Towards Transparent Legal Formalization

Tomer Libal[1,2] and Tereza Novotná[3(✉)]

[1] American University of Paris, Paris, France
tomer.libal@uni.lu
[2] University of Luxembourg, Esch-sur-Alzette, Luxembourg
[3] Masaryk University, Brno, Czech Republic
tereza.novotna@law.muni.cz

Abstract. A key challenge in making a transparent formalization of a legal text is the dependency on two domain experts. While a legal expert is needed in order to interpret the legal text, a logician or a programmer is needed for encoding it into a program or a formula. Various existing methods are trying to solve this challenge by improving or automating the communication between the two experts. In this paper, we follow a different direction and attempt to eliminate the dependency on the target domain expert. This is achieved by inverting the translation back into the original text. By skipping over the logical translation, a legal expert can now both interpret and evaluate a translation.

Keywords: Legal knowledge base · Annotation editor · Formal representation

1 Introduction

Machine legal reasoning has been around for more than 30 years and various implementations have been developed [3,11,14]. Nevertheless, the authors consider it safe to claim that legal machine reasoning is still not widely used, despite the effort invested.

One of the most challenging tasks towards automated legal reasoning is the ability to encode legislation in a machine readable form [13]. In this step, a legislation written by law and policy makers needs to be converted into a formal representation, which can then be read and analyzed by computer programs.

Among the difficulties in obtaining a perfect translation, one can mention:

1. The need to translate a possibly ambiguous legal text into an unambiguous formal representation.
2. The need to decide on, usually, a specific interpretation of various legal terms, which are left open in the legislation.
3. The fact that two separate domain experts are needed, one for the target, formal, domain, and a legal expert capable of interpreting the legislation.

The first two difficulties are hard to overcome, although there is some work in that direction [15]. On the other hand, there are more than a few approaches and solutions to the third problem.

D. Calvaresi et al. (Eds.): EXTRAAMAS 2021, LNAI 12688, pp. 296–313, 2021.
https://doi.org/10.1007/978-3-030-82017-6_18

One way to bridge the gap between the two domain experts is to facilitate communications between them. In [2], the problem of bridging between two domain experts is likened to the software engineering problem of bridging between client requirements and software. The agile methodology in software engineering suggests that short iterations and frequent consultation with the client can help to close the gap. The methodology described in this paper aims at translating an intermediate legislation back into text, so the legal expert can communicate problems back to the logician.

A second way to bridge the gap is described in [9] and is based on the same legislation editor used in this paper. This methodology, which is based on another Agile technique called Behavior-driven development[1], supports dividing the work between the two domain experts. The legal expert task is to write legal "cases" (users stories in Agile), while the logician is still in charge of the legal formalization process. The interaction between the two is achieved by executing the cases against the formalized legislation. If a case fails, then the logician must fix the formalization.

Another way is to introduce a formalization language which is closely related to the legal text. A popular such language is SBVR [1]. While such languages greatly facilitate the formalization effort [4], they are usually not expressive enough to capture the semantics of legal texts, as is attested by the contiguous search for more expressive languages. Several tools for multi-agent normative reasoning exists, for example Operetta for constructing normative specifications[2], however our tool is different mainly because of the interaction with the user.

Our solution is different and is based on taking out from the equation one of the experts. By building on top of a previous work [5], in which a one-to-one mapping between a legislation and its translation was established, we hope that a legal expert can execute both the formalization and the evaluation steps.

The work in [5] was concerned with the question of "How similar is the translation to the original legislation?. Existing translations, such as [12] and [14], are normally very different from the legislation. For example, to obtain the translation of a sentence of the form (part of the Regulation (EU) 2016/679 - General Data Protection Regulation, hereinafter as "GDPR", Article 13 par. 1):

> the controller shall, at the time when personal data are obtained, provide the data subject with all of the following information:
> a) the identity and the contact details of the controller and, where applicable, of the controller's representative;
> b) the contact details of the data protection officer, where applicable;"

Current approaches will generate two or three different statements, each containing parts of the general conditions and adding new conditions and conclusion. On the other hand, the extension in [5] to the annotation language described in [8] enables a one-to-one mapping between the original legislation and its translation.

[1] https://www.agilealliance.org/glossary/bdd/.
[2] http://www.cs.uu.nl/research/projects/opera/.

In this paper, we build on the fact that there is a one-to-one mapping between the legislation and its translation and translate it back into a version of the original legislation. This last translation also enjoys a one-to-one mapping, which provides us with the main advantage of the method. The new methodology requires only a legal expert for asserting the quality of the translation. The expert is provided with the input and output of each paragraph and can determine if the translated formalization faithfully represents the specific legal interpretation of the input text, thus enhancing the transparency of the legal knowledge base.

This paper is organized as follows. In the next section, we present the editor and describe the additional features which were implemented in order to support the results of this paper. The third section is dedicated to describing the methodology and its application to article 7 of the GDPR. We conclude with an overview of future work and improvements.

2 The Legislation Editor

This section is adapted from [8].

The legislation editor integrates theorem proving technology into a usable graphical user interface (GUI) for the computer-assisted formalization of legal texts and applying automated normative reasoning procedures on these artifacts. In particular, the system includes

1. a legislation editor that graphically supports the formalization of legal texts via the use of annotations,
2. means of assessing the quality of entered formalizations, e.g., by automatically conducting consistency checks and assessing logical independence,
3. ready-to-use theorem prover technology for evaluating user-specified queries wrt. a given formalization, and
4. the possibility to share and collaborate, and to experiment with different formalizations and underlying logics.

The system is realized using a web-based Software-as-a-service and is available using a browser. It comprises a GUI that is implemented as a Javascript browser application, and a NodeJS application on the back-end side which connects to theorem provers, data storage services and relevant middleware. Using this architectural layout, no further software is required from the user perspective for using the editor and its reasoning procedures, as all necessary software is made available on the back end and the computationally heavy tasks are executed on the remote servers only. The results of the different reasoning procedures are sent back to the GUI and displayed to the user.

2.1 The Annotation Editor

The annotation editor allows users to create formalizations of legal documents that can be subsequently used for formal legal reasoning. The general functionality of the editor is described in the following.

One of the main ideas of the editor is to hide the underlying logical details and technical reasoning input and outputs from the user. We consider this essential, as the primary target audience of the tool are legal experts who are not necessarily logicians. It could greatly decrease the usability of the tool if a solid knowledge about formal logic was required. This is realized by letting the user annotate legal texts and queries graphically and by allowing the user to access the different reasoning functionalities by simply clicking buttons that are integrated into the GUI.

Another main idea of the editor is to enable a direct annotation of the original text, without the need to modify it. This property is essential to the core result of this paper - the ability to regenerate the original text from the annotations, which can then be compared. It should be noted that many times, legal texts contain implicit information. For example, the GDPR often talks about the processing of data, but without always specifying that this is done by a processor. In such cases, the users need to add the implicit text explicitly in the editor, so it can be annotated. We encourage the user to put such text in square brackets, in order to simplify the comparison between the original and the generated text at the end of the process.

These two properties raise the need for an expressive annotation language, which can capture not only language properties but also meta-language properties, such as exceptions and other complex relations. In order to support such a language the annotations are organized into three layers.

In order to demonstrate the different elements of the editor, we will use GDPR article 7, paragraph 2 as an example. The full annotation of this paragraph can be seen in Fig. 1.

The editor employs a hierarchical approach to annotations. At the base, there are "term" annotations, which are used to mark all relevant entities and relations. These annotations are assigned specific colors, corresponding to their entities. Entities normally denote a class of items or people while relations denote actions or relations between different entities. An example of an entity which can be seen in Fig. 1 is "give_consent", which corresponds to a relation between a data subject, a controller and a specific data, processing and time.

On top of the term annotations, there are logical and normative properties, such as obligations, conditions and negations. These annotations place a specific context around term annotations, as well as other logical and normative ones. For example, a term annotation might imply another, in which case the two relate to each other as a condition and implication. If an annotation is an obligation, such an annotation should be applied. In Fig. 1, we can see that the relation between the first two terms and the following five is a condition in which the conclusion is an obligation. This is obtained by setting each group of the two groups of terms as a conjunction and setting the relationship between the two groups as a conditional obligation, where the first part consists of the conditions and the second consists of the obligation.

Lastly, the editor enables the user to annotate concepts which go beyond direct logical and normative properties. For example, a sentence might be an

exception of another, in which case the two must be bound together, despite the fact that they may not appear together. Another example is when a sentence refers to another but requires the replacement of some concepts with others. The current text requires but a simple use of this feature. In Fig. 1, we can see that each paragraph is being associated with its specific numbering using the "Labeling" annotation. Such labels can be used later, for example in exceptions. More complex examples can be found in the formalization of article 13 of the GDPR, which is described in [6].

The formalization proceeds as follows: The user selects some text from the legal document and annotates it, either as a term or as a composite (complex) statement. In the first case, a name for that term is computed automatically, but it can also be chosen freely. Different terms are displayed as different colors in the text. In the latter case, the user needs to choose among the different possibilities, which correspond to either logical connectives or higher-level sentence structures called macros. The composite annotations are displayed as a box around the text and can be done recursively.

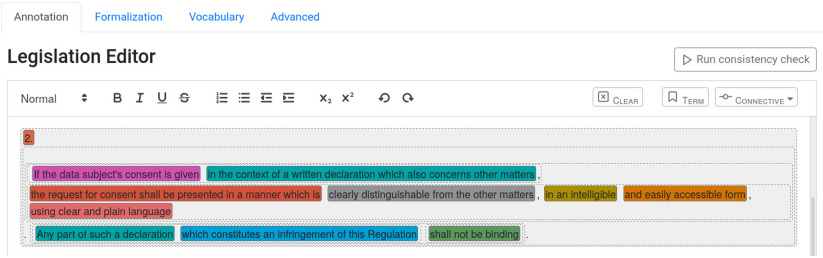

Fig. 1. Article 7, par 2: full annotation

The editor also features direct access to the consistency check and logical independence check procedures (as buttons). When such a button is clicked, the current state of the formalization will be translated and sent to the back-end provers, which determine whether it is consistent resp. logically independent.

User queries are also created using such an editor. In addition to the steps sketched above, users may declare a text passage as *goal* using a dedicated annotation button, whose contents are again annotated as usual. If the query is executed, the back-end provers will try to prove (or refute) that the goal logically follows from the remaining annotations and the underlying legislation. This way, the tool can answer YES/NO questions on whether the goal logically follows the facts and logical relations of formalized legislation.

A very important feature of the editor is the ability to check the formalization. Often times, formalizations cannot be checked for correctness and can contain, therefore, many errors and typos[3]. By clicking the "Save" button, the

[3] Please refer to [5] for more information and examples.

annotations are being converted into a formal structure, which is then being converted into a legal text, for comparison. This process fails if the structure of the annotations is incorrect and serves as a kind of "compiler".

In the next section, we are going to discuss in more detail the generated legal text, which can be found under the "Formalization" tab.

3 Legal Formalization

Formalization of legal text into a machine readable format is a rather complex process. In such a process, the user inevitably encounters a number of problems as described in Sect. 1. In previous research, the authors tackled a few such difficulties by proposing an approach of interactive legal text formalization, where a user or legal expert decides about some of the types of difficulties based on her expertise in certain legal field [8]. An interpretation of different statements and terms in a legislation or legal text is generally one of the biggest problems during automatic formalization. Authors deal with the interpretation issue in such a manner that legal experts as users solely interpret the legal text being formalized. Using this approach the editing tool becomes more of a support tool while formalizing legal text and only depends on the decisions of the user. This approach is further advantageous because it can flexibly react on the development of interpretation of a certain legal text in time.

In this article, the authors develop this approach further and tackle another relevant problem. As described in Sect. 1, legal formalization using an editing tool requires knowledge of both law and logic which in most cases requires the cooperation of a logician and a lawyer. The authors propose a solution in providing user-friendly output of a formalization of legal text which is easily comprehensible for lawyers with basic knowledge of logic. This approach aims to reduce the level of expertise in logic required and assumes that the legal expert is able to both formalize the legislation and assess the accuracy of the output.

3.1 Article 7 of GDPR

We use GDPR as a representative legislation for the editing tool functioning presentation. GDPR is one of the most discussed and in many aspects controversial European legislations, which makes it an optimal use case for the purpose of this study. The GDPR defines the principles of personal data processing and the conditions of lawfulness of their processing. It also regulates the conditions for expressing the consent given to the processing of data and the provision of information and access to personal data. The GDPR is universally binding and applicable in all Member States of EU. Regarding its universality, uniqueness and relative strictness in duties and sanctions, it is not surprising that a great deal of effort is being made to ensure compliance of personal data processing with the regulation in the private sector and to analyze and interpret the wording of the legislation in the public and academic sectors. This statement may be also

supported by the rich case law concerning GDPR of both the general and European courts and other administrative institutions. The robustness of the GDPR application then creates the need to simplify and make the legal document ideally accessible to a wide range of citizens and institutions in a transparent and simple form.

In a current state, the user is capable of formalizing the whole legislation, which is the recommended way of using the editing tool. Regarding the scope of this article, we decided to present the formalization of the whole of Article 7 of GDPR - *Conditions for consent*. For the same reason we do not formalize the Article in context with the rest of the legislation, however this systematic formalization could be done following the same methodology.

Consent, in the context of GDPR, is one of the lawful ways of processing personal data in accordance with Article 6 par. 1 (a) of GDPR and it is one of the most common ways to lawfully process personal data as it is part of a majority of services a human can use or buy nowadays. Article 7 states necessary conditions for the consent of a subject of personal data to be lawful and in accordance with the European regulation.

The full wording of Article 7 GDPR is:

1. Where processing is based on consent, the controller shall be able to demonstrate that the data subject has consented to processing of his or her personal data.

2. If the data subject's consent is given in the context of a written declaration which also concerns other matters, the request for consent shall be presented in a manner which is clearly distinguishable from the other matters, in an intelligible and easily accessible form, using clear and plain language. Any part of such a declaration which constitutes an infringement of this Regulation shall not be binding.

3. The data subject shall have the right to withdraw his or her consent at any time. The withdrawal of consent shall not affect the lawfulness of processing based on consent before its withdrawal. Prior to giving consent, the data subject shall be informed thereof. It shall be as easy to withdraw as to give consent.

4. When assessing whether consent is freely given, utmost account shall be taken of whether, inter alia, the performance of a contract, including the provision of a service, is conditional on consent to the processing of personal data that is not necessary for the performance of that contract.

3.2 Formalization of Article 7 Using the Legislation Editor

The formalization of legal text means the transformation of the text, including its semantics, into the logical representation. The legislation editor uses the formalization performed by the user - legal expert - as an input. The formalization is performed in the editor using annotations of the text as described in Sect. 2.1 in the Annotation tab.

The formalization in the legislation editor is based on the annotations of three categories: terms, connectives and macro concepts. Terms are parts of the text that the tool recognizes as an entity, entities can appear in the formalization

any number of times. Connectives are logical constructs aimed to catch the logical relationship among terms. Terms are marked as colored rectangles and connectives and macros as grey borders in Fig. 1, the same term is always colored with the same color. Annotations are performed by the expert who marks the text according to her interpretation of the text and regarding the goal of the formalization - what kind of questions about the legal text the user wants to answer.

As it was mentioned in Sect. 1, the annotations methodology follows hierarchical order. It is recommended to firstly determine the legal text or parts of the legal text which user wants to formalize and then to clarify the logical relationships between the parts of the text. Given the example of Article 7 GDPR, this article contains 4 paragraphs, all paragraphs relate to the conditions of consent, which is gradually refined, therefore it is possible to start the formalization of individual paragraphs separately.

Subsequent formalization shall follow a similar trend - firstly determine all the terms (entities) in the text which the user considers important and mark them as terms in the Annotation tab. After this step, the user shall mark the logical relationships among marked terms. During this step, it is again necessary to annotate the relations hierarchically, i.e. to start with external relationships and gradually move on to internal ones.

Given Article 7 par. 1: *1. Where processing is based on consent, the controller shall be able to demonstrate that the data subject has consented to processing of his or her personal data.*

This paragraph contains an obligation of a controller to demonstrate the consent of a specific subject for the processing of his or her personal data if the processing is based on consent. The essential entities of this paragraph are 6 following: processing, based on consent, controller, to demonstrate, data subject and consent to processing of personal data. Therefore, we accept these 6 parts of the sentence as terms bearing the meaning of the sentence. We use the legislation editor annotations scheme to mark all these terms and choose the term names and variables. Variables follow first-order logic methodology and they are contained in the brackets behind the term names. They need to start with upper case letter to be recognized as variables. Variables allow the user to address specific real-life subjects and objects and reason about them. Let us take the first term *processing*. While processing, there is a specific subject and her personal data to be processed by a specific controller in a specific situation in time. All these characteristics of processing are changeable depending on the specific situation, therefore we accept them as variables. When annotating in the Annotation tab of the tool, we mark first part of the sentence "*Where processing*" as a term using the "Term" button as in Fig. 1 and choose the name **processing(Data, Subject, Time, Controller)**, where the order of variables is arbitrary, as in Fig. 2. We annotate the other 5 terms in the same manner with the variables as following:

"*is based on consent*" as term **based_on_consent(Data, Subject, Time, Controller, Consent, Processing)**

"*the controller*" as term **controller(Controller)**

"*shall be able to demonstrate*" as term **to_demonstrate(Controller, Consent)**

"*that the data subject*" as term **data_subject(Subject)** "*has consented to processing of his or her personal data*" as term **give_consent(Data, Subject, Time, Controller, Consent, Processing)**

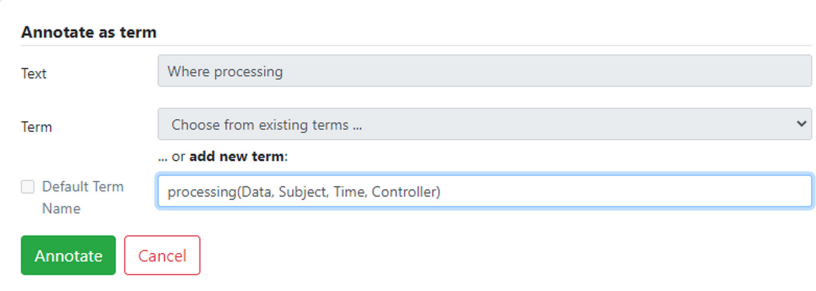

Fig. 2. Annotation of a "Term"

Additionally, we annotate the number of paragraph "1." as a label for this paragraph using the "Term" button and choosing the name **paragraph1**. We do this step in order to be able to use the sentence as a whole later in the situation, where it is needed to state a specific logical relationship between different paragraphs. Therefore, there is no need to use variables for this term, since we are choosing just a name for the paragraph.

"*1.*" as term **paragraph1**

This way we divided the first paragraph of Article 7 into different parts marked as terms in the Annotation tab, all of the marked terms bearing different meaning and having different roles in the sentence. The second step in formalization using the legislation editor is to define the logical relationships between the terms in legal text.

As it was mentioned beforehand, during the annotation of logical relationships, it is necessary to proceed from external logical relationships to internal ones. Practically, annotation of logical relationships is similar to annotation of the terms. We have to mark the whole text to which we intend to assign a logical relationship and choose the correct logical relationship from the "Connectives" button, which provides several options as we can see in Fig. 3.

First, we have to use the "Labelling" of the paragraph, which is a macro concept listed in "Connectives" as "Labelling sentences". This macro concept accepts the first term in the marked text as a label and the rest as a sentence to be labelled. In this case, the first marked term is the term **paragraph1**. This means that if we need to refer to this paragraph in the future while formalizing of

the regulation, we can use the term **paragraph1** as a reference to this paragraph. Labelling is a usual first step as it delimits the part of a text we will formalize as a whole.

Fig. 3. The list of connectives

Secondly, we have to decide on the logical relationship of a paragraph (sentence). This task requires knowledge of legal notions and legal language used for expressing different legal statements. In the case of the first paragraph of Article 7, this sentence expresses an obligation of the controller to demonstrate the consent, however this obligation is conditioned by the first part of sentence, according to which the processing must be based on a consent for the following obligation to apply. We can deduce that the obligation of the controller in the second part of the paragraph is conditioned by the first part of the paragraph. This situation clearly leads to an implication of obligation, in the "Connectives" list in Fig. 3 marked as "If/Then Obligation". Therefore, we mark the whole sentence, except the label **paragraph1** as "If/Then Obligation".

The first two terms - **processing(Data, Subject, Time, Controller)** and **based_on_consent(Data, Subject, Time, Controller, Consent, Processing)** are conditions for the obligation of the controller to apply. Thus, we mark these two terms with another internal logical connective "And" (as a conjunction).

The obligation of the controller is expressed in rest of the terms in this paragraph - **controller(Controller)**, **to_demonstrate(Controller, Consent)**,

and **data_subject(Subject)** and **give_consent(Data, Subject, Time, Controller, Consent, Processing)**. Similarly, we mark these terms again with the internal logical connective "And".

To summarize this process, we can say that the controller is legally obligated to demonstrate a consent expressed by the data subject for specific processing of her personal data, if such processing is based on consent. Full obligation of Article 7 paragraph 1 is in Fig. 4.

Fig. 4. Annotation of Article 7 par. 1

The legislation editor contains another two important buttons - "Save" and "Run consistency check". We save the formalization of the first paragraph with the "Save" button and we check whether our formalization is correct by pressing the "Run consistency check" button as in Fig. 4. This feature of the legislation editor was already described in previous work of the authors, however the issue tackled in this article is the following: even though the consistency check is correct, is the user (lawyer) capable of assessing her formalization and deciding whether it is correct and meaningful from both a legal and logical perspective?

To tackle this issue and simplify this task, we propose a work-in-progress solution in the Formalization tab in the Legislation Editor.

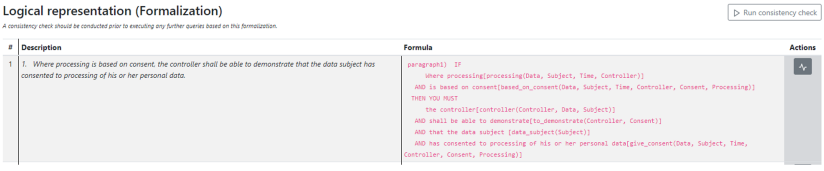

Fig. 5. Formalization of Article 7 par. 1

The Formalization tab in Fig. 5 offers comparison of the original text and logical formulae applied to the original text. Furthermore, logical formulae are provided in such a way that the user can easily understand the logical relationships and evaluate their correctness. The current presentation displays the internal logic formulae next to the relevant text. These formulae are denoted in first-order Deontic logic [7] and variables are implicitly quantified over each of the whole formulae displayed on the formalization tab. In future versions, this logical terms will be replaced by their matching entities in a relevant ontology (see for example, [10]).

It should also be noted that we make a distinction between the formula used for representing the knowledge, and the formula used for actual reasoning over the knowledge. Once a theorem prover is being called, the representation formula is being translated into the prover input logic. Issues such as conditional obligations, negation-as-failure and others are resolved at this point, depending on the target theorem prover.

The goal of the methodology of translating the original text into a formula and then back into text, is that the output is as comprehensible and as similar as the input, i.e. the original legal text in the left part of the Formalization tab. At the same time, it is necessary to preserve the logical formulae used while formalizating the text and present them in the output in a compatible way with the text. To find the right balance of these two goals - to present both the original text and logical formulae comprehensibly together - is the future work for the authors of this article, the current setting is just the first attempt on this path.

As we can see in Fig. 5, first the legislation editor shows the label of the paragraph. Subsequently, it shows the specific part of the original legal text and the term allocated to this specific part of the text behind together with selected variables. To clearly distinguish different semantic parts of texts, it lists the terms on separate rows in the order as in the original text, therefore it preserves the sequence of the original legal text. Additionally, different terms (different parts of original text) are connected with logical words representing different logical relationships.

Specifically, we can first observe the word "IF" which means that the following terms are conditions. These following terms are connected with "AND". The obligation dependent on previous conditions is marked as "THEN YOU MUST" and the following terms are the consequences that need to happen in order to meet the obligation of this part of the regulation. The legal consequences are again connected with "AND" to show that all of them need to apply. Detailed formalization of the first paragraph is in Fig. 6.

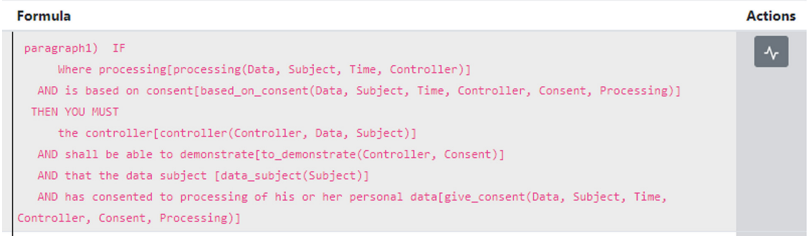

Fig. 6. Formalization of Article 7 par. 1 - formulae

In this case, the comprehensibility is achieved through the indentation, separation of the original text and allocated terms and differentiation of connectives in upper case and terms or original text in lower case. The visual distinction of logical parts of legal text is another way of presenting the output, although we

are aware that in this direction, the legislation editor is limited and we accept this as a future work issue. Nevertheless, we believe that our approach is the right first step in the comprehensibility of formalization of legal text.

Regarding paragraph 1 of Article 7, the output is easily comparable with the input in the left part of the Formalization tab. The check of formalization is then easily performed by simply reading through the right formalized output and assessing its legal meaning when comparing to the meaning of the original text.

Given this formalization methodology, we can proceed with the rest of the paragraphs in the same manner. In paragraph 2 as we can see in Fig. 1 we used following terms:

"*2.*" as term **paragraph2**

"*If the data subject's consent is given*" as term **give_consent(Data, Subject, Time, Controller, Consent, Processing)**

"*in the context of a written declaration which also concerns other matters*" as term **written_declaration_other_matters(Data, Subject, Time, Controller, Consent)**

"*the request for consent shall be presented in a manner which is*" as term **request_for_consent(Data, Subject, Time, Controller, Consent)**

"*clearly distinguishable from the other matters*" as term **distinguishable(Data, Subject, Time, Controller, Consent)**

"*in an intelligible*" as term **intelligible(Data, Subject, Time, Controller, Consent)**

"*and easily accessible form*" as term **accessible(Data, Subject, Time, Controller, Consent)**

"*using clear and plain language*" as term **language(Data, Subject, Time, Controller, Consent)**

"*Any part of such a declaration*" as term **written_declaration_other_matters (Data, Subject, Time, Controller, Consent)**

"*which constitutes an infringement of this Regulation*" as term **infringement**

"*shall not be binding*" as term **binding**

Subsequently, after labelling the paragraph, we chose two obligations based on conditions as connectives for the two separate sentences in paragraph 2. Therefore, we annotated them separately as "IF/THEN OBLIGATION" connectives, where **give_consent(Data, Subject, Time, Controller, Consent, Processing)** and **written_declaration_other_matters(Data, Subject, Time, Controller, Consent)** are conditions for first implication and the following terms are consequences. **written_declaration_other_matters(Data, Subject, Time, Controller, Consent)** and **infringement** are conditions for second implication and **binding** (annotated as a "NEGATION" first) is a consequence.

The resulting formalization is in Fig. 7.

```
paragraph2)     IF
        If the data subject's consent is given[give_consent(Data, Subject, Time, Controller, Consent,
Processing)]
    AND in the context of a written declaration which also concerns other
matters[written_declaration_other_matters(Data, Subject, Time, Controller, Consent)]
   THEN YOU MUST
        the request for consent shall be presented in a manner which is[request_for_consent(Data,
Subject, Time, Controller, Consent)]
    AND clearly distinguishable from the other matters[distinguishable(Data, Subject, Time, Controller,
Consent)]
    AND in an intelligible[intelligible(Data, Subject, Time, Controller, Consent)]
    AND and easily accessible form[accessible(Data, Subject, Time, Controller, Consent)]
    AND using clear and plain language[language(Data, Subject, Time, Controller, Consent)]

  AND    IF
        Any part of such a declaration[written_declaration_other_matters(Data, Subject, Time, Controller,
Consent)]
    AND which constitutes an infringement of this Regulation[infringement]
   THEN YOU MUST
        NOT
     shall not be binding[binding]
```

Fig. 7. Formalization of Article 7 par. 2 - formulae

Again, we can observe two separated implications with obligations starting with "IF" statement and continuing with "THEN YOU MUST". These two implications (sentences) are connected with the connective "AND" as two parts of one paragraph labelled as "paragraph2". Given the scope of this article, we are not able to go through the formalization of all the paragraphs step by step, however we are providing the outputs in Fig. 8 and Fig. 9.

```
paragraph3)     YOU ARE ALLOWED
        The data subject[data_subject(Subject)]
    AND shall have the right to withdraw his or her consent at any time[withdraw_consent(Data, Subject,
Time, Controller, Consent, Processing)]
  AND    IF
    The withdrawal of consent[withdraw_consent(Data, Subject, Time, Controller, Consent, Processing)]
   THEN YOU MUST
        shall not affect the lawfulness of processing[processing(Data, Subject, Time, Controller)]
    AND based on consent before its withdrawal[based_on_consent(Data, Subject, Time, Controller, Consent,
Processing)]

  AND    YOU MUST
        Prior to giving consent[give_consent(Data, Subject, Time, Controller, Consent, Processing)]
    AND the data subject[data_subject(Subject)]
    AND shall be informed thereof[inform(Data, Subject, Time, Controller, Consent)]
  AND    YOU MUST
        It shall be as easy[easy(Data, Subject, Time, Controller, Consent, Processing)]
    AND to withdraw as to give consent[withdraw_consent(Data, Subject, Time, Controller, Consent,
Processing)]
```

Fig. 8. Formalization of Article 7 par. 3 - formulae

We used a permission to formalize the first sentence of paragraph 3 as a right of subject of personal data to withdraw the consent. This permission is

```
paragraph4)  IF
     When assessing whether consent[assessing(Data, Subject, Time, Controller, Consent)]
  AND is freely given[freely_given(Data, Subject, Time, Controller, Consent)]
 THEN YOU MUST
     utmost account shall be taken of [take_account]
  AND whether, inter alia, the performance of a contract, including the provision of a
service[performace_of_contract(Data, Subject, Time, Controller, Consent)]
  AND is conditional[conditional(Data, Subject, Time, Controller, Consent, Processing)]
  AND on consent to the processing of personal data[give_consent(Data, Subject, Time, Controller,
Consent, Processing)]
  AND     NOT
    that is not necessary[necessary(Data, Subject, Time, Controller, Consent, Processing)]
  AND for the performance of that contract[performace_of_contract(Data, Subject, Time, Controller,
Consent)]
```

Fig. 9. Formalization of Article 7 par. 4 - formulae

simple, it is not conditioned by any other facts or actions. Similarly, we used simple obligations without implications for the third and fourth sentences in this paragraph given that these sentences mean simple obligations non-conditioned by any precedent facts or actions to apply.

Regarding paragraph 4, we annotated the whole sentence as an obligation with the implication "IF/THEN OBLIGATION" given that "*whether assessing if the consent is freely given*" are conditions and considering the necessity of processing of personal data for the performance of a contract are consequences that must be met to comply with the GDPR regulation.

3.3 The Comprehensibility of Formalization - Future Work

It is evident that our attempt to provide a meaningful and comprehensible formalization tool for lawyers to be used without extensive knowledge of first-order logic and logical reasoning is just a first step to achieve this rather ambitious task. Our new feature of the legislation editor in the current setting offers only limited visualization help for lawyers. This feature is based mainly on visual indentation of different parts of texts (terms and connectives) and hierarchical sequence of parts of the text. Nevertheless, we assume that the most important lesson learned is that using as much of the original legal text as possible to present in the output formulae, together with logical connectives, increases the comprehensibility significantly. And mainly, it allows the user to compare the original legal text with the formalized text in logical formulae which makes the formalization task less time-consuming and more user-friendly. Our legislation editor can do so because it stores the parts of original text as "Terms" and although it continues to work only with the annotated terms (or labels), the stored text is very helpful when the tool presents the output of the formalization to the user.

As a future work, we intend to achieve higher comprehensibility with better visualization of the output formulae using graphical features such as colours or diagrams and connecting the formalized terms to the ontological dictionary. We believe that enhancing the comprehensibility of output formulae is the main task

to be solved on the way to usable formalization tool for lawyers without further knowledge of advanced logic.

4 Conclusion

One of the major obstacles for the creation of a formal legal knowledge base, is the dependency on both target and source domain experts. The target domain experts, who are usually logicians or programmers, are most suitable for encoding knowledge into formulae and programs, but they usually lack a deep understanding in the legal domain. On the other hand, most legal experts would find it difficult to follow the legal meaning of a legislation in a completely formal form. This dependency on two domain experts means that the trust legal experts can place in a knowledge base is limited.

In this article, we build on our previous research ideas and applications in legal reasoning and formalization of legal texts in order to make the formalization process transparent. This is achieved by increasing the confidence of the legal expert in the formalized knowledge.

Our approach is based on an assumption that if the formalization of legal text is user-friendly enough and if the output of formalization is comprehensible enough, the formalization of legal text can be performed by an expert in specific legal domain without a deeper knowledge of first-order logic. We are trying to achieve this goal by extending an existing legislation editor with a visualized output of a formalization.

Using our approach, a lawyer with limited knowledge of first-order logic is capable of translating certain legal text into formal language using a simple annotation editor according to his expertise in a legal domain related to this legal text. The formalization is only dependent on his expert interpretation of the text. Furthermore, the lawyer is capable of properly checking the formalized formulae of legal text and eventually, to correct the formalization according to his best knowledge of interpretation and legal domain.

To demonstrate this approach, we formalize Article 7 of GDPR. We show the practical use of the legislation editor in the formalization of Article 7 of GDPR in a step by step manner. Subsequently, we show the visualized output of the logical formulae as a result of legal text formalization. We argue that the proposed design of the output formulae is sufficient and comprehensible enough for a lawyer to assess the logical formulae and compare it to the original plain legal text and potentially use it to correct the formalization.

This paper describes a very basic prototype with the sole purpose of demonstrating the approach. In the future, we plan on enhancing the translation back into legal text so it will be as similar to the original text as possible. For example, a legal text might use various words to denote an obligation, such as “must”, “should” and “obliged”. All these verbs are currently uniformly being translated into “It is an obligation that”. By parsing the original text and extracting the specific “obligation” verb, we can increase the similarity of the generated text to the original.

Another enhancement to the current version would be to create a two-dimensional display of the legal text. Currently, both the original text and the generated formula which denote a certain term are appearing in the translation. In a two-dimensional approach, only the original text would be displayed and the user would have the possibility to see which ontological entity is mapped to it by clicking on or hovering over it.

Acknowledgment. Tereza Novotn acknowledges the support of the ERDF project "Internal grant agency of Masaryk University" (No. CZ.02.2.69/0.0/0.0/19_\0073/0016943).

References

1. Bajwa, I.S., Lee, M.G., Bordbar, B.: SBVR business rules generation from natural language specification. In: 2011 AAAI Spring Symposium Series. Citeseer (2011)
2. Bartolini, C., Lenzini, G., Santos, C.: An agile approach to validate a formal representation of the GDPR. In: Kojima, K., Sakamoto, M., Mineshima, K., Satoh, K. (eds.) JSAI-isAI 2018. LNCS (LNAI), vol. 11717, pp. 160–176. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-31605-1_13
3. Governatori, G., Shek, S.: Regorous: a business process compliance checker. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law, pp. 245–246 (2013)
4. Johnsen, Å., Berre, A.: A bridge between legislator and technologist-formalization in SBVR for improved quality and understanding of legal rules. In: International Workshop on Business Models, Business Rules and Ontologies, Bressanone, Brixen, Italy. Citeseer (2010)
5. Libal, T.: A meta-level annotation language for legal texts. In: Dastani, M., Dong, H., van der Torre, L. (eds.) CLAR 2020. LNCS (LNAI), vol. 12061, pp. 131–150. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44638-3_9
6. Libal, T.: Towards automated GDPR compliance checking. In: Proceedings of the Workshop on the Scientific Foundations of Trustworthy AI, Integrating Learning, Optimisation and Reasoning (to appear)
7. Libal, T., Pascucci, M.: Automated reasoning in normative detachment structures with ideal conditions. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law, pp. 63–72 (2019)
8. Libal, T., Steen, A.: NAI: the normative reasoner. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law, pp. 262–263 (2019)
9. Libal, T., Steen, A.: Towards an executable methodology for the formalization of legal texts. In: Dastani, M., Dong, H., van der Torre, L. (eds.) CLAR 2020. LNCS (LNAI), vol. 12061, pp. 151–165. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44638-3_10
10. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53
11. Prakken, H., Wyner, A., Bench-Capon, T., Atkinson, K.: A formalization of argumentation schemes for legal case-based reasoning in ASPIC+. J. Logic Comput. **25**(5), 1141–1166 (2015)

12. Robaldo, L., Bartolini, C., Palmirani, M., Rossi, A., Martoni, M., Lenzini, G.: Formalizing GDPR provisions in reified I/O logic: the DAPRECO knowledge base. J. Logic Lang. Inf. **29**(4), 401–449 (2020)
13. Routen, T., Bench-Capon, T.: Hierarchical formalizations. Int. J. Man Mach. Stud. **35**(1), 69–93 (1991)
14. Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T.: The British nationality act as a logic program. Commun. ACM **29**(5), 370–386 (1986)
15. Vitali, F., Zeni, F.: Towards a country-independent data format: the Akoma Ntoso experience. In: Proceedings of the V Legislative XML Workshop, pp. 67–86 (2007)