NAI: TOWARDS TRANSPARENT AND USABLE SEMI-AUTOMATED LEGAL ANALYSIS

Tomer Libal / Alexander Steen

Assistant Professor, American University of Paris, Dep. of Computer Science, Mathematics and Environmental Science
102, rue St Dominique, 75007 Paris, FR
tlibal@aup.edu; http://tomer.libal.info/

Postdoctoral Researcher, Universität Luxemburg, Fakultät für Naturwissenschaften, Technologie und Kommunikation
6, avenue de la Fonte, L-4364 Esch-sur-Alzette, LU
alexander.steen@uni.lu; https://alexandersteen.de

Keywords: ***Legal Reasoning, Automated Theorem Proving, Graphical User Interface***

Abstract: *A prototype for automated reasoning over legal documents, called NAI, is presented. It uses formalized representations of legal documents that are created using a graphical editor that is also provided as part of NAI. The prototype supports several automated reasoning procedures over the given formalizations, including the execution of user queries. The application of NAI is studied using a fragment of the Scottish Smoking Prohibition (Children in Motor Vehicles) Act 2016.*

1. Introduction

Computer-assisted legal reasoning technologies are becoming increasingly relevant in practice. New court cases and legislations are accumulated every day and navigating through the vast amount of complex information is far from trivial. In contrast to this situation, the employment of automated legal reasoning tools is still underrepresented in practice [de Bruin et al. 2003] albeit being a relevant and active field of research since the 1980s [Stamper 1980, Sergot et al. 1986, Biagioli et al. 1987]. In recent years automatic procedures, e.g. for courtroom management[1], legal language processing/management [Boella et al. 2016], and normative compliance tools[2] have been introduced. At the same time, approaches for automatic reasoning over sets of norms have been developed, such as in the courtroom [Guillaume et al. 2019], for business compliance [Hashmi/Governatori 2018] and General Data Protection Regulation (GDPR) compliance [Palmirani/Governatori 2018].

One of the reasons for the relatively restricted number of practical applications of automated reasoning in the legal domain might be the lack of systems that are both usable, also by non-logicians, and simple to integrate in existing processes. Also, creating formal representations of legal documents is usually quite laborious, and there currently seems to be no platform for exchanging (or trading with) already formalized works. Furthermore, asserting correctness of logical representations of the legal documents is challenging. Among existing results, one can find a methodology for building legal ontologies [Mockus/Palmirani 2017] and more concretely to our approach, one for validating formal representations of legal texts [Bartolini et al. 2018]. Finally, legal reasoning systems have to deal with many difficult aspects, such as exceptions, counterfactuals and cross-references [Routen 1989] and further logical challenges. There exist some engines that deal with (some of) these aspects [Governatori/Shek 2013, Libal/Pascucci 2019], and recent work on higher-order automation suggests that there might be a holistic base logic for many of these aspects rooted in type theory [Benzmüller 2019].

[1] See http://softpert.com/legal/court-management/winjuris.
[2] See https://cst.cnpd.lu/portal for GDPR compliance checking.

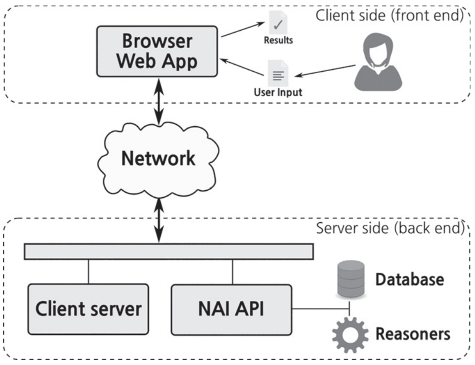


Figure 1: Software-as-a-service architecture of the NAI platform. The front-end software is designed as a browser application; the back-end side runs on remote servers. Data flow is indicated by arrows.

In this paper, we describe the prototypical normative reasoning framework NAI (for Normative AI), which tries to address these problems. NAI features an annotation-based editor which abstracts over the underlying logical language. It also contains an easily accessible functionality for quality assurance and a transparent analysis of the created formalized document. NAI also supports an approach for assessing the correctness of formalizations via execution of behavioural tests using so-called queries. Lastly, it provides an interface for the creation of such queries and for checking their validity.

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 3 under GPL-3.0 license.

2. The NAI Suite

The NAI suite integrates state-of-the-art reasoning technology into a usable graphical user interface (GUI) for the computer-assisted formalization of legal texts and the application of automated normative reasoning procedures on these artefacts. In particular, NAI includes

- a legislation editor that graphically supports the transformation of legal documents into computer-assessable formal knowledge bases,
- assistance technology for quality assurance, e.g., using automated checks for different logical properties,
- off-the-shelf availability of reasoning systems that evaluate user-specified queries with respect to a given knowledge base, and
- a knowledge transfer platform for sharing and reusing already existing knowledge basis.

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 reasoning systems, 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. All necessary software is made available on the NAI 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.

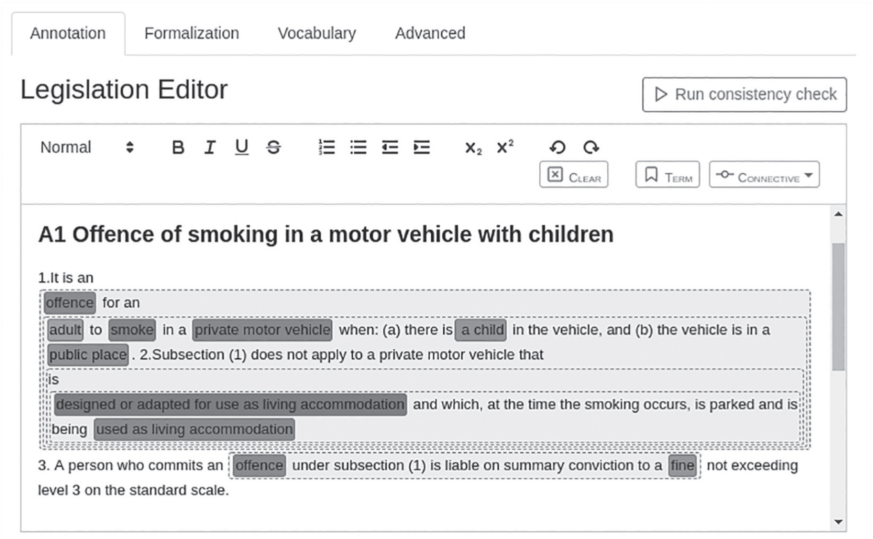
2.1. The Annotation Editor

The annotation editor of NAI is one of its central components. Using the editor, users can create formalizations of legal documents that can subsequently be used for formal legal reasoning. The general functionality of the editor is described in the following. A more detailed case study highlighting the usage of the annotation editor on a concrete legal document is presented in Sect. 3.

One of the main motivations 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, but rather legal professionals. Therefore, it would significantly decrease the usability of a tool if a solid background knowledge about formal logic and programming was required. This is currently realized by allowing the user to graphically annotate legal documents and queries, and by making the different reasoning functionalities accessible via simple graphical means in the GUI (e.g., by buttons). Note that, if desired, a user can still inspect the automatically generated logical formulae that result from the annotation process, and also input these formulae directly. However, this feature is considered advanced and not the primary usage put forward by NAI.

The formalization of legal documents proceeds as follows: The user selects some text fragment from the legal document and annotates it, either as a so-called term or as a composite 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. An example of an annotation result is displayed in Fig. 2.

The editor also features, via buttons, direct access to the quality control features, including consistency checking and assessing logical independence (cf. Sect. 2.2 below for details on these concepts). When one of these procedures is invoked, the current state of the formalization will be translated into a machine-readable translation and sent to the back-end reasoning systems, which determine whether the current draft is logically consistent respectively logically independent.



A1 Offence of smoking in a motor vehicle with children

1.It is an offence for an adult to smoke in a private motor vehicle when: (a) there is a child in the vehicle, and (b) the vehicle is in a public place. 2.Subsection (1) does not apply to a private motor vehicle that is designed or adapted for use as living accommodation and which, at the time the smoking occurs, is parked and is being used as living accommodation

3. A person who commits an offence under subsection (1) is liable on summary conviction to a fine not exceeding level 3 on the standard scale.

Figure 2: Annotation example in the graphical editor of NAI. Different colours represent different notions, grey boxes with dashed borders indicate composite statements.

User queries are created using a similar annotation editor. In addition to the steps sketched above, users may declare a text passage as so-called goal using a dedicated annotation button, whose contents are again annotated as usual. If the query is executed, the back-end reasoning systems will try to validate that the goal is an immediate (logical) consequence which necessarily follows from the underlying formalized legal document (the knowledge base) and the contextual assumptions of the query at hand (the concrete case).

The editor was successfully used to formalize legal texts such as articles of the GDPR and other legislation. It was already used by lawyers and required but a little logical knowledge. Such a knowledge is many times possessed by legal practitioners such as lawyers, jurists. Nevertheless, we are constantly working on improving the interface in order to make it even more accessible.

2.2. The Reasoning Technology

The NAI suite supports formalizing legal documents into knowledge bases, as depicted above, using its legislation editor. The second core component of NAI is its reasoning infrastructure that is used for applying different automated logical operations on these knowledge bases. These operations can be separated into two main categories, that is, quality assurance and query answering.

Quality assurance procedures assist the user in the process of validating that the formalization of the legal document (i.e. the annotations created using the editor) do not contain – roughly speaking – logical fallacies. To this end, NAI offers so-called consistency checking of knowledge bases and queries and checks for logical independence. The first check finds logically contradictory information in the formalizations that arise from, possibly mistakenly, poorly chosen annotations. Inconsistency of a knowledge base can always be considered as an error and needs to be addressed. The second check finds redundancies in the formalization that also indicate possible formalization errors. Using the quality assurance procedures, the user can spot both kinds of errors and amend the formalization appropriately.

Query answering can be regarded as the core reasoning capability of NAI. Given an already formalized legal document, NAI supports creating and executing so-called queries on that formalization. A query is essentially a question of the form «*[given the information of the underlying legal document **L**,] is it the case that **X** in a situation where **Y** holds?*». Here, **X** is referred to as the goal of the query and **Y** represents the situational context (also called assumptions). These questions can be translated into a logical representation and amount to formally proving that **X** is a logical consequence of both **L** and **Y** using automated reasoning technology. These systems will then either prove that **X** is a necessary consequence or, in the negative case, provide additional information as to why this is not the case.

After formalization, the formal representation of the legal documents is stored in NAI in an intermediate expressive machine-readable format. There exist many different logical formalisms that have been discussed in the literature 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 intermediate format as result of the formalization process. Once a certain logical formalism for conducting the logical analysis is chosen, NAI will automatically translate the intermediate format into this specific formalism. Currently, NAI supports a deontic logic based on a bi-modal logical formalism [Libal/Pascucci 2019]; however, the architecture of NAI is designed in such a way that further formalisms can easily be supported. The automated theorem prover MleanCoP [Otten 2014] for first-order multi-modal logics is employed at the back end of NAI.

There are many challenges for the formalization of legal texts. On the one hand, they should support contrary-to-duty (CTD) obligations, temporal statements and defeasible norms. On the other, there needs to be an efficient automated processing by the computer. The current logic supports CTD obligations by default but does not provide any special support for the other requirements. This means that one needs to use the tools already existing in first-order logic, such as adding comparison relations for temporal reasoning and using classical negation (monotonic) for exception handling. This approach can be used in a satisfactory way in practice but will be clearly found inadequate by some readers. As is written above, the tool is modular and other underlying logics can be used. For example, one can use Prolog and its non-monotonic features to deal with defeasible norms.

2.3. Cloud-Based External Services

All the reasoning features of NAI can also be accessed by third-party applications. To this end, the NAI suite exposes a web-interface which allows (external) applications to run consistency checks, checks for independence as well as user queries and use the result for further processing. The supply of this interface is in particular interesting for external legal applications that want to make use of the already formalized knowledge bases hosted by NAI and operations thereon. A simple example of such an application is a tax counselling web site which advises its visitors using legal reasoning implemented in the NAI suite.

3. Case Study: Scottish Smoking Regulation

The usage of the NAI suite is demonstrated in the following using a case study of an existing legal document, called the «Smoking Prohibition (Children in Motor Vehicles) (Scotland) Act 2016».[3] This piece of legislation contains 19 articles regulating the act of smoking in vehicles when children are present. In this paper, only the formalization of the first article is discussed more thoroughly. A more comprehensive formalization is available online through the NAI system.[4]

The first article reads as follows:

Article 1: Offence of smoking in a motor vehicle with children

1. It is an offence for an adult to smoke in a private motor vehicle when: (a) there is a child in the vehicle, and (b) the vehicle is in a public place.

2. Subsection (1) does not apply to a private motor vehicle that is designed or adapted for use as living accommodation and which, at the time the smoking occurs, is parked and is being used as living accommodation.

3. A person who commits an offence under subsection (1) is liable on summary conviction to a fine not exceeding level 3 on the standard scale.

In order to apply automated reasoning procedures to this text, first the user's understanding of the document's meaning needs to be formalized. In other words, a legal interpretation of the text needs to be formalized in the system.

While in general different legal interpretations are possible for a given legal text, the interpretation of the example document at hand is comparably straight-forward. For the purpose of this example, the article is interpreted as a prohibition for adults to smoke in a private motor vehicle in case: (1) there is a child in the vehicle, (2) the vehicle is in a public space and (3) the vehicle is not adapted or designed to be used, and at the same time is being used, as living accommodation. When violating this prohibition, the adult is liable to a fine via a summary conviction.

3.1. Formalizing the Article

The formalization process essentially comprises of a computer-assisted translation of an informal natural language text into a formal logical formula (or code). The steps for formalizing a document in NAI are:

– Copying the legislation.
– Identification of all legal terms.
– Capturing the meaning of the sentences via annotations.

This section describes the usage of the online tool. We therefore recommend that the reader be logged in to the demo account. Once logged in to NAI, the user can choose to create a new or edit an existing formalized legislation. Using the demo account specified above, the one existing legislation can be selected and opened. In the account, three articles of the legislation have been copied, including article 1 which is discussed here.

[3] See https://www.legislation.gov.uk/asp/2016/3/contents.

[4] Please visit http://nai.uni.lu and log in with the credentials: smoking@nai.lu / nai.

The second step requires marking all legal terms in the text. In NAI, one can use for this purpose a rich, first-order language which contains propositions, arguments and variables and which can capture terms and their relationships. The user highlights a specific text fragment, clicks the «Term» button from the editor's options and chooses a logical term name to represent the text. For example, the word «adult» is annotated in the example using the term «condition_article1_adult», while the text «used as living accommodation» is annotated using the term «exception_article1_used_homecar». The list of annotations and their corresponding texts can be found when visiting the «Vocabulary» tab. If the user would like to place a relation between terms, i.e. to denote that the adult is driving a car, one can annotate the adult using the first-order term «adult(X)», whose meaning is that for every variable X, X is an adult. Similarly, one annotates the car with «car(Y)» and creates a relation between them using «drive(X,Y)». For the purpose of having a simple and clear example in the remaining of this section, we have focused on using propositional terms only.

While the user is free to choose any term to annotate the text, we recommend to use meaningful ones. The text «adult» is best annotated using a term containing the word «adult», while the fact that it is a condition can be captured in the name as well. The annotated text is then displayed in colors where the same terms are always annotated using the same colors.

Once all relevant legal terms have been annotated, the user can proceed with formalizing the legal interpretation. For this purpose, one needs to specify which part of a sentence is the condition, or an obligation, etc. In NAI, this is obtained by highlighting whole sentences which contain one or more annotations and then choosing a connective from the dropdown menu. The way of using each connective is displayed when hovering over it. For example, the connective «If / Then» expects two elements and its meaning is that if the first element turns out to hold, then it is expected that the second holds as well.

In the first sentence, there is a need to annotate what is an offence according to article 1. Logically, this can be obtained by saying that the term «offence» is equivalent to its definition. This definition contains the conditions in subsection 1 as well as the exceptions in subsection 2. We have, therefore, highlighted the whole two sentences of subsections 1 and 2 and chose the connective «Equivalence». This connective expects two arguments – the annotation of «offence» and of its definition. We use the «And» connective ,which can be used with any number of annotations, to annotate all the conditions.

The formalization of the text now reads as follows. The term «offence» is equivalent to the existence of all the remaining terms in the two sentences together. This is clearly not a correct interpretation. The two last terms are not conditions to the existence of an offence but are exceptions of it. We have captured that by highlighting these two terms and annotating them with the «Not» Connective. Meaning that an «offence» does not hold if these two terms are both holding. To capture the fact that both terms need to hold, as well as to adhere to the requirement of the «Not» connective to apply to a single term or sentence, we have further annotated the two with the «And» connective. We have thus finished formalizing subsections 1 and 2 of article 1. By clicking «Save» and visiting the «Formalization» tab, the user can see the logical formula [Libal/Pascucci 2019] which was generated according to the annotation. This formula will be later given to a theorem prover in order to allow automated reasoning.

The last subsection of article 1 explains the result of committing an offence according to subsection 1. There are different ways in which it can be interpreted. We have chosen to understand it as a strong permission for the state to penalize the offending person. The state is allowed to give a fine if and only if, an offence was committed. This can be obtained by highlighting the two legal terms and by choosing the «If / Then Strong Permission» connective. The difference between the «If / Then» family of connectives to the «Always / If» one is the purpose of each term. While in the first, we consider the first element as the condition and the second as the conclusion, these roles are reversed in then «Always / If» family of connectives.

During the formalization process, the user is encouraged to repeatedly use the correctness features which were discussed in Section 2.2. Both the consistency and the independency checks are instrumental for the generation of a correct formalization.

3.2. Resolving Legal Questions using the Query Editor

Once the user is confident that the formalization is faithful to the interpretation, she can trust it to resolve legal questions with regard to specific cases. In order to do that, she needs to switch the legislation editor with the query one. The demo account contains several such queries. In order to create a query, the user needs first to write the query text and then annotate it in a similar way to the way the legal text was annotated. As discussed in Section 2.1, there is one important difference. In queries, one also needs to denote which part of the query should be affirmed. This is done by highlighting this text and annotating it as the goal.

Consider the following legal question.

Case 1. *A client got a fine while driving his home car while smoking. His teen daughter was sitting next to him. Is there a case to appeal this decision?*

Here a user might want to check if there was an obligation in the law not to give the client a fine. In case it is true, an appeal should be successful. This can be achieved by first annotating all legal terms. In this example, we have annotated «offence» as the goal. The case annotation is still incorrect. The question which it tries to answer is not if the state has given a fine, but if the state was **allowed** to give one. We have therefore further used the «Permission» connective in order to capture that.

By clicking «Execute query», one gets that a conclusion cannot be drawn (the query is counter-satisfiable). The reason for that is because some of the conditions and exceptions are not used. Since there might be two different values for these conditions, the reasoning engine cannot determine which of the different conclusions holds. In this case, one can find the «Vocabulary» tab to be useful. After examining the vocabulary, the following information is obtained. There is a further condition – the car should be in public space – and one further exception – the car should also be used as a home car, and not only be designed as one. The client is required to share more information about the case.

Case 2. *The client adds further that he was indeed driving in public space. The home car though, was not used as a home car at the time. The client has removed the home facilities and is using the car for transportation of goods.*

The addition of the new annotations allows the execution of the query to terminate with a success. The policeman was indeed permitted to give the fine. The client could enjoy the exception of subsection (b), but he failed to use the car for accommodation. It seems better not to appeal the fine.

4. Summary and Discussion

In this paper, the NAI suite for legal reasoning is introduced. NAI can be used to formalize normative documents which can then be assessed using automated reasoning technology. The application of the annotation editor is depicted using a brief case study, where also the query answering functionalities are motivated.

The annotation editor constitutes the cornerstone of the NAI reasoning capabilities. While the process of hand-crafted annotating legal documents is currently time consuming and somewhat tedious, there are several benefits of this approach that should be highlighted: After the annotation is completed, there is an inherent one-to-one correspondence between the formalized code and the original natural language expressions. This correspondence can be utilized in several ways: First, the correspondence acts as implicit documentation of the given interpretation; this increases the transparency of formalized legal documents and allows for critical assessments of the formalized code. Secondly, the annotation procedure generates an explicit symbolic representation of an expert's interpretation of the legal document. This allows for well-grounded deductive reasoning procedures over such knowledge bases and improved explainability, reproducibility and transparency – which seems more challenging in inductive resp. predictive approaches used in machine learning (ML) based set-ups. Finally, the stored data link between the natural language text and the annotations enable the employment of ML techniques for improving the user experience. As an example, learning-based techniques can be integrated for automatically suggesting annotation skeletons of complex documents.

The NAI platform as presented in this paper is a prototype. Further work is required on both the tools and their underlying logical formalisms, in order to make the formalization of legal texts easier and more intuitive. Currently, the NAI suite supports an expressive deontic first-order language that can capture many scenarios which appear in legal texts; still important semantical aspects, e.g., the support of exceptions, temporal sentences, counter-factuals and arithmetic [Routen 1989], are ongoing work. Similarly, the currently supported deduction engine can already be used for many interesting tasks. For example, deduction can be used for compliance checking [Palmirani/Governatori 2018]. On the practical usability side, there is ongoing work to include natural language processing (NLP) techniques to the NAI front end that is planned to assist the annotation procedure. This way, annotations could be generated semi-automatically using suggestions, which can then be further refined or modified by the user. This allows for a hybrid approach in which the transparency of the symbolic annotation process is kept, while the time efficiency of the process in improved at the same time. Also, the generation of explanations for invalid queries is planned.

5. Literature

Aucher, Guillaume and Berbinau, Jean and Morin, Marie-Laure, Principles for a judgement editor based on binary decision diagrams. IfCoLog Journal of Logics and their Applications, 6(5):781–815, 2019.

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

Benzmüller, Christoph, Universal (Meta-)Logical Reasoning: Recent Successes, In Science of Computer Programming, volume 172, pp. 48–62, 2019.

Boella, Guido and Di Caro, Luigi and Humphreys, Llio and Robaldo, Livio and Rossi, Piercarlo and van der Torre, Leon, 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.

de Bruin, Hugo et al., 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.

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

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

Libal, Tomer and Pascucci, Matteo, Automated reasoning in normative detachment structures with ideal conditions. In Proc. of ICAIL, Montreal, pp. 63–72, 2019.

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

Otten. Jens, MleanCoP: A connection prover for first-order modal logic. In IJCAR 2017, pp. 269–276, 2014.

Palmirani, Monica and Governatori, Guido, Modelling legal knowledge for GDPR compliance checking. In Legal Knowledge and Information Systems: JURIX, volume 313, pp. 101–110. IOS Press, 2018.

Routen, Tom, Hierarchically organised formalisations. In Proceedings of the 2nd international conference on Artificial intelligence and law, pp. 242–250. ACM, 1989.

Stamper, Ronald, LEGOL: Modelling legal rules by computer, Computer Science and Law (1980): 45–71.

Sergot, M. J. and Sadri, F. and Kowalski, R. A. and Kriwaczek, F. and Hammond, P. and Cory, H. T., The British Nationality Act as a logic program, Communications of the ACM 29, no. 5 (1986): 370–386.