Towards Automated GDPR Compliance Checking

Tomer Libal[1] [0000−0003−3261−0180]

Luxembourg University `tomer.libal@uni.lu`

Abstract. The GDPR is one of many legal texts which can greatly benefit from the support of automated reasoning. Since its introduction, efforts were made to formalize it in order to support various automated operations. Nevertheless, widespread and efficient automated reasoning over the GDPR has not yet been achieved. In this paper, a tool called the NAI suite is being used in order to annotate article 13 of the GDPR. The annotation results in a fully formalized version of the article, which deals with the transparency requirements of data collection and processing. Automated compliance checking is then being demonstrated via several simple queries. By using the public API of the NAI suite, arbitrary tools can use this procedure to support trust management and GDPR compliance functions.

Keywords: Automated Reasoning · Compliance Checking · Data Protection

1 Introduction

Computer systems are playing a substantial role in assisting people in a wide range of tasks, including search in large data and decision-making; and their employment is progressively becoming vital in an increasing number of fields. One of these fields is legal reasoning: New court cases and legislations are accumulated every day and navigating through the vast amount of complex information is far from trivial. In addition, the understanding of those texts is reserved only for experts in the legal domain despite the fact that they are usually of interest to the general public.

This characteristic of the law goes beyond simply keeping the public well informed. In some domains, such as privacy and ethics of AI, there is a strong need to regulate and govern over a vast amount of applications and possible infringements. On the 8th of April this year (2019), an independent expert group on AI, set up by the European Commission (EC), has published ethical guidelines for trustworthy AI[1]. These guidelines, which respond to an earlier publication

[1] European Commission, Independent High-Level Expert Group on Artificial Intelligence. Ethics Guidelines for Thrustworthy AI. `https://ec.europa.eu/futurium/en/ai-alliance-consultation`, April 2019.

of the EC vision for AI[2], have identified three components which should be part of trustworthy AI systems - they should be lawful, ethical and robust.

There is an inherent problem though between these three components and the fact that laws and regulations require a human for their interpretation and for decision making. The number of AI applications grows far faster than the number of regulators and governors.

As an example, let us consider the General Data Protection Regulation (GDPR). This legal document aims at regulating data protection over a vast number of applications, among them, one can count almost all websites, but not only. The loss of privacy is of high interest to all of us. Nevertheless, the human regulators can cover only a fraction of possible infringements.

Despite the need for automation in the legal domain, the employment of automated legal reasoning tools is still underrepresented in practice [6] albeit being a relevant and active field of research since the 1980s [2, 22, 23]. Some examples of such tools and approaches are for courtroom management [1], legal language processing/management [4], for business compliance [9] and GDPR compliance [20].

None of the above approaches though has been applied to the GDPR or similar tasks. More information about the reasons behind that can be found in [6,8], In this paper, one of the recent tools in the domain, the NAI suite [13,14], is applied to article 13 of the GDPR. This article, which is concerned with the transparency of data collection and processing, is first formalized in a way such that its legal meaning is being captured by the machine. Automated reasoning over the formalization is then being demonstrated on several queries which can be used for compliance testing. When combined with the NAI suite public API, tools can be created which can take advantage of this automated compliance checking. Moreover, together with works on natural language processing of legal texts [17], fully automated lawful, ethical and robust trust management systems might become possible.

Discussion of related work. The recency of the GDPR means that it features in relatively few works on automation. Among them, the DAPRECO knowledge base [21] is the most extensive one. This knowledge base is a formalization of most articles of the GDPR into formulas of IO logic [15]. On the other hand, there are currently no known tools which allow for reasoning over it. In addition, this knowledge base contains some typos and semantical errors. A discussion about one of them is presented in Section 3.2.

A very recent modeling of articles 5 and 6 of the GDPR appears in [16]. This modeling allows automated reasoning via a Prolog interpreter. Using Prolog for the formalization of legal texts is very powerful and was applied to various examples (see for example [22]). Nevertheless, both the formalization process and the usage of this formalization are non trivial and require, if not logic programming skills, at least an understanding of first-order logic. In addition, the direct formalization into a low level logical language might give raise to errors, as is discussed in Section 3.2 in the context of the DAPRECO knowledge base.

[2] COM(2018)237 and COM(2018)795.

The paper has several contributions. It presents a full formalization of article 13 of the GDPR, which allows for efficient automated reasoning. Since the formalization in the NAI suite is freely available via a public and RESTFull API, it can be immediately integrated into third-party tools which require automation over this article. Second, it demonstrates how fully automated compliance checking can be achieved. Lastly, some errors in previous efforts to formalize the GDPR are discussed and corrected.

In the next section, an introduction to the NAI suite, its theoretical foundations and its graphical user interface is given. The following section describes the extension of the tool and its application to the formalization of article 13. Section 4 describes the support for automated reasoning and gives some examples of compliance checking. A conclusion and possible future work are given last.

2 The NAI suite

The NAI suite integrates novel 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, NAI includes

1. a legislation editor that graphically supports the formalization of legal texts,
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.

NAI is realized using a web-based Software-as-a-service architecture, cf. Fig. 1. 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 NAI 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. The major components of NAI are described in more detail in the following.

2.1 The Underlining Logic

The logical formalism underlying the NAI framework is based on a universal fragment first-order variant of the deontic logic **DL*** [12], denoted $\mathbf{DL}^*_1$. Its syntax is defined as follows. Let V, P and F be disjoint sets of symbols for

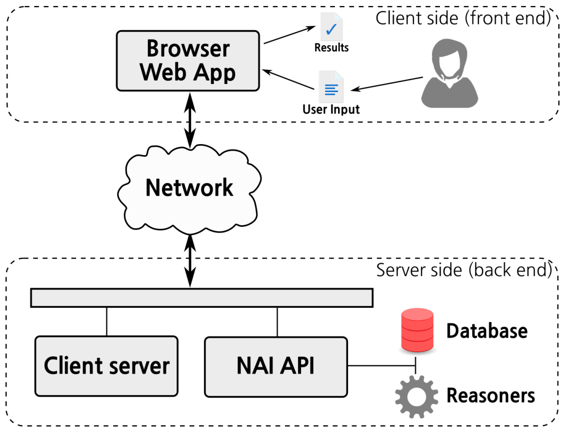


Fig. 1: Software-as-a-service architecture of the NAI reasoning framework. The front end software runs in the user's browser and connects to the remote site, and its different services, via a well-defined API through the network. Data flow is indicated by arrows.

variables, predicate symbols (of some arity) and function symbols (of some arity), respectively. **DL**$^*_1$ formulas ϕ, ψ are given by:

$$\begin{aligned}\phi, \psi ::= &\; p(t_1, \ldots, t_n) \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \Rightarrow \psi \\ &\mid \mathsf{Id}\,\phi \mid \mathsf{Ob}\,\phi \mid \mathsf{Pm}\,\phi \mid \mathsf{Fb}\,\phi \\ &\mid \phi \Rightarrow_{\mathsf{Ob}} \psi \mid \phi \Rightarrow_{\mathsf{Pm}} \psi \mid \phi \Rightarrow_{\mathsf{Fb}} \psi\end{aligned}$$

where $p \in P$ is a predicate symbol of arity $n \geq 0$ and the t_i, $1 \leq i \leq n$, are terms. Terms are freely generated by the function symbols from F and variables from V. ⌟

DL$^*_1$ extends Standard Deontic Logic (SDL) with the normative concepts of ideal and contrary-to-duty obligations, and contains predicate symbols, the standard logical connectives, and the normative operators of obligation (Ob), permission (Pm), prohibition (Fb), their conditional counter-parts, and ideality (Id). Free variables are implicitly universally quantified at top-level.

This logic is expressive enough to capture many interesting normative structures. For details on its expressivity and its semantics, we refer to previous work [12].

2.2 The Reasoning Module

The NAI suite supports formalizing legal texts and applying various logical operations on them. These operations include consistency checks (non-derivability of falsum), logical independence analysis as well as the creation of user queries that can automatically be assessed for (non-)validity. After formalization, the formal representation of the legal text is stored in a general and expressive machine-readable format in NAI. This format aims at generalizing from concrete logical formalisms that are used for evaluating the logical properties of the legal document's formal representation.

There exist many different logical formalisms that have been discussed for capturing normative reasoning and extensions of it. Since the discussion of such formalisms is still ongoing, and the choice of the concrete logic underlying the reasoning process strongly influences the results of all procedures, NAI uses a two-step procedure to employ automated reasoning tools. NAI stores only the general format, as mentioned above, as result of the formalization process. Once a user then chooses a certain logic for conducting the logical analysis, NAI will automatically translate the general format into the specific logic resp. the concrete input format of the employed automated reasoning system. Currently, NAI supports only the $\mathbf{DL}^{*}_1$ logic from §2.1; however, the architecture of NAI is designed in such a way that further formalisms can easily be supported.

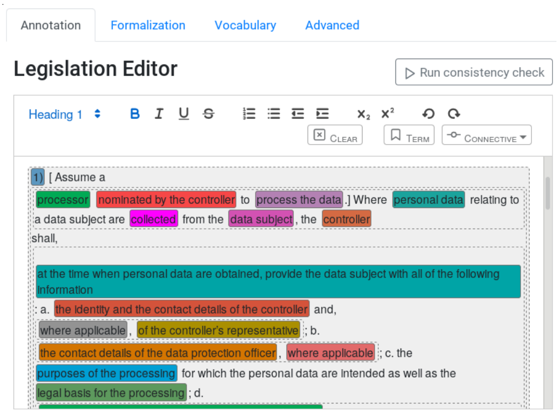
The choice in favor of $\mathbf{DL}^{*}_1$ is primarily motivated by the fact that it can be effectively automated using a shallow semantical embedding into normal (bi-) modal logic [12]. This enables the use of readily available reasoning systems for such logics; in contrast, there are few to none automated reasoning systems available for normative logics (with the exception of [7]). In NAI, we use the MleanCoP prover [19] for first-order multi-modal logics as it is currently one of the most effective systems and it returns proof certificates which can be independently assessed for correctness [18]. It is also possible to use various different tools for automated reasoning in parallel (where applicable). This is of increasing importance once multiple different logical formalisms are supported.

2.3 The Annotation Editor

The annotation editor of NAI is one of its central components. Using the editor, users can create formalizations of legal documents that can subsequently used for formal legal reasoning. The general functionality of the editor is described in the following.

One of the main ideas of the NAI 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 NAI suite are not necessarily logicians and 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. Note that the user can still inspect the logical formulae that result from the annotation process and also input these formulae directly. However, this feature is considered advanced and not the primary approach put forward by NAI.

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. An example of an annotation result is displayed in Fig. 2



1) [Assume a processor nominated by the controller to process the data .] Where personal data relating to a data subject are collected from the data subject , 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 ; c. the purposes of the processing for which the personal data are intended as well as the legal basis for the processing ; d.

Fig. 2: Article 13, par 1: 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.

2.4 The Application Programming Interface (API)

All the reasoning features of NAI can also be accessed by third-party applications. The NAI suite exposes a RESTful (Representational state transfer) API which allows (external) applications to run consistency checks, checks for independence as well as queries and use the result for further processing. The exposure of NAI's REST API is particularly interesting for external legal applications that want to make use of the already formalized legal documents hosted by NAI. A simple example of such an application is a tax counseling web site which advises its visitors using legal reasoning over a formalization of the relevant tax law done in the NAI suite.

3 Formalizing GDPR Article 13

An essential element in the compliance checking of privacy policies and data collection procedures, article 13 [3] of the GDPR is concerned with their transparency. This article contains 4 paragraphs, where the first two contain each 6 subsections. The third paragraph extends and modify the second one while the last states a situation in which the three previous paragraphs do not hold.

This section describes the formalization of this article using the NAI suite and the extensions to the suite which were implemented in order to support this formalization. While all necessary information is contained in the paper, the reader is still invited to follow this section and the one immediately after while simultaneously looking at the formalization in the tool itself. The formalization and queries which resulted from this work can be freely accessed via the NAI suite web application on a demo account. No registration is needed and no information (such as IP addresses) is recorded[4].

3.1 Annotating Paragraph 1

The NAI suite, as described in the previous section, requires us first to create a new legislation and then copy the original text into the editor pane. The first paragraph describes a situation in which a controller is obliged to communicate different information to the data subject, according to different conditions. Although not explicitly written, this paragraph also talks about processors and the processing of the data itself, as well as of its collection. In order to formalize the article, we must make these elements explicit in the text. We therefore add this information in brackets ('[',']') as can be seen in Fig. 3a.

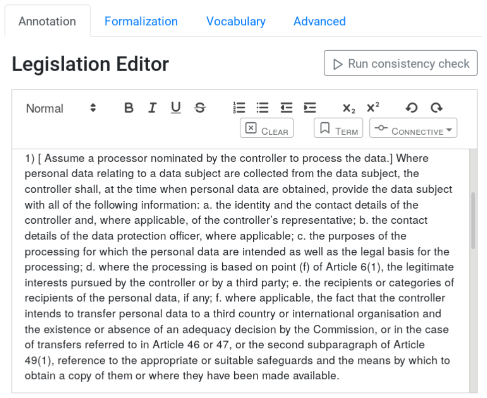
Given the explicit text of the paragraph, we are ready to annotate it. The first step of every formalization using the NAI suite is to point out all the legal terms used in the legal text. Those terms correspond to the colorful annotations in the editor. There are many relations between the different legal terms. For example, the personal data of the data subject is being collected and processed and is subject to the supervision of a Data Privacy Officer (DPO). Such complex relations require an expressive language such as fist-order logic, which is used in the annotations of the article.

As an example of term annotations, one can consider the phrases "data subject" and "personal data", for which the following first-order terms were assigned (respectively): `data_subject(Subject)` and `personal_data(Data, Subject)`. As described in Sec. 2, words starting with a lower-case letter are considered as constants while those starting with a capital letter are considered as variables which are quantified over the whole logical expression. An example of this step can be seen in Fig. 4a.

The full list of annotated legal terms can be found on the "Vocabulary" tab in the legislation editor. An example of this tab can be seen in Fig. 3b.

[3] `https://eur-lex.europa.eu/eli/reg/2016/679/oj`

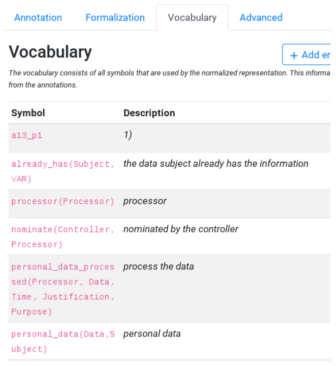
[4] Please login to `https://nai.uni.lu` using the email address: gdpr@nai.lu and password: nai. Please note that this account is write protected and cannot be changed.



Legislation Editor

1) [Assume a processor nominated by the controller to process the data.] Where personal data relating to a data subject are collected from the data subject, 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; c. the purposes of the processing for which the personal data are intended as well as the legal basis for the processing; d. where the processing is based on point (f) of Article 6(1), the legitimate interests pursued by the controller or by a third party; e. the recipients or categories of recipients of the personal data, if any; f. where applicable, the fact that the controller intends to transfer personal data to a third country or international organisation and the existence or absence of an adequacy decision by the Commission, or in the case of transfers referred to in Article 46 or 47, or the second subparagraph of Article 49(1), reference to the appropriate or suitable safeguards and the means by which to obtain a copy of them or where they have been made available.

(a) Adding implicit assumptions



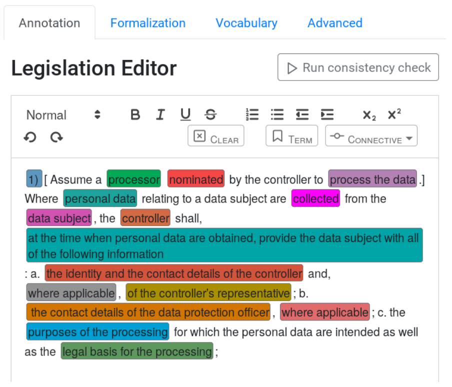
Vocabulary

The vocabulary consists of all symbols that are used by the normalized representation. This informa from the annotations.

Symbol	Description
`a13_p1`	*1)*
`already_has(Subject, VAR)`	*the data subject already has the information*
`processor(Processor)`	*processor*
`nominate(Controller, Processor)`	*nominated by the controller*
`personal_data_processed(Processor, Data, Time, Justification, Purpose)`	*process the data*
`personal_data(Data,Subject)`	*personal data*

(b) Some vocabulary

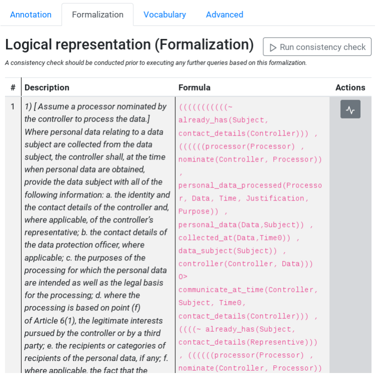
Fig. 3: Article 13, par. 1



Legislation Editor

1) [Assume a processor nominated by the controller to process the data .] Where personal data relating to a data subject are collected from the data subject , 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 ; c. the purposes of the processing for which the personal data are intended as well as the legal basis for the processing ;

(a) Some term annotations



Logical representation (Formalization)

A consistency check should be conducted prior to executing any further queries based on this formalization.

#	Description	Formula	Actions
1	*1) [Assume a processor nominated by the controller to process the data.] Where personal data relating to a data subject are collected from the data subject, 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; c. the purposes of the processing for which the personal data are intended as well as the legal basis for the processing; d. where the processing is based on point (f) of Article 6(1), the legitimate interests pursued by the controller or by a third party; e. the recipients or categories of recipients of the personal data, if any; f. where applicable, the fact that the*	`((((((((((~ already_has(Subject, contact_details(Controller))) , ((((((processor(Processor) , nominate(Controller, Processor)) , personal_data_processed(Processor, Data, Time, Justification, Purpose)) , personal_data(Data,Subject)) , collected_at(Data,Time0)) , data_subject(Subject)) , controller(Controller, Data))) O> communicate_at_time(Controller, Subject, Time0, contact_details(Controller))) , ((((~ already_has(Subject, contact_details(Representive))) , ((((((processor(Processor) , nominate(Controller, Processor))`	

(b) Some generated formulae

Fig. 4: Article 13, par. 1

Once all legal terms are annotated, we can proceed with annotating the relationships between them as well. The NAI suite supports two kinds of such annotations. The basic connectives cover logical and deontical annotations. The macros cover all annotations which do not fall within the first group and which normally correspond to certain sentence structures. Connectives of both groups can be found under the "Connectives" tab in the editor. All the connectives within these groups are used for annotating other, already existing, annotations.

The structure of the paragraph is the following. A set of conditions for the whole paragraph are followed by an obligation to communicate information. The precise information to communicate then follows in each of the items, possibly with some further conditions. Such a sentence structure is beyond most logics. In particular, it is not supported by the logic DL*, which is currently used by the NAI suite. A correct formalization of this sentence will consist therefore of a conjunction of obligations, each with its own set of conditions, which are formed by combining the general conditions of the paragraph with the specific condition of each item.

In order for the annotation process to be as simple as possible, we need to avoid major changes to the original text. We need therefore to be able to annotate the original text such that the target structure mentioned above is generated.

The NAI suite supports such a sentence structure via the "Multi-obligation Macro". This macro accepts an annotation denoting the general conditions and a second annotation denoting the additional obligations and their specific conditions. When applied, the macro generates a conjunction of obligations, one for each of the annotated obligations and with the general conditions as well as the specific ones. We therefore annotate the whole paragraph 1 with this annotation.

We now proceed with annotating the conditions and obligations. First, we use the "And" connective to annotate all the general conditions, such as the existence of a process and a controller, etc. We then proceed by using the "And" connective to annotate all the obligations. Each obligation can have one of three forms. A simple obligation contains just a term. The macro will convert this obligation into a conditional obligation where the conditions are all the general ones from the first conjunction. The more complex obligations are either "If / Then" and "Always / If".

The difference between these two connectives is syntactical only. While in "If / Then", the conditions are specified by the first annotation and the conclusion with the second, the order is reversed in "Always / If". By using one of these two annotations for the differed obligations, the macro will know to add the additional, specific conditions to the conditions of each resulted obligation. An example of the fully annotated text can be seen in Fig. 2.

The automatically generated annotation of the first item of paragraph 1 can be seen in Fig. 5

Once annotating the text has finished and the user clicks the "Save" button, the tool generates representation of the formalization in the logic $\mathbf{DL}^*_1$. The formulae are accessible via the "Formalization" tab. An example can be seen in Fig. 4b.

```
IF-THEN-OB(AND(processor(Processor),nominate(Controller,
Processor),personal_data_processed(Processor,Data,Time,
Justification,Purpose),personal_data(Data,Subject),
collected_at(Data,Time0),data_subject(Subject),
controller(Controller, Data)),communicate_at_time(Controller,
Subject,Time0,contact_details(Controller)))
```

Fig. 5: Automatically generated annotation of item 1 in paragraph 1

The specific annotations can be seen by hovering with the mouse over the relevant parts of the text.

The annotation of paragraph 2 is similar.

3.2 Annotating Paragraph 3

Paragraph 3 is relatively short syntactically but complex semantically. It expands on paragraph 2 and places further conditions and obligations. While paragraph 2 describes the obligations in case of the first data processing, paragraph 3 describes those in all subsequent ones.

The annotation of this paragraph makes use of two additional macros. The ”Labeling” macro supports naming previous annotations while the ”Copying” macro uses such labels in order to copy previous annotations and modify them.

The modification which is required for this paragraph is the addition of the conditions involving further processing. In addition, the macro makes sure to apply all the obligations from paragraph 2 to the additional processing instead of to the original one. Lastly, the macro adds the additional obligation for the controller to communicate the purpose of this additional processing as well.

Facilitating Correct Formalizations There is a further interesting issue which relates to this paragraph. Clearly, its annotation is far from trivial and is error prone. Still, by using macros and annotations, the chance of error occurring is reduced since there is a clear connection between the text itself and its annotation. This is not the case when the paragraph is translated into a logical formula manually. As an example, consider the DAPRECO formalization of the GDPR [21]. As mentioned in the introduction to this paper, this ambitious knowledge base contains a manual translation of almost all articles of the GDPR.

Nevertheless, if we focus on the translation of this paragraph [5], we can see several errors. First, the translation does not mention at all the fact that all of the required information mentioned in paragraph 2 is required here as well. Even more important though, there is no distinction between the two processing events of the data, except in the name of the variable used to denote them. There is a relation between the times of the two occurrences but it is defined as ”greater or

[5] Please search for the text ”statements51Formula” in `https://raw.githubusercontent.com/dapreco/daprecokb/master/gdpr/rioKB_GDPR.xml`, of a version no later than 11/2019.

equal". Clearly, this formalization will always apply to any processing, whether first or not, since one can substitute for the universally quantified variables the same processing and the same time.

Such an error is not easy to spot, when one translates regulations manually. Using macro annotations makes errors easier to spot and avoid due to several reasons. First, annotating the original text makes the formalization closer to the text, than when replacing this text with another, which is closer to the end formula. Second, the well known software engineering principle - Don't Repeat Yourself (DRY) - is applicable also here. If we manually expand paragraph 1, for example, we obtain at least 6 different new sentences of very similar content. The same goes with duplicating all the requirements of paragraph 2 in paragraph 3. The same benefits of using this principle in software engineering apply also here. For example, any update or change to the repeated information needs to be done only once and therefore, the chances of typos and mistakes are reduced. Last, it should be noted that the formalization process is far from being trivial. For example, the process of formalizing just article 13 took the author almost a day. Automating as much of this process by the use of macros has saved some time and as a consequence, reduced the change of making an error.

On the other hand, when using annotations and macros, we could more easily spot this and use a correct annotation.

3.3 Annotating Paragraph 4

The last paragraph in the article has a standard legal form. Its purpose is to set exceptional circumstances in which other paragraphs do not hold. In our example, none of the obligations in the previous paragraphs should hold in case the data subject already has the required information.

In the previous subsection, we have seen a utility macro for labeling annotations. This macro is handy here as well as we will need to be able to refer to other annotations, in order to apply a macro for exceptional circumstances.

The "Exception" macro takes the labels of other annotations and apply to them the negation of a further condition. In this case, the negated condition is that the data subject does not already has the required information.

4 Automated GDPR Compliant Checking

The previous section listed some of the technical issues we face, as well as their solution, when trying to correctly formalize complex legal documents. Section 2 has described several automated deduction based tools, which can be used for checking the correctness of the formalization. Nevertheless, the main usage of automated deduction within the NAI suite is for allowing the computer to answer questions and make legal deductions. Given a correctly formalized legislation, NAI can currently answer Yes / No questions. This is done by employing the state-of-the-art theorem prover MleanCoP [19], which in turn tries to build a formal proof that the question logically follows from the formalization and assumptions. Since this process is incomplete, some negative answers cannot be given. The NAI suite displays a warning in this case.

The main expected usage of this feature is by third-party tools, which will use the deduction engine of the NAI suite over an already formalized legislation in order to answer arbitrary questions. For example, the queries we are going to see in this section might be constructed automatically by such a third-party tool and sent, via the API, to the NAI suite for compliance checking.

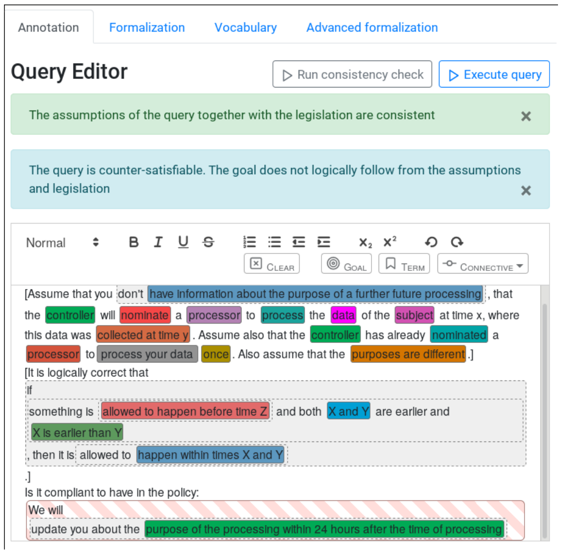
Nevertheless, the NAI suite also supports the possibility of writing queries directly in the tool. This feature is mainly used for testing and as a support tool for lawyers and jurists. Similarly, this feature can be used in order to demonstrate automatic reasoning over the article.

The example queries in this section are all about the compliance of certain sentences from possible privacy policies. There are various ways for checking compliance and some are mentioned in [20]. In this section, compliance is being checked by asking if something is permitted. In all the examples, we will ask if a sentence appearing in the privacy policy can be considered as permitted according to the law. This gives raise to at least two interpretation of permissions, weak permissions allow something in case it is not explicitly forbidden by law while strong permissions allow something in case it is explicitly permitted [10]. Both can be obtained by using the correct formalizations but the more straightforward, at least with regard to automated reasoning and in the absence of meta-logical operators such as negation-as-failure, is the strong permission. We will therefore take permission to mean strong permission in this section. These questions can be found and executed on the ”Queries” menu of the tool.

In order to best illustrate how the tool can be used for compliance checking, a specific scenario is chosen. We assume that the privacy policy contains a sentence about a possible future use of the subject data which was not considered when the data was collected. The policy then describes the time frame in which the purpose of this use will be communicated to the subject. We distinguish three different time frames and discuss for each how automated reasoning can provide an answer for compliance.

Time frame 1: within 24 hours after the additional processing. in order to create a query to this affect, one must first build the whole scenario. In our case, it contains information about data, data subject, controller, the first processor and the second processor. For the reader who is following the demo account on nai.uni.lu, this information appears within brackets for all the compliance queries. Figures 6 and 7 shows the annotation and generated formalization of this query, respectively.

We are now in a position to ask our specific yes/no question: ”Is it compliant to have in the policy: We will update you about the purpose of the processing within 24 hours after the time of processing”? The way we formalize this question as a compliance test is as follows: ”Is it permitted by article 13 of the GDPR to communicate this information to the data subject within 24 hours after the time of processing”?



Query Editor

The assumptions of the query together with the legislation are consistent

The query is counter-satisfiable. The goal does not logically follow from the assumptions and legislation

[Assume that you don't have information about the purpose of a further future processing , that the controller will nominate a processor to process the data of the subject at time x, where this data was collected at time y . Assume also that the controller has already nominated a processor to process your data once . Also assume that the purposes are different .]
[It is logically correct that
if
something is allowed to happen before time Z and both X and Y are earlier and
X is earlier than Y
, then it is allowed to happen within times X and Y
.]
Is it compliant to have in the policy:
We will
update you about the purpose of the processing within 24 hours after the time of processing

Fig. 6: Annotation of first compliance query

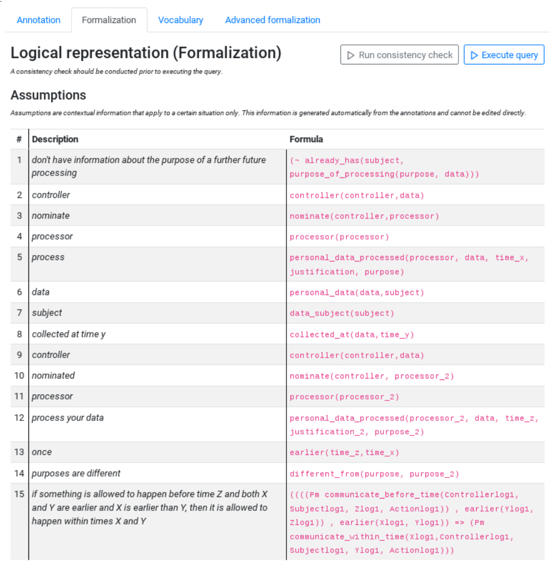
The above query and assumptions, as well as the formalization of the GDPR are then being translated into multi-modal first-order logic and sent to the MleanCoP prover. The prover will find a proof only if the above normative question can be derived. Clearly, this time frame is beyond that allowed to the controller to communicate this information and indeed, the theorem prover will answer ”No” to the question.

In Fig. 6, one can see the result of executing the query displayed above the annotation, as well as the result of running a consistency check.

The reader might note that this question concerns time and that time computations require numerical operators such as ”greater than”, etc. The logic currently in use does not support such operations out of the box and they need to be included in the formalization. The formalization gives us the certain time of each of the processings as well as the time of the collection of data. We additionally use temporal sentences such as before, after, at and within. In this case, paragraph 3 of the article obliges the controller to notify the subject any time before the second processing takes place. We ask a question about a time frame which is within 24 hours after this processing took place. For the compliance test to be accurate, we must add logical expressions associating the terms ”before” and ”within” correctly, such as $\forall X, Y, Z, Action : before(Action, Z) \wedge earlier(X, Y) \wedge earlier(Y, Z) \Rightarrow within(Action, X, Y)$.

We add this formula to the formalization and click ”Execute query”. The NAI Suite answers ”No”. We conclude that the above sentence which appears in a privacy policy is not compliant with the law.

Time frame 2: anytime before the additional processing. This query is much easier as it asks for permission of something which is already an obligation.


Annotation Formalization Vocabulary Advanced formalization

Logical representation (Formalization)

Run consistency check · Execute query

A consistency check should be conducted prior to executing the query.

Assumptions

Assumptions are contextual information that apply to a certain situation only. This information is generated automatically from the annotations and cannot be edited directly.

#	Description	Formula
1	*don't have information about the purpose of a further future processing*	`(~ already_has(subject, purpose_of_processing(purpose, data)))`
2	*controller*	`controller(controller,data)`
3	*nominate*	`nominate(controller,processor)`
4	*processor*	`processor(processor)`
5	*process*	`personal_data_processed(processor, data, time_x, justification, purpose)`
6	*data*	`personal_data(data,subject)`
7	*subject*	`data_subject(subject)`
8	*collected at time y*	`collected_at(data,time_y)`
9	*controller*	`controller(controller,data)`
10	*nominated*	`nominate(controller, processor_2)`
11	*processor*	`processor(processor_2)`
12	*process your data*	`personal_data_processed(processor_2, data, time_z, justification_2, purpose_2)`
13	*once*	`earlier(time_z,time_x)`
14	*purposes are different*	`different_from(purpose, purpose_2)`
15	*if something is allowed to happen before time Z and both X and Y are earlier and X is earlier than Y, then it is allowed to happen within times X and Y*	`((((Pm communicate_before_time(Controllerlog1, Subjectlog1, Zlog1, Actionlog1)) , earlier(Ylog1, Zlog1)) , earlier(Xlog1, Ylog1)) => (Pm communicate_within_time(Xlog1,Controllerlog1, Subjectlog1, Ylog1, Actionlog1)))`

Fig. 7: Automatically generated assumptions of the first compliance query

We expect the tool to answer "Yes" and indeed it does. Therefore, having the sentence above is compliant with the law.

Time frame 3: at the moment of collecting the data. This is a more interesting query which depends again on some temporal reasoning. In this case, we need to say that if something is permitted before time Z and that time X is earlier than time Z, then it is also permitted in time X. Nevertheless, the tool answers "Yes" even without us adding a logical expression of the above.

The reason for that is the following. The interpretation which was given by the author to paragraph 2 of the article is that it holds for any processing, whether it is the first, second or later. At the same time, paragraph 3 holds for any processing except the first one. Therefore, it seems that both paragraphs apply to any processing except the first, while for the first processing, only paragraph 2 applies. This might be a simplistic view of the law, not taking into account different legal interpretations such as those which consider paragraph 3 as "lex specialis" to paragraph 2 and therefore exclude the subject of paragraph 3 from paragraph 2. For the normative reasoning done by the tool to be correct, a "correct" legal interpretation must be used. This discussion about the "correct" legal interpretation of the GDPR is beyond the scope of this paper and we take paragraph 3 to be "lex posterior" to paragraph 2.

Still, whether because of our legal interpretation of because of the addition of the temporal expressions, the tool answers "Yes". It is indeed compliant with the law to communicate to the subject the purpose of the second processing already at the time of collecting the data.

5 Conclusion and Future Work

Automated reasoning over legal texts in general, and the GDPR in particular, is advantageous. Automatizing reasoning over the GDPR can support lawful, ethical and robust data privacy governing as well as compliance checking.

This paper has presented a formalization of article 13 of the GDPR. The article was first annotated using the NAI suite. The logical interpretation of the article was then automatically extracted. Several examples of queries over this formalization were then presented. The real potential though is by connecting various third-party tools to the NAI suite API. This will allow the creation of tools for supporting the work of controllers, data privacy officers, regulators and auditors, to name a few.

This work can be extended to include more articles of the GDPR as well as relevant case law. At the same time, web applications can be created based on the formalization to support the work of various people, as discussed in the previous paragprah.

Currently, the NAI suite supports an expressive deontic first-order language. This language is rich enough to describe many scenarios which appear in legal texts. Nevertheless, more work is required in order to capture all such scenarios. Among those features with the highest priority, we list support for exceptions, temporal sentences and arithmetic. In Section 3.3, one possible direction for addressing exceptions was given. Other possible solutions for these issues already exist in the form of tools such as non-monotonic reasoners [11], temporal provers [24] and SMT solvers [5].

On the level of usability, the tool currently does not give any information as to why a query is counter-satisfiable. The user needs to look on the vocabulary in order to determine possible reasons. Integrating a model finder, such as Nitpick [3], will help "debugging" formalizations and enriching the query language.

NAI's graphical user interface (GUI) aims at being intuitive and easy to use and tries to hide the underline complexities of the logics involved. A continuously updated list of new features can be found on the GUI's development website [6] .

References

1. Aucher, G., Berbinau, J., Morin, M.L.: Principles for a judgement editor based on binary decision diagrams. IfCoLog Journal of Logics and their Applications **6**(5), 781–815 (2019)
2. Biagioli, C., Mariani, P., Tiscornia, D.: Esplex: A rule and conceptual model for representing statutes. In: Proceedings of the 1st international conference on Artificial intelligence and law. pp. 240–251. ACM (1987)
3. Blanchette, J.C., Nipkow, T.: Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In: ITP. pp. 131–146. Springer (2010)
4. Boella, G., Di Caro, L., Humphreys, L., Robaldo, L., Rossi, P., van der Torre, L.: Eunomos, a legal document and knowledge management system for the web to provide relevant, reliable and up-to-date information on the law. Artificial Intelligence and Law **24**(3), 245–283 (2016)

[6] `https://github.com/normativeai/frontend/issues`

5. Bouton, T., de Oliveira, D.C.B., Déharbe, D., Fontaine, P.: verit: an open, trustable and efficient smt-solver. In: CADE. pp. 151–156. Springer (2009)
6. de Bruin, H., Prakken, H., Svensson, J.S.: The use of legal knowledge-based systems in public administration: what can go wrong? In: Evaluation of Legal Reasoning and Problem-Solving Systems. pp. 14–16 (2003)
7. Governatori, G., Shek, S.: Regorous: a business process compliance checker. In: Proc. of the 14th Int. Conf. on Artificial Intelligence and Law. pp. 245–246. ACM (2013)
8. Groothuis, M.M., Svensson, J.S.: Expert system support and juridical quality. In: Proceedings of JURIX. vol. 110. IOS Press, Amsterdam (2000)
9. Hashmi, M., Governatori, G.: Norms modeling constructs of business process compliance management frameworks: a conceptual evaluation. Artif. Intell. Law **26**(3), 251–305 (2018). https://doi.org/10.1007/s10506-017-9215-8, `https://doi.org/10.1007/s10506-017-9215-8`
10. Henrik, G., Wright, V.: Norm and action, a logical enquiry (1963)
11. Kifer, M.: Nonmonotonic reasoning in flora-2. In: International Conference on Logic Programming and Nonmonotonic Reasoning. pp. 1–12. Springer (2005)
12. 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. ACM (2019)
13. Libal, T., Steen, A.: The nai suite - drafting and reasoning over legal texts. In: JURIX. To appear (2019)
14. Libal, T., Steen, A.: Nai: The normative reasoner. In: ICAIL. pp. 262–263. ACM (2019)
15. Makinson, D., Van Der Torre, L.: Input/output logics. Journal of philosophical logic **29**(4), 383–408 (2000)
16. de Montety, C., Antignac, T., Slim, C.: Gdpr modelling for log-based compliance checking. In: IFIP International Conference on Trust Management. pp. 1–18. Springer (2019)
17. Nanda, R., Siragusa, G., Di Caro, L., Theobald, M., Boella, G., Robaldo, L., Costamagna, F.: Concept recognition in european and national law. In: JURIX. pp. 193–198 (2017)
18. Otten, J.: Implementing connection calculi for first-order modal logics. In: IWIL@ LPAR. pp. 18–32 (2012)
19. Otten, J.: Mleancop: A connection prover for first-order modal logic. In: 7th International Joint Conference, IJCAR. pp. 269–276 (2014). https://doi.org/10.1007/978-3-319-08587-6_20
20. Palmirani, M., Governatori, G.: Modelling legal knowledge for gdpr compliance checking. In: JURIX. pp. 101–110 (2018)
21. Robaldo, L., Bartolini, C., Palmirani, M., Rossi, A., Martoni, M., Lenzini, G.: Formalizing gdpr provisions in reified i/o logic: The dapreco knowledge base. Journal of Logic, Language and Information pp. 1–49 (2019)
22. Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T.: The british nationality act as a logic program. Communications of the ACM **29**(5), 370–386 (1986)
23. Stamper, R.: Legol: Modelling legal rules by computer. Computer Science and Law pp. 45–71 (1980)
24. Suda, M., Weidenbach, C.: A pltl-prover based on labelled superposition with partial model guidance. In: IJCAR. pp. 537–543. Springer (2012)