The NAI Suite – Drafting and Reasoning over Legal Texts

Tomer LIBAL [a] and Alexander STEEN [b]
[a] *American University of Paris*
[b] *University of Luxembourg*

Abstract. A prototype for automated reasoning over legal texts, called NAI, is presented. As an input, NAI accepts formalized logical representations of such legal texts that can be created and curated using an integrated annotation interface. The prototype supports automated reasoning over the given text representation and multiple quality assurance procedures.

Keywords. Legal Reasoning, Deontic Logic, Automated Reasoning

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. In addition, international organizations like the European Union are constantly aiming at combining and integrating separate legal systems [3]. In contrast to this situation, the automation of legal reasoning is still underdeveloped albeit being a growing field of research. Approaches for automatic reasoning over sets of norms have been developed, such as for business [6] and GDPR compliance [11].

One of the reasons for the relatively restricted number of applications of automated reasoning to the legal domain is the lack of editing tools which can be used by non-logicians. Indeed, the applications mentioned above are mainly based on the work of logicians. The most popular approach to the formalization of legal texts is by using a logic programming language. This approach has enjoyed success [4] in the last 40 years and is still popular today. Nevertheless, basic knowledge of logic programming is still required. In order to have a wider use of legal reasoning, other professionals, such as lawyers and jurists, should be able to use the tools.

A second reason is the lack of tools and methodologies for asserting the correctness of the logical representations of the legal texts. Among existing results, one can find a methodology for building legal ontologies [8] and more concretely to our approach, one for validating formal representations of legal texts [1].

Lastly, the scarcity of legal reasoning software prevents the utilization of such formalizations, even if proven correct. One can mention here the engines for defeasible log-

ics [5], Higher-order logics [2] Deontic logics with contrary-to-duty obligations [7], as well as logic programming [4].

In this paper we describe the new normative reasoning framework NAI, which addresses these problems by providing functionality and methodology for lawyers and jurists. NAI is a web application and is readily available at `https://nai.uni.lu`. NAI is also open-source, its source code is freely available at GitHub[1] under GPL-3.0 license.

NAI features an annotation-based editor which abstracts over the underlining logical language. It also contains an easily accessible functionality for ensuring that the formalization is consistent and that the formalized sentences are independent from each other. NAI also supports a methodology for proving the correctness of formalizations via execution of behavioral tests. Lastly, it provides an interface for the creation of queries and for checking their validity.

The architecture of NAI is modular, which allows using different logics and reasoning engines. It also provides an API, which can be used by other tools in order to reason over the formalized legislation.

In this paper we give a technical description of a new tool for legal formalization and reasoning which utilizes an innovative annotations interface. An example of its usage and a demonstration of a new Agile methodology for formalizing legal texts are presented in the full paper [2] and can be seen on a demo account [3].

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. 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.

[1] See `https://github.com/normativeai`.

[2] `http://arxiv.org/abs/1910.07004`

[3] Please login to `https://nai.uni.lu` using Email address: smoking@nai.lu / Password: nai

2.1. 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 Deontic logic described in [7]; however, the architecture of NAI is designed in such a way that further formalisms can easily be supported.

The current choice of Deontic logic is primarily motivated by the fact that it can be effectively automated using a shallow semantical embedding into normal (bi-)modal logic [7]. This enables the use of readily available reasoning systems for such logics; in contrast, there are relatively few dedicated automated normative reasoning systems such as the one described in [5]. In NAI, we use the MleanCoP prover [10] 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 [9]. 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.

2.2. 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. A more detailed exemplary application on a concrete legal document is presented in the demo.

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 roughly correspond to logical connectives) and the containing text can be annotated recursively. Composite statements are displayed as a box around the text.

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.

2.3. The Abstract 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.

References

[1] Cesare Bartolini, Gabriele Lenzini, and Cristiana Santos. An interdisciplinary methodology to validate formal representations of legal text applied to the gdpr. In *JURISIN*, 2018.

[2] Christoph Benzmüller, Ali Farjami, Paul Meder, and Xavier Parent. I/O logic in HOL. *IfCoLoG Journal of Logics and their Applications*, 6(5):715–732, 2019.

[3] Anne-Marie Burley and Walter Mattli. Europe before the court: a political theory of legal integration. *International organization*, 47(1):41–76, 1993.

[4] Trevor JM Bench-Capon et al. Logic programming for large scale applications in law: A formalisation of supplementary benefit legislation. In *Proceedings of ICAIL*, pages 190–198. ACM, 1987.

[5] Guido Governatori and Sidney Shek. Regorous: a business process compliance checker. In *Proc. of the 14th Int. Conf. on Artificial Intelligence and Law*, pages 245–246. ACM, 2013.

[6] Mustafa Hashmi and Guido Governatori. Norms modeling constructs of business process compliance management frameworks: a conceptual evaluation. *Artif. Intell. Law*, 26(3):251–305, 2018.

[7] Tomer Libal and Matteo Pascucci. Automated reasoning in normative detachment structures with ideal conditions. In *Proc. of ICAIL*, pages 63–72, 2019.

[8] Martynas Mockus and Monica Palmirani. Legal ontology for open government data mashups. In *2017 Conference for E-Democracy and Open Government (CeDEM)*, pages 113–124. IEEE, 2017.

[9] Jens Otten. Implementing connection calculi for first-order modal logics. In *IWIL@ LPAR*, pages 18–32, 2012.

[10] Jens Otten. Mleancop: A connection prover for first-order modal logic. In *7th International Joint Conference, IJCAR*, pages 269–276, 2014.

[11] Monica Palmirani and Guido Governatori. Modelling legal knowledge for gdpr compliance checking. In *Legal Knowledge and Information Systems: JURIX*, volume 313, pages 101–110. IOS Press, 2018.