An Experimental Analysis of the Legal Coding Process

Matteo Cristani[1,*,†], Guido Governatori[2], Francesco Olivieri[3], Monica Palmirani[4] and Gabriele Buriola[1,†]

[1]Dept. of Computer Science, University of Verona, Verona, Italy
[2]School of Engineering and Technology, Central Queensland University, Rockhampton, Australia
[3]Independent researcher, Brisbane, Australia
[4]Dept. of Legal Studies, Alma Mater Studiorum University of Bologna, Bologna, Italy

Abstract

This paper presents a new methodology for legal coding using Deontic Defeasible Logic. Starting from normative text fragments, the approach maps them into rules and tests their accuracy through example scenarios. We illustrate the method with a sample text and report on experiments in which participants encoded various legal fragments, measuring the effort based on specific features. The Houdini tool was used for the process, and we introduce a model to predict the coding time based on legal knowledge, process familiarity, text length, and reference depth.

Keywords

Legal coding, Explainable AI, Deontic Defeasible Logic

1. Introduction

Legal coding [1, 2, 3] is the process of translating laws into a formal language, a task complicated by factors such as the complexity of legal language, specialized reasoning, and conflicting logical constraints. Many scholars [4] suggest that the *RegTech* approach could be an effective strategy. This approach proposes that we can trust decisions made by rule-based intelligent systems provided that we understand the reasoning system and ensure the rules correctly represent the domain being coded. It relies on the following conceptual steps:

- Starting from the legal text, provide a **code** onto a *standard* coding system, accepted by a large community of practice in the domain;
- **Verify** the correctness of the code by *tests*, namely applications of the coded fragment into relevant *scenarios*.

RuleML+RR'25: Companion Proceedings of the 9th International Joint Conference on Rules and Reasoning, September 22–24, 2025, Istanbul, Turkiye

*Corresponding author.
†These authors contributed equally.
matteo.cristani@univr.it (M. Cristani); g.governatori@cqu.edu.au (G. Governatori); francesco.olivieri.phd@gmail.com (F. Olivieri); monica.palmirani@unibo.it (M. Palmirani); gabriele.buriola@univr.it (G. Buriola)
0000-0001-5680-0080 (M. Cristani); 0000-0002-9878-2762 (G. Governatori); 0000-0003-0838-9850 (F. Olivieri); 0000-0002-8557-8084 (M. Palmirani); 0000-0002-1612-0985 (G. Buriola)

The purpose of this contribution is two-fold. On the one hand, the paper outlines a novel methodology for *legal coding*. On the other hand, our aim is to accurately value the time span required to execute a coding. To achieve these goals, the following tools are needed:

1. A logical apparatus consistent with human legal reasoning. In this paper, we adopt a commonly employed formalism known as *Defeasible Deontic Logic* as defined in a number of studies [5, 6]. A framework-independent language, LegalRuleML [7, 8, 9], that maps Deontic Defeasible Logic and other formalisms, is employed to provide a processable source for the devised rules.

2. A technology efficiently implementing the logical apparatus. Here, we resort to Houdini, a deontic defeasible reasoner implemented in Java [10, 11, 12].

3. A coding methodology. In this paper, a simplified methodology employed for the sole purposes of the experiment is proposed.

4. An instrumental experimental apparatus to guarantee the correct measure of the coded fragments. We espouse the paradigm of empirical software engineering that has been conceived for general measures [13].

To ease the presentation, we describe the translation process targeting only Deontic Defeasible Logic, and not LegalRuleML. The steps towards such a translation are provided in [14].

The rest of the paper is organised as follows: we briefly define the framework of Deontic Defeasible Logic in Section 2; Section 3 exposes an example of translation, showing how the translation process could be performed; Section 4 introduces some guidelines recommended to be used during the translation process; in Section 5 we describe the experiment, its design, and purpose; Section 6 discusses the results of the aforementioned experiment, and Section 7 takes some brief conclusions sketching further investigations.

2. Introduction to Deontic Defeasible Logic

The logical apparatus applied for our investigation is a deontic extension of Defeasible Logic (DL) [15]. We start by defining the language.

Let PROP be the set of propositional atoms, then the set of (plain) literals is defined as $\mathrm{PLit} := \mathrm{PROP} \cup \{\neg p \mid p \in \mathrm{PROP}\}$. The *complementary* of a literal p is denoted by $\sim p$: if p is a positive literal q then $\sim p$ is $\neg q$, if p is a negative literal $\neg q$ then $\sim p$ is q. Literals are denoted by lower-case Roman letters. Let Lab be a set of labels to represent the names of rules, which will be denoted as lower-case Greek letters.

A defeasible theory D is a tuple $(F, R, >)$, where F is the set of facts (indisputable statements), R is the rule set, and $>$ is a binary relation over R. R is partitioned into three distinct sets of rules, with different meanings to draw different "types of conclusions". *Strict rules* are rules in the classical fashion: whenever the premises are the case, so is the conclusion. *Defeasible rules* represent (along which defeaters) the non-monotonic part of the logic: if the premises are the case then typically the conclusion holds, unless we have contrary evidence that prevents us to draw such a conclusion. Lastly, *defeaters* are special rules whose purpose is to prevent contrary

evidence to be the case. It follows that in DL, through defeasible rules and defeaters, we can represent in a natural way exceptions (and exceptions to exceptions, and so forth).

We finally have the superiority relation $>$ a binary relation between rules, that is the mechanism to solve conflicts. Given two rules α and β, if we have $(\alpha, \beta) \in >$ (or simply $\alpha > \beta$), then in the scenario where both rules may fire (can be activated), α's conclusion will be preferred to β's.

A rule $\alpha \in R$ has the form $\alpha\colon A(\alpha) \rightsquigarrow C(\alpha)$, where: (i) $\alpha \in \mathrm{Lab}$ is the unique name of the rule, (ii) $A(\alpha) \subseteq \mathrm{PLit}$ is α's (set of) antecedents, (iii) $C(\alpha) = l \in \mathrm{PLit}$ is its conclusion, and (iv) $\rightsquigarrow \in \{\rightarrow, \Rightarrow, \rightsquigarrow\}$ defines the type of rule, where: $\rightarrow$ is for strict rules, $\Rightarrow$ is for defeasible rules, and $\rightsquigarrow$ is for defeaters.

Some standard abbreviations. The set of strict rules in R is denoted by R_s, and the set of strict and defeasible rules by R_{sd}; $R[l]$ denotes the set of all rules whose conclusion is l. Other abbreviations will be adopted after the introduction of modal operators.

A *conclusion* of D is a *tagged literal* with one of the following forms:

$\pm\Delta l$ means that l is *definitely proved* (resp. *strictly refuted*) in D, i.e., there is a definite proof for l in D (resp. a definite proof does not exist).

$\pm\partial l$ means that l is *defeasibly proved* (resp. *defeasibly refuted*) in D, i.e., there is a defeasible proof for l in D (resp. a defeasible proof does not exist).

The definition of proof is also the standard in DL. Given a defeasible theory D, a proof P of length n in D is a finite sequence $P(1), P(2), \ldots, P(n)$ of tagged formulas of the type $+\Delta l$, $-\Delta l$, $+\partial l$, $-\partial l$, where the proof conditions defined in the rest of this section hold. $P(1..n)$ denotes the first n steps of P.

Next, we introduce the notion of extension of a defeasible theory; informally, an extension is everything that is derived and disproved.

Definition 1 (Theory Extension). *Given a defeasible theory D, we define the set of positive and negative conclusions of D as its* extension*:*

$$E(D) = (+\Delta, -\Delta, +\partial, -\partial),$$

where $\pm\# = \{l |\, l$ *appears in* D *and* $D \vdash \pm\# l\}$, $\# \in \{\Delta, \partial\}$.

The deontic part of the logic encompasses obligations, permissions and prohibitions. A prescriptive behaviour like "At traffic lights it is forbidden to perform a U-turn unless there is a 'U-turn Permitted' sign" can be formalised via the general obligation rule

$$AtTrafficLight \Rightarrow_{\mathsf{O}} \neg UTurn$$

and the exception through the permissive rule

$$UTurnSign \Rightarrow_{\mathsf{P}} UTurn.$$

An alternative equivalent notation for permissions and obligations is to move the obligation or permission tag into the formula as shown below.

$$AtTrafficLight \Rightarrow \mathsf{O}\neg UTurn$$

and

$$UTurnSign \Rightarrow \mathsf{P}\, UTurn.$$

The obligation to a negated literal establishes the prohibition of the literal; therefore, the above determines the prohibition of passing on a red traffic light.

While [16] discusses how to integrate strong and weak permission in Deontic Defeasible Logic, in this paper, we restrict our attention to the notion of strong permission, namely, when permissions are explicitly stated using permissive rules, i.e., rules whose conclusion is to be asserted as a permission.

Following the ideas of [17], obligation rules gain more expressiveness with the *compensation operator* $\otimes$ which is used to model reparative chains of obligations. Roughly, $a \otimes b$ means that a is the primary obligation, but if we fail to obtain (or to comply with) a (by either not being able to prove a, or by proving $\sim a$), then b becomes the new obligation in force. This operator is used to build chains of preferences (or repair chains), called $\otimes$-expressions, that are formed as follows: (i) every plain literal is an $\otimes$-expression, (ii) if A is an $\otimes$-expression and b is a plain literal, then $A \otimes b$ is an $\otimes$-expression [16]. In this paper, repair chains appear only as conclusion of an obligaion rule. Moreover, if the conclusion $C(\alpha)$ of an obligation rule is a $\otimes$-expression $\hat{\otimes}$, then (with a slight abuse of notation) we denote with $C(\alpha)$ also the set of literals appearing in $\hat{\otimes}$

We summarise proof conditions, starting with applicability and discardability, and following the structure of proofs for constituents and deontic formulas. In the following, R^{C} denotes the set of constitutive rules (i.e. without modal operators), R^{P} the set of permission rules, and R^{O} the set of obligation rules. We keep the notation for $R[l]$ adopted before; moreover, we extend it to include $\otimes$-expression, namely $R^{\mathsf{O}}[l, i]$ denotes the set of obligation rules with literal l in position i of the reparative chain.

Definition 2 (Applicability). *Let $D = (F, R, >)$ be a deontic defeasible theory. We say that rule $\alpha \in R^{\mathsf{C}} \cup R^{\mathsf{P}}$ is* applicable *at $P(n+1)$ iff for all $a \in A(\alpha)$*

1. *if $a \in \mathrm{PLit}$, then $+\partial_{\mathsf{C}} a \in P(1..n)$,*
2. *if $a = \Box q$, then $+\partial_{\Box} q \in P(1..n)$, with $\Box \in \{\mathsf{O}, \mathsf{P}\}$,*
3. *if $a = \neg\Box q$, then $-\partial_{\Box} q \in P(1..n)$, with $\Box \in \{\mathsf{O}, \mathsf{P}\}$.*

We say that rule $\alpha \in R^{\mathsf{O}}$ is applicable at index i and *$P(n+1)$ iff Conditions 1–3 above hold and*

4. *$\forall c_j \in C(\alpha)$, if $j < i$, then $+\partial_{\mathsf{O}} c_j \in P(1..n)$ and $+\partial_{\mathsf{C}} \sim c_j \in P(1..n)$.*[1]

Definition 3 (Discardability). *Assume a deontic defeasible theory D, with $D = (F, R, >)$. We say that rule $\alpha \in R^{\mathsf{C}} \cup R^{\mathsf{P}}$ is* discarded *at $P(n+1)$, iff there exists $a \in A(\alpha)$ such that*

1. *if $a \in \mathrm{PLit}$, then $-\partial_{\mathsf{C}} a \in P(1..n)$, or*
2. *if $a = \Box q$, then $-\partial_{\Box} q \in P(1..n)$, with $\Box \in \{\mathsf{O}, \mathsf{P}\}$, or*

[1] As discussed above, we are allowed to move to the next element of an $\otimes$-expression when the current element is violated; to have a violation, we need (i) the obligation to be in force, and (ii) that its content does not hold. See [18] for a deeper discussion.

3. *if* $a = \neg\Box q$, *then* $+\partial_{\Box} q \in P(1..n)$, *with* $\Box \in \{\mathsf{O}, \mathsf{P}\}$.

We say that rule $\alpha \in R^{\mathsf{O}}$ *is* discarded at index i and $P(n+1)$ *iff either at least one of the Conditions 1–3 above hold, or*

4. $\exists c_j \in C(\alpha)$, $j < i$ *such that* $-\partial_{\mathsf{O}} c_j \in P(1..n)$, *or* $-\partial_{\mathsf{C}} {\sim} c_j \in P(1..n)$.

Note that discardability is obtained by applying the principle of *strong negation* to the definition of applicability. The strong negation principle simplifies a formula by moving all negations to an innermost position in the resulting formula, replacing the positive tags with the respective negative tags, and the other way around see [19]. Positive proof tags ensure that there are effective decidable procedures to build proofs; the strong negation principle guarantees that the negative conditions provide a constructive and exhaustive method to verify that a derivation of the given conclusion is not possible. Accordingly, condition 3 of Definition 2 allows us to state that $\neg\Box p$ holds when we have a (constructive) failure to prove p with mode $\Box$ (for obligation or permission), thus it corresponds to a constructive version of negation as failure.

Definition 4 (Constitutive Proof Conditions). *Given a DDL theory* $(F, R, >)$

$+\partial_{\mathsf{C}} l$: *If* $P(n+1) = +\partial_{\mathsf{C}} l$ *then*
- (1) $l \in F$, *or*
- (2) (1) $\sim l \notin F$, *and*
  - (2) $\exists \beta \in R^{\mathsf{C}}_{sd}[l]$ *s.t.* β *is appl., and*
  - (3) $\forall \gamma \in R^{\mathsf{C}}[\sim l]$ *either*
    - (1) γ *is disc., or*
    - (2) $\exists \zeta \in R^{\mathsf{C}}[l]$ *s.t.*
      - (1) ζ *is appl. and*
      - (2) $\zeta > \gamma$.

$-\partial_{\mathsf{C}} l$: *If* $P(n+1) = -\partial_{\mathsf{C}} l$ *then*
- (1) $l \notin F$ *and either*
- (2) (1) $\sim l \in F$, *or*
  - (2) $\forall \beta \in R^{\mathsf{C}}_{sd}[l]$, *either* β *is disc., or*
  - (3) $\exists \gamma \in R^{\mathsf{C}}[\sim l]$ *such that*
    - (1) γ *is appl., and*
    - (2) $\forall\, \zeta \in R^{\mathsf{C}}[l]$, *either*
      - (1) ζ *is disc., or*
      - (2) $\zeta \not> \gamma$.

A literal is defeasibly proved if: it is a fact, or there exists an applicable, defeasible rule supporting it (such a rule cannot be a defeater), and all opposite rules are either discarded or defeated. To prove a conclusion, not all the work has to be done by a stand-alone (applicable) rule (the rule witnessing condition (2.2) in $+\partial_C l$): all the applicable rules for the same conclusion (may) contribute to defeating applicable rules for the opposite conclusion.

Example 1. *Let* $D = (F = \{a, b, c, d, e\}, R, > = \{(\alpha, \varphi), (\beta, \psi)\})$ *be a theory such that*

$$R = \{\alpha\colon a \Rightarrow_{\mathsf{C}} l \qquad \beta\colon b \Rightarrow_{\mathsf{C}} l \qquad \gamma\colon c \Rightarrow_{\mathsf{C}} l$$
$$\varphi\colon d \Rightarrow_{\mathsf{C}} \neg l \qquad \psi\colon e \Rightarrow_{\mathsf{C}} \neg l \qquad \chi\colon g \Rightarrow_{\mathsf{C}} \neg l\}.$$

Here, $D \vdash +\partial_{\mathsf{C}} f$, for each $f \in F$ and, by Condition (1) of $+\partial$. Therefore, all rules except χ (which is discarded) are applicable: χ is indeed discarded since no rule has g as consequent nor g is a fact. The team defeat supporting l is made by α, β and γ; whereas the team defeat supporting $\neg l$ is made by φ and ψ. Given that α defeats φ and β defeats ψ, we conclude that $D \vdash +\partial_{\mathsf{C}} l$. Note that, despise being applicable, γ does not effectively contribute in proving $+\partial_{\mathsf{C}} l$, i.e., D without γ would still prove $+\partial_{\mathsf{C}} l$.

Suppose to change D such that both α and β are defeaters. Even if γ defeats neither φ nor ψ, γ is now needed to prove $+\partial l$ as Condition (2.2) requires that at least one applicable rule must be a defeasible rule. Below we present the proof conditions for obligations.

Definition 5 (Obligation Proof Conditions). *Given a DDL theory* $(F, R, >)$

$+\partial_{\mathsf{O}} l$: *If* $P(n+1) = +\partial_{\mathsf{O}} l$ *then*
$\exists \beta \in R^{\mathsf{O}}_{sd}[l, i]$ *s.t.*
(1) β *is applicable at index* i *and*
(2) $\forall \gamma \in R^{\mathsf{O}}[\sim l, j] \cup R^{\mathsf{P}}[\sim l]$ *either*
(1) γ *is discarded (at index* j*), or*
(2) $\exists \zeta \in R^{\mathsf{O}}[l, k]$ *s.t.*
(1) ζ *is applicable at index* k *and*
(2) $\zeta > \gamma$.

$-\partial_{\mathsf{O}} l$: *If* $P(n+1) = -\partial_{\mathsf{O}} l$ *then*
$\forall \beta \in R^{\mathsf{O}}_{sd}[l, i]$ *either*
(1) β *is discarded at index* i*, or*
(2) $\exists \gamma \in R^{\mathsf{O}}[\sim l, j] \cup R^{\mathsf{P}}[\sim l]$ *s.t.*
(1) γ *is applicable (at index* j*), and*
(2) $\forall \zeta \in R^{\mathsf{O}}[l, k]$ *either*
(1) ζ *is discarded at index* k*, or*
(2) $\zeta \not> \gamma$.

Note that: (i) in Condition (2) γ can be a permission rule as explicit, opposite permissions represent exceptions to obligations, whereas ζ (Condition 2.2) must be an obligation rule as a permission rule cannot reinstate an obligation, and that (ii) l may appear at different positions (indices i, j, and k) within the three $\otimes$-chains. The example below supports the intuition behind the restriction to obligation rules in Conditions (2.2). Since they are very similar to the obligation ones, the proof conditions for permission are omitted.

3. An example of translation

To illustrate DDL's practical use, we present two coding examples outside the experimental set. Chosen to reflect a broad range of cases within time constraints, these simple yet meaningful examples highlight the inherent diversity in legal text length and complexity within the selected normative framework.

The first example features a legal excerpt from Italian Criminal Law, used as the translation source. The full text provided for translation to the experimenters is available upon request. We present the translation process for the main article and the relevant exception only. This text appears as entry 1 in Table 1 .

Art. 575 (Italian Criminal Code) - Homicide

Whoever causes the death of a person is punished with imprisonment of no less than twenty-one years.

Exceptions - Aggravating Circumstances:
The penalty of **life imprisonment** applies when the crime is committed:

- Using poisonous substances or other insidious means;
- With premeditation.

The translation process made by the experimenters generates a Deontic Defeasible Logic expression, adhering to the above specified guidelines.

Article 575 can be translated as follows (the reported one is the actual translation by one of the experimenters).

$\Rightarrow \mathsf{O} \sim \text{death} \otimes \text{basic\_punishment}$
$\text{basic\_punishement} \Rightarrow \text{imprisonment} := 21\text{years}$

For the exceptions, we adopt the principle that the translation should be on two steps. Firstly we map the deontic rule as it would have been a primary one, and not an exception. In the above case we shall have the translations below.

poisonous_means $\Rightarrow$ O $\sim$ death $\otimes$ life_imprisonment
premeditation $\Rightarrow$ O $\sim$ death $\otimes$ life_imprisonment

In addition to the legal background, we included case descriptions illustrating situations where the codes might apply. Here, we present a single case related to homicide to demonstrate the specific procedure.

Brothers Alberto and Mario lived together since their parents' death, never marrying or having stable relationships. Alberto mistreated Mario for years, forcing him to do chores and mocking him. After Alberto became wheelchair-bound due to illness, the abuse worsened with frequent rage and insults. Unable to endure it any longer, Mario poisoned Alberto with arsenic, leading to his death within three days.

On request of judgment by human experimenters, this situation is translated into the applicability of the article 575 and the aggravating circumstances of usage of poisonous means and premeditation. In practice, the translation is a set of facts and propositional rules:

FACTS	**RULES**
Alberto	Alberto, Mario, Mario_buys_Arsenic,
Mario	Mario_poisons_Alberto $\Rightarrow$ Death
living_together	Alberto, Mario, Mario_buys_Arsenic,
Alberto_mistreats_Mario	Mario_poisons_Alberto $\Rightarrow$ Poisonous_means
ill_Alberto	Alberto, Mario, Mario_buys_Arsenic,
wheelchair_Alberto	Mario_poisons_Alberto $\Rightarrow$ Premeditation
rage_Alberto_Mario	
Mario_buys_Arsenic	
Mario_poisons_Alberto	

Based on the facts and rules, we compute the *extension* of the propositional theory. Applying the deontic rules for homicide and aggravating circumstances, we logically conclude that Mario must be sentenced to life imprisonment.

In private law, we created many definitions beyond direct deontic rules with punishments, such as rules for coding contracts. One of the norms we coded, taken from the Italian Private Law Code, is shown below.

Art. 1470 Italian private law code
The sale is the contract that aims to transfer the ownership of a thing or the transfer of another right in exchange for the payment of a price.

Again, we have the case:

Celeste saw a TV infomercial for Marvellous Blade knives, claimed to cut cobblestones and drainpipes without damage. Skeptical but curious, she called to buy only after testing them herself. The company, confident and eager for sales, agreed and sold her the set.

The totality of experimenters classified the verbal agreement reached by Celeste and Marvellous Blade a sale.

4. Coding guidelines

The experimental pipeline is described in Section 5 and precisely in Figure 1. To frame our investigation, we focused on accurately measuring the time needed for manual coding. The experimental setup assessed both translation quality—using standard prediction performance indicators—and coding duration. A preliminary workshop was held to train participants before providing data and conducting the coding. We then tested both the normative background and scenarios using an automated tool, resulting in a reliable estimate of coding time.

Details on the organisation of the workshop are provided, along with the organisation of the experiments, in Section 5. The encoding and testing methodology we introduce here is essentially formed by three phases:

1. The *normative background encoding phase*. In this process, a coder transforms a legal text into a coded version using Deontic Defeasible Logic (DDL). While LegalRuleML—supported by existing editors with annotation tools—might simplify implementation, we use DDL exclusively in this work. Despite its broader compatibility, LegalRuleML has drawbacks such as lower transparency and more verbose syntax, making DDL more suitable for our current purposes. The implemented rules shall be of four types:
   - *Strict rules*, namely definitions, that are named, in the terminology of LegalRuleML, *constituent rules*.
   - *Propositional defeasible rules* aiming at capturing concrete aspects of the behaviour of the reality, such as physics, biology but also very elementary social rules, without obligations and permissions.
   - *Superiorities* specifically, preferences between two propositional or deontic defeasible rules indicate that when both are applicable (i.e., their antecedents are satisfied), the preferred rule takes precedence.
   - *Deontic Defeasible rules* representing deontic operators.
2. The *scenario encoding phase* in which a coder maps scenarios fitting a specific legal background into LegalRuleML (using Deontic Defeasible Logic). There are two types of scenario coding: Private Law, which requires matching meta-rules [20, 21, 22, 23], and Criminal Law, which uses flat Deontic Defeasible Logic with numeric constraints and assignments to model repair chains like imprisonment time or fines. This matter has been discussed in [10].
3. The *evaluation phase* involves testing the coded examples against the normative background. The coder compares their decision on the case with the decision made by a machine, specifically the reasoner processing both the background and scenario. If the decisions differ, it signals a problem that requires further evaluation.

At the end of the methodology, we produced a baseline implementation of the specification, including test cases with expected outcomes. We encoded, tested, and evaluated a normative background, with results marked as pass or review based on errors identified in specific scenarios.

Inevitably, the methodology has some critical aspects that we need to address in the very beginning:

- *Differences in language*. Encoding a large set of norms would be impractical for a single person, so a team approach is necessary. However, this could lead to inconsistencies, as different coders might use varying terms to represent the same concepts. Even a single coder might overlook the need to standardize terms for synonyms or antonyms. To address this, three tools are recommended: (1) a glossary with a thesaurus and archetypical terms for synonyms and antonyms, (2) a set of guidelines to minimize variations, and (3) a cyclic methodology where these steps are repeated until errors fall below an acceptable threshold.

- *Implicitness of defeasible rules*. Coders may assume certain conditions are obvious—for example, that the rule *running is forbidden in quiet area* does not apply to individuals unable to run. However, if such exceptions (e.g., wheelchair users) aren't explicitly encoded, they won't be considered. To address this, more time should be devoted to encoding based on empirical observations, possibly by assigning this task to specific team members.

The recommendations we provided during the instruction workshop can be summarised in the following points:

a) When translating a noun phrase, any adjective that specializes the noun should be combined with it into a single positive literal. For example, *divorced spouse* becomes *divorced_spouse*.

b) When a copulative verb (e.g., *to be*, *to become*) appears, the subject is defined by the nominal part using a strict rule. For example, *A sale is a contract* is translated as *sale* $\rightarrow$ *contract*.

c) When a verb different from copulative ones occurs in the sentence it has to be attached to subject and complements to form a single literal. For instance *Mario buys Arsenic* translates into *Mario_buys_Arsenic*

d) When a modal of obligation or permission appears, it should stay attached to the noun or noun phrase as generated. Similarly, if a punishment follows specific behavior, a *repair chain* is introduced with the punishment as a literal. This punishment literal heads a propositional rule assigning a value for its duration. For general obligations or permissions expressed generically, we recommend using a deontic rule without a tail. For example, *Whoever causes the death of a person is punished with no less than twenty-one years* is translated as in the basic example of Section 3.

e) Any conditional sentence—explicit or implied by verbs like "causes" or "generates"—should be translated into a defeasible rule, possibly with deontic labels and repair chains if punishments are involved.

These guidelines are not automation rules, as automated legal translation has so far yielded disappointing results. In practice, despite the guidelines, individual variations may still occur.

5. An experiment in legal coding

The coding example in Sec. 3 was carried out to provide a code for a legal text along with its translation into Deontic Defeasible Logic. The translation team was intentionally composed of non-experts in both DDL and legal matters to ensure a neutral perspective. Qualitatively, the search for explanations proved important, but more significantly, it shaped a coding process focused on measurable performance.

This coding activity reflects observable human behavior, making it measurable with software engineering metrics. While measuring parameters like coding time is straightforward, it is more challenging to establish correlations with factors related to the humans involved in the process. To investigate this aspects, we devised the following experiment whose structure is reported in Figure 1.

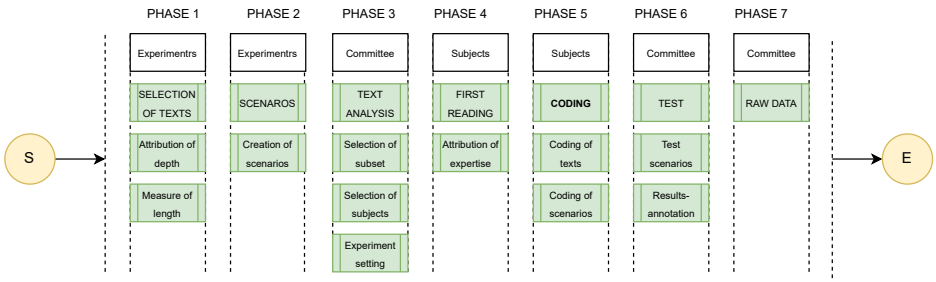

Figure 1: The experimental design.

Preparing the setting

We selected eleven legal texts from the Italian penal and private codes, along with their related legal references. Three expert jurists conducted this work to identify relevant variables for measurement. The jurists evaluated the texts based on a *depth scale* of 1 to 5, where depth 1 corresponds to constitutional norms or related implementation laws, depth 2 covers laws dependent on the constitution and related norms, and so on. The jurists agreed on a maximum depth of 5 for the relevant norms. Texts from international treaties or conventions (e.g., the Universal Declaration of Human Rights) are considered depth 0. Depth 1 is for constitutional norms, depth 2 for general norms like fundamental codes, depth 3 for general laws, depth 4 for decrees and regulations, and depth 5 for technical annexes. We selected seven of the eleven texts to balance length growth ensuring a good sample distribution across depths, as shown in Table 1. The length was measured by the *number of characters*—an objective metric unaffected by format or structure.

After selecting the legal texts, we asked the experimenters to add six scenarios where the normative background applies and to identify the resulting legal consequences. These six were selected from a larger set of 17 scenarios previously prepared.

Text	1	2	3	4	5	6	7
Length	673	891	1287	1455	1822	2011	2344
Depth	2	3	3	2	4	3	3

Table 1
Text lengths and depths for the experiment.

Subject	**T. 1**	**T. 2**	**T. 3**	**T. 4**	**T. 5**	**T. 6**	**T. 7**
1	0.7	0.6	0.9	1.0	0.5	0.8	0.8
2	0.7	0.9	0.9	1.0	0.5	0.8	0.8
3	0.8	0.6	0.6	1.0	0.6	0.8	0.8
4	0.8	0.5	0.6	1.0	0.6	0.9	0.9
5	0.6	0.8	0.9	0.6	0.6	0.8	0.8
6	0.6	0.9	0.9	0.6	0.6	0.8	0.8
7	0.8	1.0	0.5	0.9	0.5	0.7	0.9
8	0.7	1.0	0.9	0.9	0.6	0.8	1.0
9	0.3	0.3	0.3	0.3	0.3	0.3	0.3
10	0.2	0.2	0.2	0.2	0.2	0.2	0.2
11	0.1	0.1	0.1	0.1	0.1	0.1	0.1
12	0.1	0.1	0.1	0.1	0.1	0.1	0.1
13	0.1	0.1	0.1	0.1	0.1	0.1	0.1
14	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Table 2
Experimental subjects and their expertise.

The legal coding

We selected 30 participants—23 jurists and 7 non-jurists—for legal coding experiments. After a workshop on Deontic Defeasible Logic basics, they produced experimental codes to assess their coding ability and estimate required time. While some performed well, only 14—both expert and less experienced jurists and knowledgeable non-jurists—completed the coding accurately. Those who succeeded were then asked to code scenarios and were enrolled in the second phase upon passing.

Participants first rated their expertise on the texts using a decimal percentage scale. Since most Italian jurists specialize in either penal or private law, we included a mix of both. Some jurists rated themselves differently across topics, complicating performance analysis. To balance this, we also selected coding experts without legal backgrounds. The self-assessments are shown in Table 2.

The fourteen selected subjects individually coded all seven texts, with access to the normative background and Deontic Defeasible Logic rules. Their work was measured by the length of the coded output and the time taken. After coding, a verification phase allowed participants to exchange codes, receive feedback, and make corrections. The combined time for this exchange and re-coding was minimal and consistent across subjects, so it was included with the initial coding time. Given its small impact and the likelihood of such consultation in practice, these

phases were not separated in performance measurements.

After coding the texts, subjects were assigned to code scenarios, with evaluators randomly selected to avoid coding scenarios for their own texts. Coders had access to both the original and coded texts. Initially, we underestimated the time needed for scenario coding, assuming it was negligible. However, scenario coding time depends on the complexity of the legal background and is roughly proportional to the time required per character for encoding the background. We adjusted the experiment accordingly to measure this time accurately.

Tests on the Scenarios

The testing phase was conducted by a second group of five technology experts who used the coded texts to test the scenarios. This phase required negligible time. Although some minor errors in the submitted files needed correction, the extra time was minimal and expected to decrease with improvements in the Houdini interface [10, 11, 12, 24]. Overall, this added time amounts to about a 20% increase over the total coding time for backgrounds and scenarios.

We assessed scenario analysis accuracy by linking the number of errors to each norm coding. Subjects then received feedback and corrected the errors, with most fixes done almost immediately. The additional time needed was roughly proportional to the initial coding time. On average, the error rate was 6%, and recoding required about 5.8%. These figures are influenced by the limitations of the procedure.

6. The results of the experiment: a general analysis

The experiment shows that legal coding is a reliable process, with a low error rate and straightforward, quick corrections compared to the overall coding time.

Regarding length, the coded texts in formal language (Deontic Defeasible Logic) are on average 19% longer than the original texts (in number of characters). There is anyway, a significant standard deviation of 9% and a span from 2% to 33%. However, while eliminating the two outliers of the extreme values the result is 20% with a standard deviation of only 3%. This leads to the claim that the sample is too small to take a reasonable conclusion on the actual distribution and therefore we should derive a preliminary acceptance of the value per se.

Analyzing performance, we focused on coding time relative to text length, measured by the *number of characters*—an objective metric unaffected by format or structure. Two key metrics emerged: mean and median coding time per character. Coding times ranged from 1.8 to 5.91 seconds per character, averaging 3.99 seconds with a standard deviation of 0.94. The median time was 4.06 seconds, close to the average.

Applying the Shapiro-Wilk test on the population [25], we get that the departure from normality is below 0.01 on a 0.05 confidence interval span. Beyond validating the experimental setup, the normal distribution—expected for continuous variables—shows a relatively narrow spread around the mean. This indicates that the average coding time is a reliable predictor, with some adjustments to be discussed later. Since the average page contains 1800 characters, we can presuppose that the time required to code a page is roughly two hours (spanning from a minimum of 54 minutes to a maximum of 3h57').

As discussed in Section 5, having collected data regarding expertise and depth we can now look at these values, aiming to understand whether there is an influence of the expertise in the delivery time.

If we measure the correlation index between the time per character with the self-valued expertise, then we have a value of -28.67% compatible with the hypothesis of a weak inverse linear correlation with the expertise; namely experts code slightly quicker than non-experts. With a class partition based on decimals, we can also figure this out as reported in Figure 2, where the time of coding per character on average by classes of expertise from 0 to 1 is exposed.

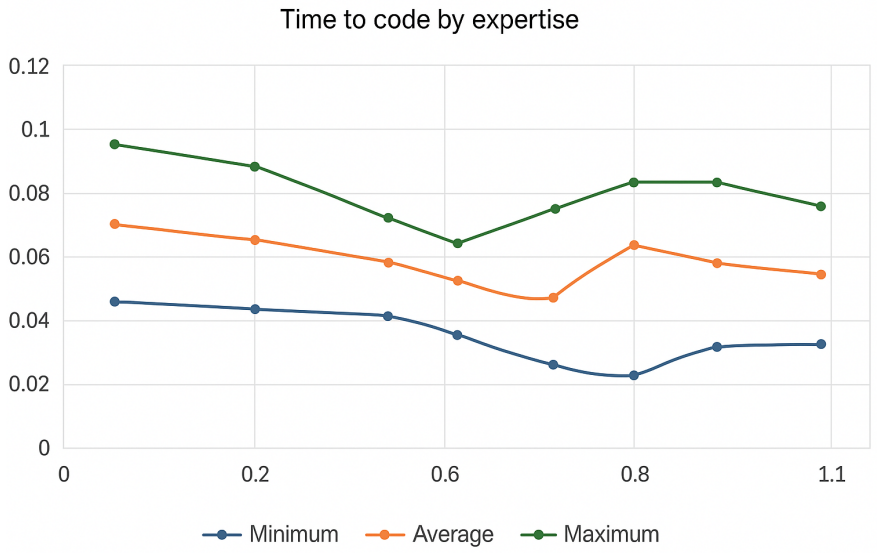

Figure 2: Time to deliver coding by classes of expertise.

A qualitative analysis of this graphical representation gives two hints:

- Up to the expertise of 0.8 there is a rather evident negative correlation of the time required to code, that improves moderately while growing the expertise;
- Once we pass the 0.8, although performance worsens on that scale, a negative correlation remains. As often seen in self-evaluations, less experienced individuals tend to rate themselves higher—a tendency known as the Dunning-Kruger effect. This may have led the most expert participants to underrate themselves, influencing the observed results.

Regarding, on the other hand, the relation between depth and time to code, there is again a very similar negative correlation, to -28.71% which in turn explains itself by the higher difficulty of a text with higher depth.

Regarding bias analysis, we have to consider the following aspects:

1. Regarding sample size, a long-term study tracking years of practical coding would provide stronger validation. However, expecting hundreds of subjects and thousands of hours

of observation to eliminate bias is unrealistic. The value of this investigation lies in the method itself, and we believe we have a solid foundation.

2. Self-estimating expertise is challenging and prone to bias. We believe pursuing complex, double-blind evaluations isn't worthwhile, as assessing expertise is highly subjective and such methods would likely hinder success.

3. Language and normative background inevitably influence the test. It's nearly impossible to design a test—such as one measuring coding time—that is entirely independent of these factors. However, the robustness of the chosen approach allows it to be adapted for different languages and normative contexts.

4. Texts on taxes, pensions, or economic topics often involve non-trivial computational content. However, if this content is limited to a few pages, its impact on coding time is minimal. If it's spread throughout the text, its effect is evenly distributed and can be captured by a suitable coefficient.

Overall, we propose that, with few exceptions, coding time scales linearly with text length. For sufficiently long texts, the time required depends mainly on length, through a coefficient linked to the text's nature rather than its specific content. Further empirical validation is needed to support this hypothesis.

A further analysis regards the variability factors of the translation operations. Although we provided, during the preparatory workshop mentioned above, a general methodology with some guidelines as described in Section 4, there are factors inevitably generating variability:

1. Natural language expresses modality in many varied and hard-to-classify ways, including modal verbs and expressions like "is entitled to," "has the duty of," or "is responsible for." These forms can be interpreted differently depending on context.

2. When a text is extended with related material, as in the first step of the methodology described above.

3. In some cases, multiple adjectives applied to a noun can lead to ambiguity—either they combine into a single noun phrase or form two distinct descriptive phrases.

The error rate is relatively low compared to similar experimental settings. This is partly due to a bias in phases 6 and 7, where the same participants who coded the norms also coded the scenarios for synchronization, reducing potential inconsistencies. If different individuals coded the scenarios and the normative background separately—even with prior synchronization—the error rate would likely increase. However, with clear guidelines, a long-term convergence process could eventually ensure consistent performance.

6.1. A scenario with the Italian Criminal Code

Based on the estimate of 4 seconds per character, we can roughly gauge the *coding effort* to translate the Italian Criminal Code. Although not the actual *elapsed time*, it serves as a useful preliminary measure.

Let us start with some data that shall help us in both of the activities defined above. In Table 3 we report raw data on the Italian Criminal Code.

Length	435,939 characters
Number of sections (libri)	3
Number of subsections (titoli)	8, 14, 4
Number of articles	832
Typical depth	1, 2
Interference with other codes	Criminal procedure code, Italian Constitution

Table 3
Raw data on the Italian Criminal Code.

Using the methodology, we estimate a total effort of 485 hours to encode the background alone. Expert opinions from the workshop suggest that scenarios are at least twice as long as the background. There is also a large collection of such cases available for reference (for instance, the UK National Archives have a Section dedicated to this topic - https://www.nationalarchives.gov.uk/help-with-your-research/research-guides/criminal-court-cases-an-overview/), so the effort would consist of *finding* right cases that could be estimated in a further 100% of the total. Overall, we can make the following estimate:

- Time to code: 485 hours;
- Time to retrieve scenarios: 485 hours;
- Time to code scenarios: 485*3=1455 hours;
- Time to test: (485+1455)*0,2=388 hours.

The total time required is 2813 hours. If one person handles both coding and testing, this amounts to roughly 23 person-months. The highest feasible parallelization is by subsections—26 in total. Assuming equal length, this could reduce the effort to about one month. However, true parallelization is unlikely for three main reasons:

- We need to forecast time to sort discrepancies. This shall account to a very long excess, that is very difficult to determine a priori, without further data.
- Better insight into internal dependencies is needed to split the work, as this division might unpredictably increase workload in other phases.
- The testing phase will result more complicated if conducted by separate teams in complex cases.
- Some additional time could be required to properly connect the formal translation to other IT tools.

Many criminal case descriptions aren't schematic, causing longer coding times and harder test predictions. Complex cases involving multiple behaviors or subjects further increase coding difficulty and scenario length.

Despite these challenges, formalizing the entire Criminal Code has notable benefits: 1) its polynomial computational cost allows for full translation without the need to extract specific subparts, and 2) if extraction is needed, the atom dependency graph makes isolating the required rules easier.

7. Conclusions and further developments

The experiment has given important hints regarding any future process of translation of the law based on scientific ground and solid engineering methodologies.

First of all, it is clear that we have by no means a technique that can be used to predict the required coding time for *single pieces of the law*. The ability to measure coding time relies on *large amounts of texts* and *numerous translators*, since the individual variance is very significant; and the mean time per character, computed as a result of the investigation, only makes sense on a large base. Secondly, the large variance referred above is mainly individual, depending on the expertise of the translator more than on the depth (in some sense, a measure of the complexity of references) of the text.

We are integrating the results of this paper with a further study using segregation methods, particularly unsupervised clustering, to divide a legal text corpus into subcorpora. Each subcorpus can be coded by a subset of the encoding team, minimizing overlap and allowing for better consensus on the language token set. This reduces individual variance and makes the translation effort feasible by involving a larger number of people.

Acknowledgments

This project is co-funded by the European Union - NextGenerationEU under the National Recovery and Resilience Plan (PNRR) - Mission 4 Education and research - Component 2 From research to business - Investment 1.1 Notice PRIN 2022 (DD N. 104 del 02/02/2022), title Smart Legal Order in DigiTal Society (SLOTS), proposal code 2022LRL2C2 - CUP J53D23005610006.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT in order to: Paraphrase and reword, Grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] N. Godfrey, M. Burdon, Fidelity in legal coding: applying legal translation frameworks to address interpretive challenges, Information and Communications Technology Law 33 (2024) 153 – 176. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85185504076&doi=10.1080%2f13600834.2024.2312620&partnerID=40&md5=efe26ddc13ea30459027c056eab70cb8. doi:10.1080/13600834.2024.2312620.

[2] A. Huggins, M. Burdon, A. Witt, N. Suzor, Digitising legislation: connecting regulatory mind-sets and constitutional values, Law, Innovation and Technology 14 (2022) 325–354. doi:10.1080/17579961.2022.2113670.

[3] A. Witt, A. Huggins, G. Governatori, J. Buckley, Encoding legislation: a methodology for enhancing technical validation, legal alignment and interdisciplinarity, Artificial Intelligence and Law 32 (2024) 293–324. doi:10.1007/s10506-023-09350-1.

[4] S. Batsakis, G. Baryannis, G. Governatori, I. Tachmazidis, G. Antoniou, Legal representation and reasoning in practice: A critical comparison, Frontiers in Artificial Intelligence and Applications 313 (2018) 31 – 40. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85059635121&doi=10.3233%2f978-1-61499-935-5-31&partnerID=40&md5=f035ae481093fb6db46378d00c74c70e. doi:10.3233/978-1-61499-935-5-31.

[5] G. Governatori, Practical normative reasoning with defeasible deontic logic, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11078 LNCS (2018) 1–25. doi:10.1007/978-3-030-00338-8_1.

[6] G. Governatori, A. Rotolo, E. Calardo, Possible world semantics for defeasible deontic logic, LNCS (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7393 LNAI (2012) 46–60. doi:10.1007/978-3-642-31570-1_4.

[7] T. Athan, H. Boley, G. Governatori, M. Palmirani, A. Paschke, A. Wyner, Oasis legalruleml, 2013, p. 3 – 12. doi:10.1145/2514601.2514603.

[8] T. Athan, G. Governatori, M. Palmirani, A. Paschke, A. Wyner, Legalruleml: Design principles and foundations, LNCS (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9203 (2015) 151 – 188. doi:10.1007/978-3-319-21768-0_6.

[9] M. Palmirani, G. Governatori, A. Rotolo, S. Tabet, H. Boley, A. Paschke, Legalruleml: Xml-based rules and norms, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7018 LNCS (2011) 298 – 312. doi:10.1007/978-3-642-24908-2_30.

[10] M. Cristani, G. Governatori, F. Olivieri, L. Pasetto, F. Tubini, C. Veronese, A. Villa, E. Zorzi, Houdini (unchained): an effective reasoner for defeasible logic, in: CEUR Workshop Proceedings, volume 3354, 2022.

[11] M. Cristani, G. Governatori, F. Olivieri, L. Pasetto, F. Tubini, C. Veronese, A. Villa, E. Zorzi, The architecture of a reasoning system for defeasible deontic logic, in: Procedia Computer Science, 2023, pp. 4214–4224. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85183536153&doi=10.1016%2fj.procs.2023.10.418&partnerID=40&md5=a2cfea0700fbe01540e05c2b80d4215b. doi:10.1016/j.procs.2023.10.418, vol. 225.

[12] M. Cristani, F. Olivieri, G. Governatori, G. Buriola, Simulating the law in a multi-agent system, in: CEUR Workshop Proc., 2024, pp. 217–232. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85200109297&partnerID=40&md5=6443fb0e73d56e6f0b6530e318cc1daf, vol. 3735.

[13] T. Moher, G. Schneider, Methodology and experimental research in software engineering, International Journal of Man-Machine Studies 16 (1982) 65 – 87. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0020085901&doi=10.1016%2fS0020-7373%2882%2980072-2&partnerID=40&md5=f1160d8d72c6a3b9bfeb18a7f0f3e554.

doi:10.1016/S0020-7373(82)80072-2.
[14] M. Palmirani, M. Martoni, A. Rossi, C. Bartolini, L. Robaldo, Legal ontology for modelling gdpr concepts and norms, Frontiers in Artificial Intelligence and Applications 313 (2018) 91 – 100. doi:10.3233/978-1-61499-935-5-91.
[15] G. Antoniou, D. Billington, G. Governatori, M. Maher, Representation results for defeasible logic, ACM Tran. on Comp. Logic 2 (2001) 255–287. doi:10.1145/371316.371517.
[16] G. Governatori, F. Olivieri, A. Rotolo, S. Scannapieco, Computing strong and weak permissions in defeasible logic, J. Philos. Log. 42 (2013) 799–829. URL: https://doi.org/10.1007/s10992-013-9295-1. doi:10.1007/s10992-013-9295-1.
[17] G. Governatori, A. Rotolo, Logic of violations: A gentzen system for reasoning with contrary-to-duty obligations, Australasian Journal of Logic 4 (2006) 193–215. URL: http://ojs.victoria.ac.nz/ajl/article/view/1780.
[18] G. Governatori, Burden of compliance and burden of violations, in: A. Rotolo (Ed.), 28th JURIX on Legal Knowledge and Information Systems, Frontieres in Artificial Intelligence and Applications, IOS Press, Amsterdam, 2015, pp. 31–40.
[19] G. Governatori, V. Padmanabhan, A. Rotolo, A. Sattar, A defeasible logic for modelling policy-based intentions and motivational attitudes, Log. J. IGPL 17 (2009) 227–265. doi:10.1093/jigpal/jzp006.
[20] G. Governatori, F. Olivieri, A. Rotolo, A. Sattar, M. Cristani, Computing private international law, Frontiers in Artificial Intelligence and App. (2021). doi:10.3233/FAIA210334.
[21] A. Malerba, A. Rotolo, G. Governatori, A logic for the interpretation of private international law, Logic, Argumentation and Reasoning (2022). doi:10.1007/978-3-030-70084-3_7.
[22] F. Olivieri, G. Governatori, M. Cristani, A. Rotolo, A. Sattar, Deontic meta-rules, Journal of Logic and Computation 34 (2024) 261 – 314. doi:10.1093/logcom/exac081.
[23] A. Rotolo, G. Sartor, Logical models for private international law, 2024. doi:10.1093/oso/9780192858771.003.0006.
[24] L. Pasetto, M. Cristani, G. Governatori, F. Olivieri, E. Zorzi, Extraction of defeasible proofs as explanations, in: CEUR Workshop Proceedings, 2023. Vol. 3546.
[25] S. Shapiro, M. Wilk, H. Chen, A comparative study of various tests for normality, J. of the American Statistical Association 63 (1968) 1343 – 1372. doi:10.1080/01621459.1968.10480932.