of (non-monotonic) logics to the law, including seminal works such as [5,6,7].

More recently, Catala [8] has emerged as a declarative language for tax law and social benefits legislation. Catala's priority rules, which were taken from defeasible logics, served as inspiration for LawFOL's priority mechanism.

A key difference between Catala and LawFOL is that the former is a functional programming language, which makes sense given the focus on more mathematical legislation. However, this choice does imply that the modeller still needs to take the control flow into account to some extent. By contrast, LawFOL is a purely declarative language adhering to the knowledge base paradigm [9].

[10] studied the relation between legal defeasible reasoning and logic programming with negation-as-failure. PROLEG [11] builds on this work by hiding the negation-as-failure in an exception mechanism. This deceptively simple mechanism is surprisingly versatile for modelling cancellations.

Finally, an important line of work in the area of AI and law are the argumentation-based approaches [12,13], which extend logic programming with argumentation frameworks. Within this line of research, they have encountered many of the same issues when formalising legislation that we have. For instance, they distinguish three types of exceptions, of which the first two correspond to cancellation from a declarative perspective, and the third is the priority relation.

Some argumentation-based formalisms also allow users to declare a hierarchy between legal acts, e.g. between the Rome I Regulation and national private international law. The rules contained in the prioritised act will take precedence in case of conflict. This is different from our hierarchical approach to conflicts in that an article with a lower priority will not apply at all.

[14] have also argued in favour of defeasible priorities. While potentially useful from an argumentation point-of-view, we have shown that with the two exception mechanisms and a reference mechanism, we are able to deal with complex issues.

Argumentation models use logical formalisms to represent legal rules and defeasible reasoning. LawFOL could be a suitable KR language for their needs.

8. Conclusions and Future Work

This paper introduced LawFOL, a KR formalism aimed specifically at legal applications. Its main focus is on the concept of an article, which can contain conditions and an outcome, and which can be referenced or modified from other articles. LawFOL is able to handle exceptions through such modifications or by prioritising articles. It is able to model the hierarchical structure of the law, making it possible to make statements about multiple legal rules at the same moment.

Currently, LawFOL allows only representations that consist of a single legal act (or related acts, such as an act and its implementing legislation). To lift this restriction, especially the priority mechanism might need to be adapted.

From a practical perspective, it would be interesting to find out how easy it is to translate to and from other legal and non-legal formalisms. One especially promising avenue in this respect would be to investigate a translation of LawFOL into FO($\cdot$) [15], the language supported by the IDP system [16], a knowledge base system for extensions of first-order logic. We believe this should be feasible since our semantics of legal acts (in particular the notion of terminal application sequences), closely mimics the induction process of the well-founded semantics [17], which is the semantics used in FO($\cdot$) for formalising inductive definitions. Such a translation into input for a knowledge base system [9] such as IDP would have as advantage that it immediately enables reasoning about legal acts in many different ways, e.g., using automated theorem proving [18] to check consistency of legislation, using model expansion [19] to generate situations that lead to a desired outcome, using explanation inference [20] for explaining decisions to citizens, or using interactive configuration methods [21,22] for providing decision support.

A final direction of potential future research include is to enrich our language with temporal or deontic statements.

Acknowledgements

The authors would like to sincerely thank Prof. Satoh for interesting discussions that inspired this work. This work was partially made possible thanks to a FWO travel grant K209626N. This research received funding from the Flemish Government under the “Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” programme.

References

- [1] Bench-Capon TJM, Coenen FP. Isomorphism and legal knowledge based systems. *Artificial Intelligence and Law*. 1992;1:65-86. Available from: <https://doi.org/10.1007/BF00118479>.
- [2] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance); 2016.
- [3] Regulation (EC) No 593/2008 of the European Parliament and of the Council of 17 June 2008 on the law applicable to contractual obligations (Rome I); 2008.
- [4] Enderton HB. *A Mathematical Introduction To Logic*. 2nd ed. Academic Press; 2001.
- [5] Allen LE. Symbolic Logic: A Razor-Edged Tool for Drafting and Interpreting Legal Documents. *The Yale Law Journal*. 1957;66(6):833-79.
- [6] Sergot MJ, Sadri F, Kowalski RA, Kriwaczek F, Hammond P, Cory HT. The British Nationality Act as a logic program. *Communications of the ACM*. 1986;29(5):370-86.
- [7] Kowalski RA. Legislation as logic programs. In: *Informatics and the Foundations of Legal Reasoning*. Springer; 1995. p. 325-56.
- [8] Merigoux D, Chataing N, Protzenko J. Catala: a programming language for the law. *Proceedings of the ACM on Programming Languages*. 2021;5(ICFP):1-29.
- [9] Denecker M, Vennekens J. Building a Knowledge Base System for an Integration of Logic Programming and Classical Logic. In: Garcia de la Banda M, Pontelli E, editors. *Logic Programming*. vol. 5366. Berlin, Heidelberg: Springer Berlin Heidelberg; 2008. p. 71-6. Available from: http://link.springer.com/10.1007/978-3-540-89982-2_12.
- [10] Satoh K, Kubota M, Nishigai Y, Takano C. Translating the Japanese presupposed ultimate fact theory into logic programming. In: *Legal Knowledge and Information Systems*. IOS Press; 2009. p. 162-71.
- [11] Satoh K, Asai K, Kogawa T, Kubota M, Nakamura M, Nishigai Y, et al. PROLEG: an implementation of the presupposed ultimate fact theory of Japanese civil code by PROLOG technology. In: *JSAI international symposium on artificial intelligence*. Springer; 2010. p. 153-64.
- [12] Prakken H, Sartor G. The role of logic in computational models of legal argument: a critical survey. *Computational logic: Logic programming and beyond: Essays in honour of robert A Kowalski part II*. 2002;342-81.
- [13] Prakken H, Sartor G. Law and logic: A review from an argumentation perspective. *Artificial intelligence*. 2015;227:214-45.
- [14] Prakken H, Sartor G. Argument-based extended logic programming with defeasible priorities. *Journal of applied non-classical logics*. 1997;7(1-2):25-75.
- [15] Denecker M, Ternovska E. A logic of nonmonotone inductive definitions. *ACM Trans Comput Log*. 2008;9(2):14:1-14:52. Available from: <https://doi.org/10.1145/1342991.1342998>.
- [16] De Cat B, Bogaerts B, Bruynooghe M, Janssens G, Denecker M. Predicate logic as a modeling language: the IDP system. In: Kifer M, Liu YA, editors. *Declarative Logic Programming: Theory, Systems, and Applications*. ACM / Morgan & Claypool; 2018. p. 279-323. Available from: <https://doi.org/10.1145/3191315.3191321>.
- [17] Denecker M, Vennekens J. Well-Founded Semantics and the Algebraic Theory of Non-monotone Inductive Definitions. In: Baral C, Brewka G, Schlipf JS, editors. *Logic Programming and Nonmonotonic Reasoning*. 9th International Conference, LPNMR 2007, Tempe, AZ, USA, May 15-17, 2007, Proceedings. vol. 4483 of Lecture Notes in Computer Science. Springer; 2007. p. 84-96. Available from: https://doi.org/10.1007/978-3-540-72200-7_9.

- [18] Fitting M. First-Order Logic and Automated Theorem Proving, Second Edition. Graduate Texts in Computer Science. Springer; 1996. Available from: <https://doi.org/10.1007/978-1-4612-2360-3>.
- [19] Mitchell DG, Ternovska E. A Framework for Representing and Solving NP Search Problems. In: Veloso MM, Kambhampati S, editors. Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA. AAAI Press / The MIT Press; 2005. p. 430-5. Available from: <http://www.aaai.org/Library/AAAI/2005/aaai05-068.php>.
- [20] Bogaerts B, Gamba E, Guns T. A framework for step-wise explaining how to solve constraint satisfaction problems. *Artif Intell.* 2021;300:103550. Available from: <https://doi.org/10.1016/j.artint.2021.103550>.
- [21] Van Hertum P, Dasseville I, Janssens G, Denecker M. The KB paradigm and its application to interactive configuration. *Theory Pract Log Program.* 2017;17(1):91-117. Available from: <https://doi.org/10.1017/S1471068416000156>.
- [22] Carbonnelle P, Vennekens J, Denecker M, Bogaerts B. Interactive Model Expansion in an Observable Environment. *Theory Pract Log Program.* 2023;23(4):648-63. Available from: <https://doi.org/10.1017/S1471068423000261>.