Explainable Rule Extraction via Semantic Graphs

Gabor Recski[1], Björn Lellmann[2,3], Adam Kovacs[1,4] and Allan Hanbury[1]

[1]TU Wien, Vienna, Austria
[2]SBA Research, Vienna, Austria
[3]Federal Ministry for Digital and Economic Affairs, Vienna, Austria (Since May 3, 2021)
[4]Budapest University of Technology and Economics, Budapest, Hungary

Abstract

We present an end-to-end system for extracting deontic logic formulae from legal text using a generic semantic parsing module and task-specific graph grammars, and for performing automated reasoning on the extracted formulae. The pipeline enables automated compliance checking and is applied to text documents of the zoning map of the city of Vienna. All components are released as open-source software, the full pipeline is showcased in an online demo.

Keywords

semantic parsing, information extraction, automated reasoning, deontic logic

1. Introduction

We present an end-to-end system for extracting deontic logic formulae from legal text using a generic semantic parsing module and task-specific graph grammars, and for performing automated reasoning on the extracted formulae. An overview of the pipeline is shown in Fig. 1. Plain text regulations are processed by a pipeline of domain-agnostic language processing tools, including a system for building syntax-independent concept graphs that represent the meaning of each sentence. These graphs serve as the input for a task-specific rule extraction module that maps them to deontic logic formulae, which in turn are used in an automated reasoning system. The proposed pipeline is applied to text documents of the zoning map of the city of Vienna[1], an exciting corpus of legal regulations whose highly structured nature renders it very well suited for formal approaches. In absence of a large-scale annotated corpus we evaluate our approach on a toy dataset of manually analyzed sentences that were selected to cover the most frequent attributes in the full dataset. Possible applications include automated compliance checking and question answering. The semantic parser and rule extractor components are both entirely rule-based, making our system an example of true explainable AI (XAI). Unlike in deep learning-based information extraction systems, extracted rules can be directly traced back to text patterns, making it straightforward to provide natural language explanations for decisions made based on them. This explainable nature also enables human-in-the-loop operation and provides safeguards against biased decision-making. Our main contributions are:

- Specification of a formal representation of deontic statements including those of the construction regulation domain for automated rule extraction and reasoning
- A preprocessed, structured corpus of sentences extracted from the zoning plan of the City of Vienna, a small subset of which is annotated manually with formal rule representations
- A grammar-based system for explainable rule extraction from semantic graphs, evaluated on the annotated corpus
- The adaption and extension of a general theorem prover to the reasoning domain, including natural language output
- System architecture and working prototype for an end-to-end system for rule extraction and automated reasoning from raw text

The paper is structured as follows. In Sec. 2 we review recent work on semantic parsing, automatic rule extraction, and automated reasoning for legal-tech applications, and present the dependencies of our pipeline: a task-independent semantic parser and a deontic logic prover. Sec. 3 presents the rule extraction method, Sec. 4 describes the architecture of the full pipeline. Sec. 5

Proceedings of the Fifth Workshop on Automated Semantic Analysis of Information in Legal Text (ASAIL 2021), June 25, 2021, São Paulo, Brazil.

gabor.recski@tuwien.ac.at (G. Recski); lellmann@logic.at (B. Lellmann); adam.kovacs@tuwien.ac.at (A. Kovacs); allan.hanbury@tuwien.ac.at (A. Hanbury)
0000-0001-5551-3100 (G. Recski); 0000-0002-5335-1838 (B. Lellmann); 0000-0001-6132-7144 (A. Kovacs); 0000-0002-7149-5843 (A. Hanbury)

[1]See https://www.wien.gv.at/flaechenwidmung/public/ for the map and https://www.data.gv.at/katalog/dataset/flachenwidmungs-und-bebauungsplan-plandokumente-wien for how to obtain the text documents (in German).

Figure 1: Overview of our pipeline. The solid rectangle indicates the newly contributed component and format, dashed rectangles mark existing tools that we modify or extend.

provides preliminary evaluation of our rule extraction method, Sec. 6 discusses next steps and draws some conclusions. All components of our system are available as open-source software[2]. The example application is showcased in an online demo[3].

2. Related work

The related work for the two main components of the system is presented: (i) semantic parsing for extracting logic expressions directly from legal text, and (ii) deontic logic for performing automated reasoning on the extracted formulae.

2.1. Semantic parsing

Semantic parsing is the task of automatically mapping natural language text to a formal representation of its meaning. Most contemporary architectures for solving information extraction tasks do not perform semantic parsing, instead relying on models which directly encode the correspondence between natural language text and some set of task-specific structures such as labels, sequences, attribute-value structures, etc. These models are primarily built using machine learning methods, whose performance is dependent on the quality and quantity of available training data and whose decisions are difficult to interpret and prone to bias. Rule-based models, on the other hand, require considerable expert effort to build and maintain and can still be difficult to adapt to changes in the task definition. The architecture we propose tackles the information extraction task in two steps: the mapping of natural language text to task- and domain-independent meaning representations (semantic parsing) followed by a task-specific information extraction step that operates on these representations. In this section we give a brief overview of common approaches to semantic parsing and of the representation framework used by our pipeline.

Unlike other forms of automated linguistic annotation such as part-of-speech tagging or syntactic parsing, semantic parsing is not in any way standardized in the natural language processing (NLP) community. Even those few frameworks that have recently attracted growing interest in the task are rarely used as intermediate representations in NLP pipelines. Abstract Meaning Representations (AMRs) [1] represent sentence meaning as directed graphs of words, but do not provide a model of word meaning and are highly English-specific, not intended as a language-independent framework of meaning representation. The task of parsing raw text to AMR graphs has recently attracted growing interest and is usually performed using deep neural networks [2, 3] trained on annotated corpora, also called sembanks. Universal Conceptual Cognitive Annotation (UCCA) [4] takes a language-agnostic approach to meaning representation, modeling sentence meaning with directed acyclic graphs (DAGs) representing *scenes* evoked by predicates. Top UCCA parsers also rely on manually annotated corpora and neural networks [5, 6, 7, 8]. Since these frameworks do not provide a generic parsing algorithm, building representations for a new language and/or new domain would require the manual compilation of large annotated datasets that could be used to train end-to-end machine learning models. For the pipeline presented in this paper we choose the language-independent `4lang` framework [9], for which a robust parsing method [10] with an open-source implementation[4] [11] is also available.

The `4lang` framework represents the meaning of both words and larger units like phrases and sentences as directed graphs of concepts. The representation is syntax-independent, concepts do not have types such as part-of-speech or even as predicate and argument. A key feature of `4lang` graphs that enables uniform treatment of syntactically different constructions is the 0-relation ($\xrightarrow{0}$), a single representation for a range of closely related semantic relationships such as the $\texttt{IS}_\texttt{A}$ relationship (e.g. *roof* $\xrightarrow{0}$ *covering*), attribution (e.g. *sidewalk* $\xrightarrow{0}$ *paved*), and predication (e.g. *platform* $\xrightarrow{0}$ *extend*). `4lang` graphs can be built from raw text automatically using a rule-based system that uses Universal Dependencies (UD) [12] as an intermediate step. UD trees encode grammatical relations (dependencies) between pairs of words in a sentence — an example of such an analysis is shown in Fig. 2, described later. UD parsers are available for dozens of languages, in this pipeline we use the stanza package[5] [13]. The transformation of UD trees into `4lang` graphs is based on a small set of rules described in [10] and implemented by [11] as parsing and decoding of an Interpreted Regular Tree Grammar (IRTG) [14], a formalism that we also use in this work for implementing the rule extraction

[2] https://github.com/recski/brise-plandok
[3] https://ir-group.ec.tuwien.ac.at/brise-extract
[4] https://github.com/adaamko/wikt2def
[5] https://stanfordnlp.github.io/stanza

mechanism that maps `4lang` semantic graphs to trees of attributes (see Sec. 3.2). Most rules map a single UD edge between two content words to `4lang` edges connecting the corresponding concepts (e.g., in the example in Fig. 2, the relations `amod` and `nsubj : pass` are mapped to a 0-edge and a 2-edge, respectively).

2.2. Automated reasoning and deontic logic

The investigation of automated reasoning methods in the legal domain has a long history, see, e.g., the seminal [15]. More recently, automated reasoning methods have been considered for legal texts or company regulations [16, 17], and a large number of reasoning frameworks and tools are available, see, e.g., [18, 19] for an overview. Following the approach in [17], here we consider a general and formalism-independent representation of the regulations, which can be translated into different frameworks. Fortunately, the structure of the intended application, the regulations of the zoning map of Vienna, is relatively clear, and mostly does not require advanced features of the formal language such as nested deontic operators in the assumptions [16] or macros [20].

The specific reasoning engine used in this paper, the theorem prover BRISEprover[6], is an extension and modification of the theorem prover deonticProver2.0[7] developed in [21] for reasoning with assumptions in *dyadic deontic logic*. In this logical framework, propositional logic is extended with dyadic *deontic operators* obl, for and per. Formulae $\mathsf{obl}(A, B)$, $\mathsf{for}(A, B)$ and $\mathsf{per}(A, B)$ are read as "it is obligatory that A given B", "it is forbidden that A given B" and "it is permitted that A given B", respectively. As the first extension considered here we extend the language with *predicate symbols* to capture properties, e.g., "building height at most 9 metres", in atomic formulae, e.g., $\texttt{buildingHeightMax}(9)$. Note that it would be straightforward to include an additional argument representing the subject, i.e., which building has a height of at most 9 metres. Since in our case this is clear from the context we simplify the representation by assuming the subject is always the same. The prover decides derivability from a set of *factual* and *deontic assumptions*, i.e., non-deontic and non-nested deontic formulae respectively. The reasoning engine supports the *specificity principle* in the form that more specific deontic assumptions override less specific conflicting ones. E.g., the assumption $\mathsf{obl}(\texttt{buildingHeightMax}(9), \texttt{facingStreet})$, stating that buildings facing the street must have a maximal height of 9 metres, overrides the less specific assumption $\mathsf{obl}(\texttt{buildingHeightMin}(10), \top)$, stating that under the always true condition $\top$ buildings must have a minimal height of 10 metres. Due to this property the reasoning process is *non-monotone*: While from the single assumption $\mathsf{obl}(\texttt{buildingHeightMin}(10), \top)$ we can derive the formula

$$\mathsf{obl}(\neg\texttt{buildingHeightExactly}(8), \texttt{facingStreet}),$$

stating that the height of buildings facing the street must not be exactly 8 metres, we cannot derive the same formula with the additional assumption $\mathsf{obl}(\texttt{buildingHeightMax}(9), \texttt{facingStreet})$ anymore. As a second modification from the original deonticProver2.0, here we consider a different conflict resolution mechanism, resulting in particular in higher efficiency of the implementation. Reasoning in this logic is implemented via backwards proof search in a sequent system with underivability statements, both in the system from [21] and in a slight modified reasoning engine. See *op. cit.* for the details of the original system and Sec. 4.4 for the modifications.

We are thankful to one of the reviewers for bringing additional relevant literature to our attention [22, 23, 24, 25, 26]. Unfortunately, space and time constraints prevented a detailed comparison for the final version.

3. Rule extraction

The pipeline presented here takes as its input raw text documents containing regulations of the zoning map of the City of Vienna, builds representations of their meaning using the `4lang` system (see Sec. 2.1), uses the resulting semantic graphs to extract the legal content of regulations and makes them available to the prover (see Sec. 2.2), which verifies whether some statement is derivable given a set of assumptions. The full architecture is described in Sec. 4, we now present the novel rule extraction component and its interfaces to semantic parsing and automated reasoning.

3.1. Representation

Since we do not want to commit to modeling regulations in a particular formalism, and to facilitate the integration of different reasoning engines we first convert the legal content of the regulations into a *generic representation*. For this we assume that a *deontic regulation*, i.e., a regulation stating an obligation, prohibition or permission, is comprised of the following parts:

- *Modality*: This states whether the regulation is an obligation, a prohibition or a permission;
- *Content*: The content of the regulation, i.e., *what* is obligatory / prohibited / permitted;
- *Conditions*: The conditions of the regulation stating when the regulation applies;

[6] See http://subsell.logic.at/bprover/briseprover/
[7] See http://subsell.logic.at/bprover/deonticProver/version2.0/

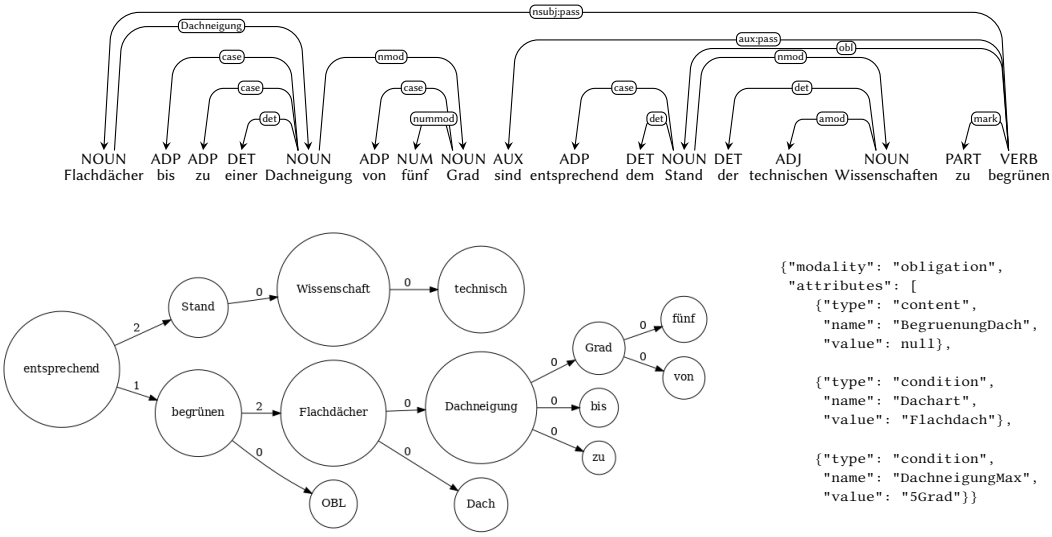

Figure 2: Universal Dependency analysis, 4lang semantic graph, and formal rule representation for the sentence *Flachdächer bis zu einer Dachneigung von fünf Grad sind entsprechend dem Stand der technischen Wissenschaften zu begrünen.* 'Flat roofs with a pitch not exceeding 5 degrees must be greened using state of the art technologies.'

- *ConditionExceptions*: Possible exceptions to the conditions, stating when the regulation does not apply. E.g., in the regulation "Flat roofs should be green roofs, unless they are glass roofs", the "glass roofs" is an exception to the condition.
- *ContentExceptions*: Possible exceptions to the content. E.g., in the regulation "Windows are prohibited except for portholes" the "portholes" are an exception to the content.

This level of granularity seems to capture the necessary details of the sentences found in the documents from the city of Vienna zoning map while being flexible enough to permit translation into different frameworks like dyadic deontic logic, defeasible deontic logic [16], argumentation based approaches [27] or input output logic [28]. Concretely, we represent this structure as a JSON object

$$\{\text{"}\texttt{modality}\text{"} : \texttt{Modality}, \text{"}\texttt{attributes}\text{"} : \texttt{List}\}$$

where the key "modality" takes one of the values "obligation", "prohibition", "permission", and where `List` is an array containing attributes of the following form:

$$\{\text{"}\texttt{name}\text{"} : \texttt{Name}, \text{"}\texttt{value}\text{"} : \texttt{Value}, \text{"}\texttt{type}\text{"} : \texttt{Type}\}.$$

Here `Name` is the name of the attribute, `Value` is its value, and `Type` is one of "Content", "Condition", "ConditionException", "ContentException". We obtained the attribute names via collaboration with domain experts from the City of Vienna Baupolizei and verified that the structure is appropriate by manually annotating several hundred sentences from the documents of the zoning map.

As an example, the generic representation of the deontic regulation "Flat roofs should be green roofs unless they are glass roofs" is given by:

```
{ "modality" : "obligation",
  "attributes" : [
    { "name" : "roofType","value" : "flatRoof",
      "type" : "condition"},
    { "name" : "greenRoof","value" : NIL,
      "type" : "content"},
    { "name" : "roofType","value" : "glassRoof",
      "type" : "conditionException"}]}
```

Here we modelled both flat roofs and glass roofs as roof types, while modelling the property of being a green roof as an atomic propositional statement because the latter in the documents corresponds to a more complex proposition.

Note that in contrast to, e.g., the approach in [17] at this stage we do *not* commit to a particular modelling of exceptions by negation-as-failure or negated conditions. This retains the flexibility of the general format necessary for subsequent specification into a large number of different formalisms. There are of course some limitations inherent in this representation, in particular we do not model negation. See Sec. 6 for a more detailed discussion.

3.2. Extraction

The mapping from semantic graphs to the rules described is implemented in two steps. An IRTG grammar similar to that of the semantic parsing system (see Sec. 2.1) is used to extract all attributes, values, and expressions of

modality that occur within a single sentence. A simple heuristic then matches these elements with each other in order to create the generic rule representations described in the previous section. The mapping we establish is between patterns in generic semantic graphs and formal rules. This two-step approach is an arbitrary simplification that makes implementation simpler and more flexible. We shall now describe the approach using some examples and point out some of its current limitations. The first component of our rule extraction system is an IRTG grammar mapping `4lang` graphs to lists of strings representing attribute names (e.g. `DachneigungMax` 'maximal roof pitch'), modalities (e.g. `OBL` 'obligation'), as well as numbers and units of measurement that may be interpreted as attribute values (e.g. `5` and `m`). We shall refer to this grammar as `fl_to_attr`. The heuristics for matching values and modalities to attribute names, described later in this section, can only disambiguate between multiple solutions if it is informed about the positions of patterns relative to each other. Hence we use a Tree Grammar as the output interpretation, which allows us to represent these strings as leaves of a tree that resembles the order in which they were recognized, corresponding to steps of composing the `4lang` graph from subgraphs. An example of this mapping is presented in Fig. 3. Each IRTG rule encoding the correspondence between a `4lang` subgraph and an attribute or tree of attributes is a mapping between rule applications in an s-graph algebra [29] and a tree algebra. S-graphs are graphs whose nodes may be marked by special labels called *sources* and the s-graph algebra's core operation `merge` creates new s-graphs by taking the union of its arguments but merging nodes that have the same source. We now illustrate this mechanism with a simple example, for a more detailed introduction to s-graph algebras and their application to semantic parsing the reader is referred to [29]. The IRTG rule presented in Fig. 4 defines two binary operations, to be performed in parallel on corresponding pairs of s-graphs and trees. The operation of the `4lang` interpretation merges two graphs along their root nodes with the two nodes of the edge *Gebäudehöhe* $\xrightarrow{0}$ *maximal* to create a single graph, while the operation of the `attr` interpretation merges the two attribute trees with each other and subsequently with a tree of two nodes (`OBL`, `GebaeudeHoeheMax`). S-graph algebras use three types of operations: the `merge` operation merges two graphs on nodes with matching sources while the `rename` and `forget` operations can be used to change or delete sources. In the example rule in Fig. 4 the two argument graphs are renamed so that their `root` sources become `src` and `tgt`, and these sources need to be deleted after the `merge` operation.

Once all relevant strings have been extracted from the semantic graph, the next step is to build the rule representation by heuristically matching modalities as well as potential attribute values to attribute names. We illustrate this process using an example with multiple attributes; consider the following sentence fragment, a subordinate clause of a longer regulation: *bei einer Straßenbreite ab 10 m entlang der Fluchtlinien Gehsteige mit einer Breite von mindestens 2,0 m herzustellen sind.* 'in case of a street width of 10 m or more, sidewalks with a width of at least 2.0 m are to be constructed along the alignment lines.'. The pipeline described so far will extract from this sentence three attribute names (`StrassenbreiteMin` 'minimum street width', `GehsteigbreiteMin` 'minimum sidewalk width', `AnFluchtlinie` 'along the alignment line'), two numbers (10, 2.0), two occurrences of the unit of measurement *m*, and the modality `OBL` 'obligation' (the latter based on the word form *herzustellen* composed of the verb *herstellen* 'construct, produce' and the infinitive marker *zu*). These eight elements are organized in a tree structure according to the order in which they appeared in the IRTG derivation of the corresponding semantic graph, shown in Fig. 5. The tree of attributes is stored in a custom data structure that in every node stores the length of the shortest path between any pair of attributes below that node. This means that by querying the root of the tree we can retrieve for any attribute a list of all other

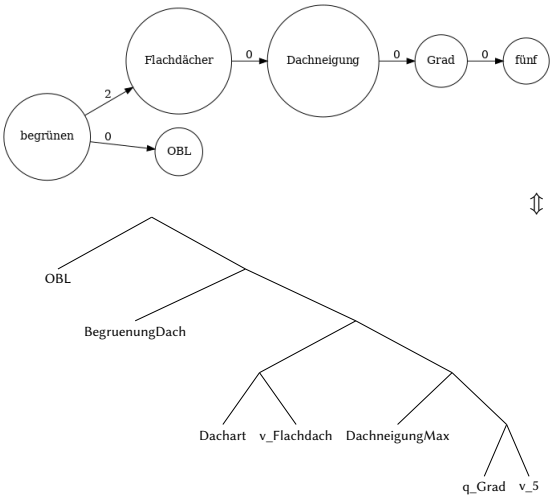

Figure 3: Example mapping implemented by the `fl_to_attr` grammar for the sentence in Figure 2.

```
E -> a_gebaeudehoehe_maximal(E, E) [100]
[fl] f_src(f_tgt(merge(r_src(?1), merge(r_tgt(?2),
"(u<src> / Gebaeudehoehe :0 (v<tgt> / maximal))"))))
[attr] *(*(?1, ?2), *("OBL", "GebaeudeHoeheMax"))
```

Figure 4: Example rule of the `fl_to_attr` IRTG grammar. The first interpretation line is wrapped for readability, the operations are explained in the text.

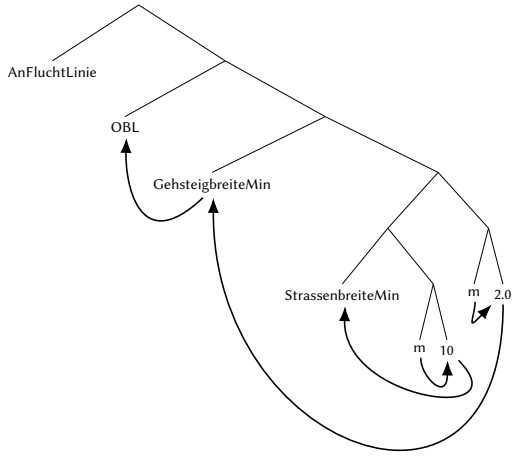

Figure 5: Example of attribute matching for the sentence *. . . bei einer Straßenbreite ab 10 m entlang der Fluchtlinien Gehsteige mit einer Breite von mindestens 2,0 m herzustellen sind.* 'in case of a street width of 10 m or more, sidewalks with a width of at least 2.0 m are to be constructed along the alignment lines.'

attributes ranked by their relative distance in the tree. We first match all units of measurement to the nearest value in the tree, allowing each value to be associated with at most one unit of measurement. Next, all non-boolean attributes are matched to the nearest value, using a greedy algorithm: all possible attribute-value pairs are sorted by their relative distance in the tree, the pair with the shortest path is stored as a match, its members are removed from the lists of attributes and values that are still to be matched, and this step is repeated until at least one of the lists becomes empty. For example, in Fig. 5 the attribute `StrassenbreiteMin` 'minimum road width' is paired with the value `10`, since they are the closest of any pairs of attribute and value in the tree (the attribute `AnFluchtLinie` is excluded from this process because it is listed as a boolean attribute). In the second step the attribute `GehsteigbreiteMin` 'minimum sidewalk width' is matched with the only remaining value, `2.0`. Finally, the type of each attribute must be detected, i.e. it must be determined whether an attribute is a `condition` of the rule, part of the `content`, or an exception to either one of these (`contentException`, `conditionException`, see Sec. 3.1). Some attrbutes are explicitly listed as always being of type `condition`, e.g., `Planzeichen` and `Widmung` which refer to the ID and designation of an area. Next, the extracted modality elements `OBL`, `FOR`, `EXC` are matched to the nearest of the remaining attributes, which are in turn determined to be part of the `content` (in case of `FOR` and `OBL`) or a `conditionException` (in case of `EXC`). Finally, all remaining attributes are given the type `condition`.

These simple heuristics, which are already capable of correctly matching attributes to their values and for distinguishing between the roles each attribute plays in a rule, allow us to keep the grammar simpler than it would be if it was to directly generate structures like our generic rule representations. The IRTG rules currently used (and exemplified in Fig. 4) simply represent one-to-one correspondences between `4lang` subgraphs and strings, but the underlying Regular Tree Grammar only uses a single nonterminal symbol, i.e. rules are not sensitive to which other rules were used to construct their arguments. It would be quite straightforward to introduce non-terminal symbols representing attribute names, numbers, measurement units, and modalities, so that the IRTG itself would enforce the structure that is currently built in a postprocessing step. The tree in Figure 5 resembles the order in which patterns corresponding to each element were found in the semantic graph, which in turn correspond to disjoint subgraphs of the semantic representation that are each connected to the concept *herstellen* and roughly correspond to fragments of the original sentence such as *Gehsteig mit einer Breite von mindenstens 2,0 m herstellen* 'construct sidewalk with a width of at least 2.0 m', *bei einer Straßenbreite ab 10 m* 'with a road width of at least 10 m', *entlang der Fluchtlinien* 'along the alignment lines', etc. Here we limit our grammar to the task of understanding each of these patterns independently because this proves sufficient for our purposes of constructing formal rules from sentences of the zoning map of the City of Vienna, which tend to be in a one-to-one correspondence with rules of the general structure described in Sec. 3.1. I.e., we greatly simplify our task by exploiting the fact that authors of this piece of legislation rarely express a rule in multiple sentences or incorporate several rules in a single sentence. A notable exception is when some conditions such as the ID or designation of the areas that a rule refers to are not repeated in every sentence within the same section. These conditions are propagated by a simple inheritence mechanism that assumes the values of such attributes to hold within a single section (see Sec. 4 for how section boundaries are detected).

3.3. Specification to dyadic deontic logic

The extracted rules in the generic format are then translated into the language of dyadic deontic logic. We chose this particular framework because the sentences of the zoning map of Vienna exhibit a very clear deontic structure, in contrast, e.g., to the largely definitional character of the British Nationality Act investigated in [15]. The translation is dependent on the modality in the following way. Given a rule representation

{"`modality`" : `Modality`, "`attributes`" : `List`}

let

$$\texttt{Cnd} := \texttt{cnd}_1(\texttt{cndV}_1) \wedge \cdots \wedge \texttt{cnd}_n(\texttt{cndV}_n)$$
$$\texttt{CntEx} := \texttt{ctEx}_1(\texttt{ctExV}_1) \vee \cdots \vee \texttt{ctEx}_m(\texttt{ctExV}_m)$$
$$\texttt{CndEx} := \texttt{cdEx}_1(\texttt{cdExV}_1) \vee \cdots \vee \texttt{cdEx}_\ell(\texttt{cdExV}_\ell)$$

where the $\texttt{cnd}_i$ are all the attributes occurring with type "condition" in `List`, and the $\texttt{cndV}_i$ are their respective values, and similarly for `ctEx` for type "contentException" and `cdEx` for type "conditionException". For the content, let further

$$\texttt{Cnt} := \texttt{cnt}_1(\texttt{cntV}_1) \wedge \cdots \wedge \texttt{cnt}_m(\texttt{cntV}_m)$$
$$\texttt{Cnt}_{\texttt{for}} := \texttt{cnt}_1(\texttt{cntV}_1) \vee \cdots \vee \texttt{cnt}_m(\texttt{cntV}_m)$$

Where again the $\texttt{cnt}_i$ are the attributes of type "content" and the $\texttt{cntV}_i$ their respective values. The translation of a rule representation with modality "obligation" then is:

$$\mathsf{obl}(\texttt{Cnt} \vee \texttt{CntEx}, \texttt{Cnd}) \wedge \mathsf{per}(\neg(\texttt{Cnt} \vee \texttt{CntEx}), \texttt{CndEx})$$

The translation of a rule with modality "prohibition" is:

$$\mathsf{for}(\texttt{Cnt}_{\texttt{for}} \wedge \neg\texttt{CntEx}, \texttt{Cnd}) \wedge \mathsf{per}(\texttt{Cnt}_{\texttt{for}} \wedge \neg\texttt{CntEx}, \texttt{CndEx})$$

For the modality "permission" the translation is:

$$\mathsf{per}(\texttt{Cnt} \vee \texttt{CntEx}, \texttt{Cnd}) \wedge \mathsf{for}(\texttt{Cnt} \vee \texttt{CntEx}, \texttt{CndEx})$$

Note that this translation commits to formalising, e.g., condition exceptions to obligations or prohibitions as additional permissions. Of course this is by no means the only possible translation: we could have chosen to embed the condition exception explicitly in the condition of the resulting formula. This choice is due to the fact that it facilitates the derivation of a general statement like "Flat roofs should be green roofs" from "Flat roofs should be green roofs unless they are glass roofs" in the particular logic used in the prover. In particular, when checking whether a flat roof in general should be a green roof we do not need to explicitly state that none of the condition exceptions are satisfied, in line with the standard approach in non-monotonic logic and default reasoning [30].

4. Architecture

We describe the system architecture of the full pipeline that takes raw text documents as input, builds semantic graphs using the system described in Section 2.1, extracts rules using our method presented in Section 3.2, maps them to deontic logic formulae as described in Section 3.3 and provides them as input to the prover (see Section 2.2). All components of our system are available as open-source software[8] under an MIT license and the end-to-end pipeline is showcased in an online demo[9] integrating all of them.

[8] https://github.com/recski/brise-plandok
[9] https://ir-group.ec.tuwien.ac.at/brise-extract

4.1. Preprocessing and segmentation

The input to our pipeline consists of PDF documents downloaded from the public website of the City of Vienna. Each PDF document contains regulations pertaining to one zoning area (*Plangebiet*), indicated by a four-digit ID. We discard the fraction of documents that are scanned images of printed documents and do not contain machine-readable text data ($253/1431 = 17.7\%$) – we could include these in our experiments by running optical character recognition (OCR). PDF documents are then converted to plain text using the `pdftotext` utility, part of the open-source Poppler library[10]. We use the `−layout` option of the tool to maintain page layout in the output text file, this greatly simplifies the subsequent extraction of document structure. Next we use a small set of regular expressions to establish section boundaries and extract section numbers from the text. Section numbering often makes use of several levels (e.g. 1, 1.1, 1.1.1, etc.), but this is not consistent across documents, therefore we only consider top-level sections in subsequent steps that are sensitive to section boundaries. Besides the inheritance mechanism described in Section 3.2, this decision is crucial for sentence segmentation, the next step in our pipeline, for which we use a customized version of the German sentence splitting model of the stanza[11] [13] library. The output of the standard model is postprocessed to undo sentence splits that have been made in error (e.g. those after periods following abbreviations characteristic of legal text) and also those made after colons (:) that separate a predicate from its object(s), such as in the text *Für die mit BB4 bezeichneten Grundflächen wird bestimmt: Die Errichtung von Gebäuden mit einer maximalen Gebäudehöhe von 8 m ist zulässig.* 'For areas marked BB4 it is determined: construction of buildings with a maximum building height of 8 m is allowed.'. The custom sentence segmentation step is followed by stanza's default German pipeline (`de − gsd`, stanza model version 1.1.0) for tokenization, part-of-speech (POS) tagging and universal dependency parsing.

4.2. Semantic parsing

The next step in our pipeline is to construct semantic graphs from each sentence. The rule extraction algorithm described in Section 3 assumes that all relevant information present in the input text is available in the semantic graph that is the output of the generic semantic parsing pipeline described in Section 2.1. To ensure that this is the case some minor modifications of the semantic parsing algorithm were also necessary. First, we introduced a small set of rules in the grammar mapping Universal Depen-

[10] https://gitlab.freedesktop.org/poppler/poppler
[11] https://stanfordnlp.github.io/stanza

dency representations to semantic graphs for common words expressing negation and modality. The lemmas *nicht* and *kein* trigger the addition of the `NEG` element to the `4lang` graph, *dürfen* 'may' and *zulässig* 'permitted' are mapped to `PER`, *untersagen* 'prohibit' and *unzulässig* 'not permitted' to `FOR`, and *müssen* to `OBL`. Additionally, the German construction consisting of the particle *zu* followed by the infinitive form of a verb must also trigger the `OBL` element, since it can express modality without any additional linguistic elements. This latter rule is implemented by two mechanisms, one that looks for the lemma *zu* with the universal part-of-speech tag (UPOS) `PART`, the other for the language-specific part-of-speech tag (XPOS) `VVIZU` marking verbs that contain the particle as an infix (e.g. *herzustellen* from *herstellen* 'create, produce'. While even the most rudimentary treatment of the semantics of German modal expressions would go beyond the simplicity of such a simple categorization (and the scope of this work), in practice this small enhancement of the semantic representation of the input text was sufficient to allow for the detection of modality by the rule extraction mechanism. Finally we also added an ad-hoc rule for detecting exceptions: the presence of the word *sofern* and *soweit*, both roughly equivalent to the English conjunction 'provided' and introducing a clause that limits the applicability of a previous statement, triggers the addition of an element `EXC` to the semantic graph which is then also available for processing by the rule extraction mechanism.

4.3. Rule extraction

We now describe the implementation details of the two-step rule extraction method presented in Section 3.2. The output of the semantic parser, which serves as the input to rule extraction, is a single directed graph for each input sentence, generated by an Interpreted Regular Tree Grammar from Universal Dependency structures (see Section 2.1 for details). For recognizing subgraphs and mapping them to attributes we also use an IRTG over an algebra of s-graphs, this allows us to pipe the output of the semantic parser directly into our rule extraction grammar. For each `4lang` graph we dynamically generate a unique grammar. The static set of rules encoding the correspondence between generic semantic structures and task-specific attributes is extended with empty terminal rules for each concept in the input graph, this ensures that the entire graph can be constructed by a sequence of operations that is derivable by the underlying RTG and thus the object can be parsed by the IRTG. The output interpretation of the IRTG is an algebra of trees, whose leaves are the individual strings that we use to construct rules in a subsequent step. The trees resemble the order in which these strings (names and values of attributes, modal elements, units of measurement) have been added to the output in parallel to the construction of the semantic graph, it is this additional information that allows us to implement the matching heuristics described in Section 3.2. For parsing of `4lang` graphs and generation of attribute trees with these IRTGs we use the open-source `alto`[12] library, which also implements s-graph algebras and tree algebras. The `alto` system also supports probabilistic parsing with weighted grammars, and we rely on rule weights to ensure that rules which map subgraphs to attributes always take precedence over the 'empty' rules that are only added to the grammar to ensure that the full graph is derivable. In those few cases when more than one such 'content' rule matches the same subgraph, precedence is given to rules that cover larger substructures. The trees output by the IRTG parser serve as the input to the heuristic construction of rules described in the previous section. Finally, rules are converted from the generic (JSON) format to the language of dyadic deontic logic, as described in Section 3.3.

4.4. The prover

The final step in our pipeline consists of an exemplary reasoning mechanism to draw inferences from the extracted rules. This step is based on our adaption[13] of the generic theorem prover deonticProver2.0[14] which implements backwards proof search in a sequent system for a dyadic deontic logic extended with rules for defeasibly reasoning from deontic assumptions [21]. Apart from specifying the prover to the language obtained from the examples we needed to further modify it in two ways. First, in order to be able to handle attributes with numerical arguments, such as `DachneigungMax` for the maximal angle of the roof, or with strings as argument, such as `Dachart` for the roof type, we extended the prover and the underlying reasoning system to handle atomic propositions with arguments. In addition, we added *ground sequents*, i.e., structures which can be used as leaves in a derivation, corresponding to basic properties of measure-like attributes with natural numbers as values: Where `msr` is a basic attribute for a measure such as `Dachneigung`, we considered a triple consisting of the attributes $\texttt{msrGenau}(n)$, $\texttt{msrMin}(n)$ and $\texttt{msrMax}(n)$, expressing the facts that `msr` is exactly n, at least n, or at most n, respectively. The relations between these three attributes are given by:

- $\texttt{msrGenau}(n) \rightarrow \texttt{msrMin}(n) \wedge \texttt{msrMax}(n)$
- $\texttt{msrMin}(n) \rightarrow \texttt{msrMin}(m)$, where $m \leq n$
- $\texttt{msrMax}(n) \rightarrow \texttt{msrMax}(m)$, where $n \leq m$
- $\texttt{msrMax}(n) \rightarrow \neg\texttt{msrMin}(m)$, where $n < m$

The ground sequents added to the prover then absorb basic reasoning on these axioms, so that, e.g., the formulae

$$\neg(\texttt{DachneigungGenau}(n) \wedge \texttt{DachneigungGenau}(m))$$

[12] https://github.com/coli-saar/alto
[13] https://github.com/blellmann/BRISEprover
[14] http://subsell.logic.at/bprover/deonticProver/version2.0/

are derivable for $n \neq m$, stating that the exact angle of a roof does not have two different values.

Second, and more significantly, to be more in line with other approaches in the area of deontic reasoning such as [31] as well as for efficiency reasons we modified the mechanism how the prover handles *specificity reasoning* when reasoning from deontic assumptions. To illustrate, assume the deontic assumption

$$\mathsf{obl}(\texttt{DachneigungMax}(5) \wedge \texttt{BegruenungDach}, \texttt{Plangeb}(7181)) \tag{1}$$

stating that the maximal angle of the roof must be 5 degrees and the roof must be green under the condition that the building is in zone 7181. This would be partially overruled by the additional *more specific* assumption

$$\mathsf{obl}(\neg\texttt{BegruenungDach}, \texttt{Plangeb}(7181) \wedge \texttt{Planzeichen}(\texttt{BB1})) \tag{2}$$

stating that roofs in areas of zone 7181 marked with the label *BB1* on the map must be not green roofs. The latter assumption is considered more specific than (1) because its condition $\texttt{Plangeb}(7181) \wedge \texttt{Planzeichen}(\texttt{BB1})$ strictly implies the condition $\texttt{Plangeb}(7181)$ of assumption (1). In deonticProver2.0 the assumption (1) could still be used to infer obligations from the part of its content not in conflict with the content of the more specific assumption (2), such as

$$\mathsf{obl}(\neg\texttt{DachneigungGenau}(7), \texttt{Plangeb}(7181) \wedge \texttt{Planzeichen}(\texttt{BB1})) \tag{3}$$

stating that in areas of zone 7181 marked with the label *BB1* the exact angle of the roof must not be 7 degrees. In our prover we changed this behaviour so that any assumption which is a in conflict with a more specific applicable one cannot be used to derive any obligations. Thus (disregarding prohibition and permission operators for the sake of exposition) to check whether (3) is derivable we now check whether there is an assumption $\mathsf{obl}(A, B)$ such that

1. $A \rightarrow \neg\texttt{DachneigungGenau}(7)$ is derivable
2. $\texttt{Plangeb}(7181) \wedge \texttt{Planzeichen}(\texttt{BB1}) \rightarrow B$ is derivable
3. there is no applicable and more specific assumption conflicting with $\mathsf{obl}(A, B)$, i.e., there is no $\mathsf{obl}(C, D)$ such that
   a) $\texttt{Plangeb}(7181) \wedge \texttt{Planzeichen}(\texttt{BB1}) \rightarrow D$ and $D \rightarrow B$ are derivable
   b) $C \rightarrow \neg A$ is derivable

Crucially, the "no-conflict" check in item (3b) above only needs to be performed between two assumptions and not between the formula to be proved and an assumption. This means that instead of checking for conflicts many times redundantly in the search for a derivation (as is done in deonticProver2.0) it suffices to perform this check once in a preprocessing stage, store for every deontic assumption the list of conflicting ones, and only check that none of the assumptions in this list is applicable and more specific during the actual computation. In our experiments this increased efficiency was necessary for reasoning with a non-trivial number of deontic assumptions. To compare the two reasoning methods the user can switch between the original ("classic") and modified ("modern") versions on the web interface[15] for the prover. For the sake of simplicity the web interface for the whole pipeline only uses the modified version.

Originally, for derivable input deonticProver2.0 outputs a pdf file with a derivation in the calculus. However, since the derivations can become rather large (even breaking the maximal limit on object size in TeX) and the average user might not be acquainted with the specific formalism used in the prover, we further extended the output module with an option to print the derivation as an explanation in pseudo-natural language. Explanations can be unfolded step by step by clicking on a button labelled "Why?" after the "The statement ... is derivable." output. In unfolding the explanation, propositional steps are skipped by default to reveal the crucial deontic statements and assumed facts used there. These intermediary steps can additionally be unfolded by clicking on a "Why does it follow from the above?" button. In the demo the user can select the output format.

We stress again that here our prover serves mainly as an example for a possible reasoning mechanism and that we do not claim that the underlying logic is necessarily the most appropriate. For this reason we also defer the theoretical details of the modifications of the underlying sequent system to a forthcoming companion paper.

5. Evaluation

The rule systems presented in Sec. 3.2 were developed based on a small annoted sample of sentences from the zoning plan of the City of Vienna. In order to establish a representative sample, we started by estimating the distribution of attributes in the entire corpus by manually labeling the sentences of 10 randomly selected documents with the attributes they mention (either as condition or content). This sample contains 344 mentions of attributes in 193 sentences (as well as 118 sentences without attribute mentions, mostly from the preambles). The number of unique attributes in the sample is 84, but 193 of the 344 instances (56%) come from the 16 most frequent attributes. We then chose 6 sentences from this sample that together contain mentions of 7 of these 16 attributes, including the 3 most frequent ones (`GebaeudeHoeheMax`, `AbschlussDachMax`, `GebaeudeHoeheArt`) that are alone responsible for 17%

[15] http://subsell.logic.at/bprover/briseprover/

of all attribute mentions in the larger sample. We annotated these 6 sentences with the full representation of all rules stated by them and developed our rule extraction system to achieve perfect performance on this toy corpus. Both this fully annotated set and the larger sample of 10 documents annotated for attribute mentions only are released along with the software[16]. While our method of selecting the sentences for the toy corpus ensures that the attribute extraction step of our method has high coverage (recall above 51% with a precision above 93% on the sample of 10 documents and 344 attribute instances), this cannot be considered as quantitative evaluation of the full rule extraction pipeline. The limited amount of annotated data also does not permit any conclusions about the effect of errors in syntactic parsing made by the stanza model, but our assumption that this should not become a bottleneck for such standard text is reinforced by the fact that we did not observe any such errors in our sample. A larger-scale annotation of attribute mentions is currently in progress.

6. Discussion

In this article we have presented a system for extracting formal rules from legal text using generic semantic parsing and domain-specific pattern-matching, and converting them to deontic logic for use in an automated reasoning system. All components of the pipeline, including those contributed in this paper, are made available as open-source software under the MIT license, for unrestricted use in future applications. Unlike machine learning based information extraction systems, our rule extraction model is fully explainable and serves as an example for a specific application of semantic parsing to domain-specific information extraction. While the semantic representation and parsing algorithms used in our pipeline are language-agnostic, they may require adaptation to new languages and domains. Furthermore, for domains and text genres that more closely resemble everyday language use, deep semantic analysis would require lexical inference, a notoriously difficult task in computational semantics [32, 33]. In our general rule representation we concentrated on deontic statements of a reasonably simple form. While this form seems to be well adapted to the regulations provided in the texts for the zoning maps of Vienna, there are some obvious limitations. First, since we always assume the presence of a deontic modality (obligation, prohibition or permission), at the moment we cannot treat *constitutive norms* [34] such as "The area marked on the map with the label *BB1* is designated a residential area". This issue could be addressed by adding an additional modality "consitutiveNorm" to the general representation together with an appropriate translation. Second, our propositional language is currently rather restricted, since we do not permit, e.g., disjunctions in the conditions or content of an obligation. Again, this could be addressed rather straightforwardly by extending the format of our representation, possibly along the lines of JsonLogic[17]. We also do not consider quantification or nested deontic operators. For the current application these features seemed not to be necessary. Most of these limitations are in line with other current approaches, e.g., [16, 28].

The proof-of-concept application presented in this paper can serve as a blueprint for semantics-based solutions to a wide range of information extraction tasks including variants of entity recognition and relation extraction. Such systems are generally more flexible, interpretable, and less prone to bias than the large neural network models used for similar tasks. However, to make such systems a viable alternative for everyday NLP applications, novel methods must be devised for the (semi-)automatic learning of task-specific rule systems like the one manually built for this project. Concerning the automated reasoning part, we plan to consider specifications to different frameworks in the future, including those of argumentation theory [27], I/O logic [28], and defeasible deontic logic [16], and integrate existing provers for these formalisms such as TOAST[18], SPINdle[19] or TurnipBox[20]. Additionally, we plan to implement alternative translations from the generic representation to the language of dyadic deontic logic, corresponding to different interpretations of the logical structure of deontic statements. Along the lines of [35] this could be used to compare such different interpretations. Finally, we would like to investigate whether the part of our pipeline creating general rule representations could be used in combination with the NAI suite [17]. Our rule-based approach could be used as a first step to automatically suggest a formalisation of a given legal text, which then could be converted into the format used in the NAI suite and run through the quality assurance function provided there. The benefit would be that the legal experts do not need to actively formulate the formalisation of a legal text, but only to potentially adjust it based on the quality assurance checks.

Acknowledgments

We are grateful to the three anonymous reviewers for their suggestions and for additional references. Work supported by BRISE-Vienna (UIA04-081), a European Union Urban Innovative Actions project.

[16] https://github.com/recski/brise-plandok

[17] https://jsonlogic.com

[18] http://toast.arg-tech.org/

[19] http://spindle.data61.csiro.au/spindle/

[20] https://turnipbox.netlify.app

References

[1] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, N. Schneider, Abstract Meaning Representation for sembanking, in: Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, Association for Computational Linguistics, Sofia, Bulgaria, 2013, pp. 178–186. URL: https://www.aclweb.org/anthology/W13-2322.

[2] C. Lyu, I. Titov, AMR parsing as graph prediction with latent alignment, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 397–407. URL: https://www.aclweb.org/anthology/P18-1037. doi:10.18653/v1/P18-1037.

[3] S. Zhang, X. Ma, K. Duh, B. Van Durme, AMR parsing as sequence-to-graph transduction, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 80–94. URL: https://www.aclweb.org/anthology/P19-1009. doi:10.18653/v1/P19-1009.

[4] O. Abend, A. Rappoport, Universal Conceptual Cognitive Annotation (UCCA), in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Sofia, Bulgaria, 2013, pp. 228–238. URL: https://www.aclweb.org/anthology/P13-1023.

[5] D. Hershcovich, O. Abend, A. Rappoport, A transition-based directed acyclic graph parser for UCCA, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 1127–1138. URL: https://www.aclweb.org/anthology/P17-1104. doi:10.18653/v1/P17-1104.

[6] D. Hershcovich, O. Abend, A. Rappoport, Multitask parsing across semantic representations, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 373–385. URL: https://www.aclweb.org/anthology/P18-1035. doi:10.18653/v1/P18-1035.

[7] H. Ozaki, G. Morio, Y. Koreeda, T. Morishita, T. Miyoshi, Hitachi at MRP 2020: Text-to-graph-notation transducer, in: Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing, Association for Computational Linguistics, Online, 2020, pp. 40–52. URL: https://www.aclweb.org/anthology/2020.conll-shared.4. doi:10.18653/v1/2020.conll-shared.4.

[8] D. Samuel, M. Straka, ÚFAL at MRP 2020: Permutation-invariant semantic parsing in PERIN, in: Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing, Association for Computational Linguistics, Online, 2020, pp. 53–64. URL: https://www.aclweb.org/anthology/2020.conll-shared.5. doi:10.18653/v1/2020.conll-shared.5.

[9] A. Kornai, The algebra of lexical semantics, in: C. Ebert, G. Jäger, J. Michaelis (Eds.), Proceedings of the 11th Mathematics of Language Workshop, LNAI 6149, Springer, 2010, pp. 174–199. doi:10.5555/1886644.1886658.

[10] G. Recski, Building concept definitions from explanatory dictionaries, International Journal of Lexicography 31 (2018) 274–311. doi:10.1093/ijl/ecx007.

[11] Á. Kovács, K. Gémes, A. Kornai, G. Recski, BMEAUT at SemEval-2020 task 2: Lexical entailment with semantic graphs, in: Proceedings of the Fourteenth Workshop on Semantic Evaluation, International Committee for Computational Linguistics, Barcelona (online), 2020, pp. 135–141. URL: https://www.aclweb.org/anthology/2020.semeval-1.15.

[12] J. Nivre, M. Abrams, Ž. Agić, L. Ahrenberg, L. Antonsen, K. Aplonova, M. J. Aranzabe, G. Arutie, M. Asahara, L. Ateyah, M. Attia, A. Atutxa, L. Augustinus, E. Badmaeva, M. Ballesteros, E. Banerjee, S. Bank, V. Barbu Mititelu, V. Basmov, J. Bauer, S. Bellato, K. Bengoetxea, Y. Berzak, I. A. Bhat, R. A. Bhat, E. Biagetti, E. Bick, R. Blokland, V. Bobicev, C. Börstell, C. Bosco, G. Bouma, S. Bowman, A. Boyd, A. Burchardt, M. Candito, B. Caron, G. Caron, G. Cebiroğlu Eryiğit, F. M. Cecchini, G. G. A. Celano, S. Čéplö, S. Cetin, F. Chalub, J. Choi, Y. Cho, J. Chun, S. Cinková, A. Collomb, Ç. Çöltekin, M. Connor, M. Courtin, E. Davidson, M.-C. de Marneffe, V. de Paiva, A. Diaz de Ilarraza, C. Dickerson, P. Dirix, K. Dobrovoljc, T. Dozat, K. Droganova, P. Dwivedi, M. Eli, A. Elkahky, B. Ephrem, T. Erjavec, A. Etienne, R. Farkas, H. Fernandez Alcalde, J. Foster, C. Freitas, K. Gajdošová, D. Galbraith, M. Garcia, M. Gärdenfors, S. Garza, K. Gerdes, F. Ginter, I. Goenaga, K. Gojenola, M. Gökırmak, Y. Goldberg, X. Gómez Guinovart, B. Gonzáles Saavedra, M. Grioni, N. Grūzītis, B. Guillaume, C. Guillot-Barbance, N. Habash, J. Hajič, J. Hajič jr., L. Hà Mỹ, N.-R. Han, K. Harris, D. Haug, B. Hladká, J. Hlaváčová, F. Hociung, P. Hohle, J. Hwang, R. Ion, E. Irimia, Ọ. Ishola,

T. Jelínek, A. Johannsen, F. Jørgensen, H. Kaşıkara, S. Kahane, H. Kanayama, J. Kanerva, B. Katz, T. Kayadelen, J. Kenney, V. Kettnerová, J. Kirchner, K. Kopacewicz, N. Kotsyba, S. Krek, S. Kwak, V. Laippala, L. Lambertino, L. Lam, T. Lando, S. D. Larasati, A. Lavrentiev, J. Lee, P. Lê Hồng, A. Lenci, S. Lertpradit, H. Leung, C. Y. Li, J. Li, K. Li, K. Lim, N. Ljubešić, O. Loginova, O. Lyashevskaya, T. Lynn, V. Macketanz, A. Makazhanov, M. Mandl, C. Manning, R. Manurung, C. Mărănduc, D. Mareček, K. Marheinecke, H. Martínez Alonso, A. Martins, J. Mašek, Y. Matsumoto, R. McDonald, G. Mendonça, N. Miekka, M. Misirpashayeva, A. Missilä, C. Mititelu, Y. Miyao, S. Montemagni, A. More, L. Moreno Romero, K. S. Mori, S. Mori, B. Mortensen, B. Moskalevskyi, K. Muischnek, Y. Murawaki, K. Müürisep, P. Nainwani, J. I. Navarro Horñiacek, A. Nedoluzhko, G. Nešpore-Bērzkalne, L. Nguyễn Thị, H. Nguyễn Thị Minh, V. Nikolaev, R. Nitisaroj, H. Nurmi, S. Ojala, M. Olúòkun, Adédayọ̀and Omura, P. Osenova, R. Östling, L. Øvrelid, N. Partanen, E. Pascual, M. Passarotti, A. Patejuk, G. Paulino-Passos, S. Peng, C.-A. Perez, G. Perrier, S. Petrov, J. Piitulainen, E. Pitler, B. Plank, T. Poibeau, M. Popel, L. Pretkalniņa, S. Prévost, P. Prokopidis, A. Przepiórkowski, T. Puolakainen, S. Pyysalo, A. Rääbis, A. Rademaker, L. Ramasamy, T. Rama, C. Ramisch, V. Ravishankar, L. Real, S. Reddy, G. Rehm, M. Rießler, L. Rinaldi, L. Rituma, L. Rocha, M. Romanenko, R. Rosa, D. Rovati, V. Roșca, O. Rudina, J. Rueter, S. Sadde, B. Sagot, S. Saleh, T. Samardžić, S. Samson, M. Sanguinetti, B. Saulīte, Y. Sawanakunanon, N. Schneider, S. Schuster, D. Seddah, W. Seeker, M. Seraji, M. Shen, A. Shimada, M. Shohibussirri, D. Sichinava, N. Silveira, M. Simi, R. Simionescu, K. Simkó, M. Šimková, K. Simov, A. Smith, I. Soares-Bastos, C. Spadine, A. Stella, M. Straka, J. Strnadová, A. Suhr, U. Sulubacak, Z. Szántó, D. Taji, Y. Takahashi, T. Tanaka, I. Tellier, T. Trosterud, A. Trukhina, R. Tsarfaty, F. Tyers, S. Uematsu, Z. Urešová, L. Uria, H. Uszkoreit, S. Vajjala, D. van Niekerk, G. van Noord, V. Varga, E. Villemonte de la Clergerie, V. Vincze, L. Wallin, J. X. Wang, J. N. Washington, S. Williams, M. Wirén, T. Woldemariam, T.-s. Wong, C. Yan, M. M. Yavrumyan, Z. Yu, Z. Žabokrtský, A. Zeldes, D. Zeman, M. Zhang, H. Zhu, Universal dependencies 2.3, 2018. URL: http://hdl.handle.net/11234/1-2895, LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

[13] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, C. D. Manning, Stanza: A python natural language processing toolkit for many human languages, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 101–108. URL: https://www.aclweb.org/anthology/2020.acl-demos.14. doi:10.18653/v1/2020.acl-demos.14.

[14] A. Koller, Semantic construction with graph grammars, in: Proceedings of the 11th International Conference on Computational Semantics, Association for Computational Linguistics, London, UK, 2015, pp. 228–238. URL: https://www.aclweb.org/anthology/W15-0127.

[15] M. Sergot, F. Sadri, R. Kowalski, F. Kriwaczek, P. Hammond, H. Cory, The British Nationality Act as a logic program, Communications of the ACM 29 (1986).

[16] G. Governatori, Practical normative reasoning with defeasible deontic logic, in: C. d'Amato, M. Theobald (Eds.), Reasoning Web 2018, volume 11078 of *LNCS*, Springer, 2018, pp. 1–25.

[17] T. Libal, A. Steen, NAI: Towards transparent and usable semi-automated legal analysis, in: IRIS 2020, Editions Weblaw, 2020, pp. 265–272.

[18] S. Batsakis, G. Baryannis, G. Governatori, I. Tachmazidis, G. Antoniou, Legal representation and reasoning in practice: A critical comparison, in: JURIX 2018, IOS Press, 2018, pp. 31–40. URL: https://doi.org/10.3233/978-1-61499-935-5-31.

[19] R. Calegari, G. Contissa, F. Lagioia, A. Omicini, G. Sartor, Defeasible systems in legal reasoning: A comparative assessment, in: JURIX 2019, IOS Press, 2019, pp. 169–174. doi:10.3233/FAIA190320.

[20] T. Libal, A meta-level annotation language for legal texts, in: M. Dastani, H. Dong, L. van der Torre (Eds.), CLAR 2020, volume 12061 of *LNCS*, Springer, 2020, pp. 131–150. doi:10.1007/978-3-030-44638-3_9.

[21] A. Ciabattoni, B. Lellmann, Sequent rules for reasoning and conflict resolution in conditional norms, in: F. Liu, A. Marra, P. Portner, F. V. D. Putte (Eds.), DEON 2020/2021, College Publications, 2021.

[22] D. Merigoux, N. Chataing, J. Protzenko, Catala: A programming language for the law, CoRR abs/2103.03198 (2021). URL: https://arxiv.org/abs/2103.03198. arXiv:2103.03198.

[23] N. O. Nawari, A generalized adaptive framework (GAF) for automating code compliance checking, Buildings 9 (2019). URL: https://www.mdpi.com/2075-5309/9/4/86. doi:10.3390/buildings9040086.

[24] A. Sleimi, N. Sannier, M. Sabetzadeh, L. C. Briand, M. Ceci, J. Dann, An automated framework for the extraction of semantic legal metadata from legal texts, Empirical Software Engineering 26 (2021) 43. doi:10.1007/s10664-020-09933-5.

[25] J. Morris, Blawx: Rules as code demonstration, MIT Computational Law Report (2020). URL: https://law.mit.edu/pub/blawxrulesascodedemonstration.

[26] J. Zhang, N. M. El-Gohary, Automated information transformation for automated regulatory compliance checking in construction, Journal of Computing in Civil Engineering 29 (2015) B4015001. doi:10.1061/(ASCE)CP.1943-5487.0000427.

[27] S. Modgil, H. Prakken, The ASPIC+ framework for structured argumentation: A tutorial, Argument and Computation 5 (2014) 31–62. URL: http://dx.doi.org/10.1080/19462166.2013.869766.

[28] X. Parent, L. van der Torre, Input/output logic, in: D. Gabbay, J. Horty, X. Parent, R. van der Meyden, L. van der Torre (Eds.), Handbook of Deontic Logic and Normative Systems, College Publications, 2013, pp. 495–544.

[29] A. Koller, M. Kuhlmann, A generalized view on parsing and translation, in: Proceedings of the 12th International Conference on Parsing Technologies, Association for Computational Linguistics, Dublin, Ireland, 2011, pp. 2–13. URL: https://www.aclweb.org/anthology/W11-2902.

[30] R. Reiter, A logic for default reasoning, Artificial Intelligence (1980).

[31] J. F. Horty, Reasons as Defaults, Oxford University Press, 2012.

[32] A. Talman, S. Chatzikyriakidis, Testing the generalization power of neural network models across NLI benchmarks, in: Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, Association for Computational Linguistics, Florence, Italy, 2019, pp. 85–94. URL: https://www.aclweb.org/anthology/W19-4810. doi:10.18653/v1/W19-4810.

[33] M. Schmitt, H. Schütze, Language models for lexical inference in context, in: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Association for Computational Linguistics, Online, 2021, pp. 1267–1280. URL: https://www.aclweb.org/anthology/2021.eacl-main.108.

[34] G. Boella, L. W. N. van der Torre, Regulative and constitutive norms in normative multiagent systems, in: D. Dubois, C. A. Welty, M. Williams (Eds.), KR2004, AAAI Press, 2004, pp. 255–266. URL: http://www.aaai.org/Library/KR/2004/kr04-028.php.

[35] B. Lellmann, F. Gulisano, A. Ciabattoni, Mīmāṃsā deontic reasoning using specificity: A proof theoretic approach, Artificial Intelligence and Law (published online 2020). doi:10.1007/s10506-020-09278-w.