Extracting Formal Models from Normative Texts

John J. Camilleri[1], Normunds Gruzitis[2], and Gerardo Schneider[1]

[1] Department of Computer Science and Engineering,
Chalmers University of Technology and University of Gothenburg, Sweden
[2] Institute of Mathematics and Computer Science, University of Latvia
{john.j.camilleri,gerardo}@cse.gu.se, normunds.gruzitis@lu.lv

Abstract. We are concerned with the analysis of *normative texts*—documents based on the deontic notions of obligation, permission, and prohibition. Our goal is to make queries about these notions and verify that a text satisfies certain properties concerning causality of actions and timing constraints. This requires taking the original text and building a representation (model) of it in a formal language, in our case the *C-O Diagram* formalism. We present an experimental, semi-automatic aid that helps to bridge the gap between a normative text in natural language and its *C-O Diagram* representation. Our approach consists of using dependency structures obtained from the state-of-the-art Stanford Parser, and applying our own rules and heuristics in order to extract the relevant components. The result is a tabular data structure where each sentence is split into suitable fields, which can then be converted into a *C-O Diagram*. The process is not fully automatic however, and some post-editing is generally required of the user. We apply our tool and perform experiments on documents from different domains, and report an initial evaluation of the accuracy and feasibility of our approach.

Keywords: information extraction, normative texts, dependency parsing, C-O diagrams

1 Introduction

Normative texts are natural language documents which are concerned with what must be done, may be done, or should not be done—also known as *deontic norms*. This class of documents often includes legal contracts, terms of services, regulations and service level agreements. Our work involves the analysis of such texts using formal methods. We achieve this by modelling normative documents within a formalism that then allows us to perform complex queries and verify properties about them. The formalism used for this task is *C-O Diagrams* [8, 15], which provides a language for visualising normative texts involving the modalities of obligation, permission and prohibition (forbiddance). These are indicated by the letters **O**, **P** and **F** respectively. It allows the expression of these norms over different agents and actions, together with *reparations* which apply when obligations and prohibitions are violated. Figure 1 shows an example *C-O Diagram*.
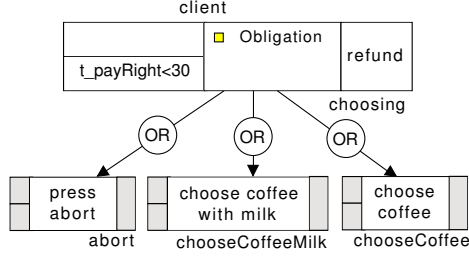
Fig. 1. Example of a *C-O Diagram*, showing an obligation of *client* to choose one of three actions within 30 time units of a previous action.

Models in this formalism can be converted in an automatic, deterministic way into networks of timed automata [1, 4], which are amenable to verification using the UPPAAL model checker [3]. There is, however, a large gap between the natural language texts as written by humans, and the formal representation used for automated analysis. Because of this, the task of modelling a text is completely manual, requiring a good knowledge of both the domain and the formalism. In this paper we present a method which helps to bridge this gap, by automatically extracting a partial model using NLP techniques.

Contributions We present here our technique for processing normative texts written in natural language and building partial models from them by analysing their syntactic structure and extracting relevant information.

Our method uses dependency structures obtained from a general-purpose statistical parser, namely the Stanford Parser [12, 18], which are then processed using custom rules and heuristics that we have specified based on a small development corpus in order to produce a table of clauses (draft predicate candidates). This can be seen as a specific information extraction task. While this method may only produce a partial model, requiring further post-editing by the user, we aim to save the most tedious work so that the user (knowledge engineer) can focus better on formalisation details.

We discuss the application of this method to a small test corpus of unseen sentences, and report on the performance based on a precision-recall metric.

2   Extracting Draft Predicate Candidates

The proposed approach is application-specific but domain-independent, assuming that normative texts (or *contracts*), tend to follow a certain restricted style of natural language, even though there are variations across and within domains. However, we do not impose any grammatical or lexical restrictions on the input texts, therefore we first apply a general-purpose parser acquiring a syntactic dependency tree representation for each sentence. Provided that the syntactic analysis does not contain significant errors, we then apply a number of interpretation rules and heuristics on top of dependency structures. If the extraction is

successful, one or more predicate candidates are acquired for each input sentence as shown in Table 1. More than one candidate is extracted in case of explicit or implicit coordination of subjects, verbs, objects or main clauses.

Table 1. Sample input and partial output.

Refin.	Mod.	Subject (S)	Verb (V)	Object (O)	Modifiers
1. *You must not, in the use of the Service, violate any laws in your jurisdiction (including but not limited to copyright or trademark laws).*					
	F	User	violate	law	V: in User's jurisdiction V: in the use of the Service
2. *You will not post unauthorized commercial communication (such as spam) on Facebook.*					
	F	User	post	unauthorized commercial communication	O: such as spam O: on Facebook
3. *You will not upload viruses or other malicious code.*					
	F	User	upload	virus	
OR	F	User	upload	other malicious code	
4. *Your login may only be used by one person - a single login shared by multiple people is not permitted.*					
	P	person	use	login of User	S: one
5. *The renter shall pay all reasonable attorney and other fees, the expenses and costs incurred by owner in protection its rights under this rental agreement and for any action taken owner to collect any amounts due the owner under this rental agreement.*					
	O	renter	pay	reasonable attorney	V: under this rental agreement
AND	O	renter	pay	other fee	V: under this rental agreement
6. *The equipment shall be delivered to renter and returned to owner at the renter's risk, cost and expense.*					
	O	equipment	[is] delivered [to]	renter	V: at renter's risk, cost and expense
AND	O	equipment	[is] returned [to]	owner	V: at renter's risk, cost and expense

The dependency representation instead of phrase-structure representation allows for a more straightforward predicate extraction based on the syntactic relations instead of phrase types of the parts of a sentence.

In our experiment, we use the Stanford Parser whose accuracy on Penn Treebank (the WSJ section) is around 90% [18]. The Stanford dependency representation [14] is being increasingly adapted to parsers for other languages as well, for instance, Chinese [6], Finnish [11] and Persian [17], and it is the basis for

the Universal Dependencies project [13]. However, our approach as such is not restricted to the specific parser or dependency representation.

2.1 Expected Input and Intended Output

The basic requirement for pre-processing the input text is that it is split by sentence and that only the relevant sentences are included. In this experiment, we have manually selected the relevant sentences, ignoring (sub)titles, introductory notes etc. Automatic analysis of the document structure is a separate issue. We also expect that sentences do not contain grammatical errors that would considerably affect the syntactic analysis and thus the output of our tool. The output is a table (in tab-separated format) where each row corresponds to a *C-O Diagram* box (clause), containing fields for:

Subject the agent of the clause;
Verb the verbal component of an action;
Object the object component of an action (optional);
Modality obligation (O), permission (P), prohibition (F), or declaration (D) for clauses which only state facts;
Refinement whether a clause should be attached to the preceding clause by conjunction (AND), choice (OR) or sequence (SEQ);
Time adverbial modifiers clearly indicating temporality;
Adverbials other adverbial phrases that modify the action;
Conditions phrases indicating conditions on agents, actions or objects;
Notes other phrases that provide additional information (e.g. relative clauses), indicating the element (head word) they attach to.

Values of the Subject, Verb and Object fields undergo certain normalisation and formatting: head words are lemmatised; Saxon genitives are converted to of-constructions if contextually possible; the preposition "to" is explicitly added to indirect objects; prepositions of prepositional objects are included in the Verb field as part of the predicate name, as well as the copula if the predicate is expressed by a participle, adjective or noun; definite and indefinite articles are omitted.

A complete document in this format can be converted automatically into a *C-O Diagram* model. Our tool however does not necessarily produce a *complete* table, in that fields may be left blank when we cannot determine what to use. There is also the question of what is considered *correct* output. It may be the case that certain clauses may be encoded in multiple ways, and, while all fields may be filled, the user may find it more desirable to change the encoding.

2.2 Rules

We make a distinction between rules and heuristics that are applied on top of Stanford Dependencies. Rules are everything that explicitly follow from the dependency relations and part-of-speech tags. For example, the head of the subject
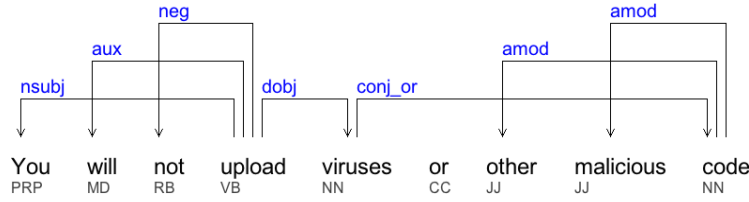
Fig. 2. Sample dependency tree.

noun phrase (NP) is labelled by `nsubj`, and the head of the direct object NP—by `dobj` (see Figure 2); fields Subject and Object of the output table can be straightforwardly populated by the respective phrases (as in Table 1).

We also count as lexicalised rules cases when the decision can be obviously made by considering both the dependency label and the head word. For example, modal verbs and other auxiliaries of the main verb are labelled as `aux` but words like "may" and "must" clearly indicate the respective modality (P and O). Auxiliaries can be combined with other modifiers, for example, the modifier "not" (`neg`) which indicates prohibition. In such cases, the rule is that obligation overrides permission, and prohibition overrides both obligation and permission.

In order to provide concise values (terms) for the Subject and Object fields, relative clauses (`rcmod`), verbal modifiers (`vmod`) and prepositional modifiers (`prep`) that modify heads of the subject and object NPs are separated in the Notes field. Adverbial modifiers (`advmod`), prepositional modifiers and adverbial clauses (`advcl`) that modify the main verb are separated, by default, in the Adverbials field.

If the main clause is expressed in the passive voice, and the agent is mentioned (expressed by the preposition "by"), the resulting predicate is converted to the active voice (as shown by the fourth example in Table 1).

2.3 Heuristics

In addition to the obvious extraction rules, we apply a number of heuristic rules based on the development examples and our intuition about the application domains and the language of normative texts.

First of all, auxiliaries are compared and classified against extended lists of keywords. For example, the modal verb "can" most likely indicates permission while "shall" and "will" indicate obligation. In addition to auxiliaries, we also consider the predicate itself (expressed by a verb, adjective or noun). For example, words like "responsible", "liable" and "require" most likely express obligation.

For prepositional phrases (PP) that are direct dependants of Verb, we first check if they reliably indicate a temporal modifier and thus should be put in the Time field. The list of such prepositions include "after", "before", "until", "during" etc. If the preposition is ambiguous, the head of the noun phrase is

checked if it bears a meaning of time. There is a relatively open list of such potential keywords, including "day", "week", "month" etc. Due to PP-attachment errors that syntactic parses often make, if a PP is attached to Object, and it has the above mentioned indicators of a temporal meaning, the phrase is put in the Verb-dependent Time field.

Similarly, we check the markers (`mark`) of adverbial clauses if they indicate time ("while", "when" etc.) or a condition (e.g. "if"), as well as values of simple adverbial modifiers, looking for "always", "immediately", "before" etc. Adverbial modifiers are also checked against a list of irrelevant adverbs used for emphasis (e.g. "very") or as gluing words (e.g. "however", "also").

NPs of the Subject and Object fields are checked for attributes: if an NP is modified by a number (`num` or `number`), these modifiers are treated as conditions and are separated in the respective field.

If there is no direct object in the sentence, or, in the case of the passive voice, no agent expressed by a prepositional phrase (using the preposition "by"), the first PP governed by Verb is treated as a prepositional object and thus is included in the Object field. Indirect objects result in additional refinement predicates.

Additionally, anaphoric references by personal pronouns are detected, normalised and tagged (e.g. "we", "our" and "us" are all rewritten as "<we>"). In the case of terms of services, for instance, pronouns "we" and "you" are often used to refer to the service and the user respectively. The tool can be customised to do such a simple but effective anaphora resolution (see Table 1).

2.4 Post-editing

Since we do not intend for our tool to be a complete replacement for a human knowledge engineer, a certain amount of post-editing is often required. This post-editing can be categorised into the following different types, listed here in approximate order of effort required:

1. Filling in empty fields.
2. Adding or removing adverbial information from the subject and object.
3. Changing the verb or modality.
4. Refinement into sub-clauses.
5. Complete paraphrasing.

3 Experiments

In order to test the potential and feasibility of the proposed approach, we have selected four normative texts from three different domains: two terms of service agreements, a rental agreement and a PhD regulations document. In the development stage, we considered first 10 sentences of each document, based on which the rules and heuristics were defined. For the evaluation, we used the next 10 sentences of each document. The four documents used in our experiments are:

1. PhD regulations from Chalmers University

2. Equipment rental agreement from RSO, Inc.[3]
3. Terms of service for GitHub, Inc.[4]
4. Terms of service for Facebook[5]

After preparing the test sets and applying our tool, the tabular output from each set was evaluated manually according to the following criteria.

3.1 Evaluation Criteria

In our initial evaluation, we use a simple precision-recall metric over the following fields: Subject, Verb, Object and Modality. The other fields (Time, Adverbials, Conditions and Notes) of our table structure are not included in the evaluation criteria as they are intrinsically too unstructured and will always require some post-editing in order to be formalised.

Precision is concerned with rating the accuracy of the output of the tool. For each value in the respective fields, we assign a point when it matches with our assessment of what the correct value should be. When a single sentence results in a refinement with multiple clauses, we score each of these individually.

Recall is a measure of how much of the intended information the tool was able to extract. For each sentence in the original text, we check whether the correct fields have been extracted by the tool, scoring accordingly. When a sentence should result in multiple clauses, we score for each of these individually.

The local scores for precision and recall are often identical, because a sentence in the original text would correspond to one row (clause) in the table. This is not the case when unnecessary refinements are added by the tool or, conversely, when coordinations in the text are not correctly added as refinements.

The evaluation was performed twice: first when using only the rules (Section 2.2), and then again when using the rules and heuristics together (Section 2.3). A summary of our experimental results can be found in Table 2, including the harmonic mean scores (F_1) between precision and recall.

Table 2. Evaluation results based on a small set of test sentences (10 per document).

Document	Rules only			Rules & heuristics		
	Precision	*Recall*	*F_1*	*Precision*	*Recall*	*F_1*
PhD	0.66	0.73	0.69	0.82	0.90	0.86
Rental	0.75	0.67	0.71	0.71	0.66	0.69
GitHub	0.46	0.53	0.49	0.48	0.55	0.51
Facebook	0.43	0.54	0.48	0.43	0.57	0.49

---

[3] http://www.rsoinc.com/pdfs/equip_rental_revb.pdf

[4] https://help.github.com/articles/github-terms-of-service/

[5] https://www.facebook.com/legal/terms

3.2   Observations

The first observation from the results is that the F_1 score varies quite a lot between documents; from 0.49 to 0.86. This is mainly due to the variations in language style present in the documents. Overall the application of heuristics together with the rules does improve the scores obtained.

On the one hand, many of the sentence patterns which we handle in the heuristics appear only in the development set and not in the test set. On the other hand, there are few cases which occur relatively frequently among the test examples but are not covered by the development set. For instance, the introductory part of a sentence, the syntactic main clause, is sometimes pointless for our formalism, and it should be ignored, taking instead the sub-clause as the semantic main clause, e.g.:

> *User understands that ...*

The small corpus size is of course an issue here, therefore we cannot make any strong statements about the representative coverage of the development and test sets.

Analysing the modal verb *shall* seems to be particularly difficult to get right. It may either be an indication of an obligation when concerning an action, or it may be used as a prescriptive construct as in *shall be* which is more indicative of a declaration.

3.3   Paraphrasing

In some sense, the task of extracting the correct fields from each sentence can be seen as paraphrasing from the given sentence into one of the known patterns which can be handled by our rules. We give here some examples of errors encountered in the experiments, which can only currently be fixed by making non-trivial paraphrasing.

> *GitHub reserves the right at any time to modify or discontinue, temporarily or permanently, your access to the API with or without notice.*

For this sentence, our tool picks up *reserve* as the verb and *right* as the object, but this should really be realised as a permission with *modify* as the verb and *access to API* as the object. This could furthermore be refined as a permission of a choice of actions (modify, discontinue temporarily, discontinue permanently). Additional phrases such as *at any time* and *with or without notice* are actually not informative here, as they reflect the default behaviour of the formalism (i.e. lack of constraints).

> *We require applications to respect your privacy, and your agreement with that application will control how the application can use, store, and transfer that content and information.*

Here we get an obligation with the subject *we*, the verb *require* and the object *applications to respect your privacy*. The correct encoding however would be to make *applications* the subject, with *respect* as the verb and the object being *your privacy*.

> *When you publish content or information using the Public setting, it means that you are allowing everyone, including people off of Facebook, to access and use that information, and to associate it with you.*

In this case, the tool fails completely, returning *it* as the subject, *mean* as the verb, and *allowing everyone* as the object, with the declarative modality (D). The entire *when* clause should be treated as a condition. The phrase *you are allowing everyone* should more correctly be paraphrased as *everyone is allowed*, making this actually a permission with the subject *everyone*, the verb *access* and the object *that information*.

> *To learn more about Platform, including how you can control what information other people may share with applications, read our Data Policy and Platform Page.*

Sentences like this one are generally unimportant for our goals and should be ignored altogether, however we currently have no way identifying unhelpful sentences and ignoring them.

4   Formal Analysis

The ultimate goal of formalising normative texts is to be able to perform automated analysis, by which we mean running queries of various kinds against the *C-O Diagram* model.

Syntactic queries are those which can be checked at a syntactic level, such as checking if a text contains any permissions for a particular agent or identifying obligations without constraints or reparations. These are based on *predicates* defined over single clauses, which serve as the building block for defining queries over the entire model. The predicate $isObl(C)$ for example is true if the clause C is an obligation. Syntactic queries are expressed by combining such predicates.

Queries which deal with timing constraints, possibility and invariance are called **semantic queries**. For example, checking whether a clause could be enacted within a certain amount of time may depend on a previous sequence of events. Determining whether these events may or may not happen cannot be done from the syntax alone. Such queries are computed by converting the *C-O Diagram* model into a network of timed automata and then applying the UPPAAL model checker. Semantic queries are expressed using UPPAAL's requirement specification language which is a subset of timed computation tree logic (TCTL).

5   Related Work

Information extraction is a large topic where examples generally tend to consist of applying standard NLP techniques combined with customised rules to some specific domain and problem.

Our work is very similar to that of Wyner and Peters [19] tackles a very similar problem to our own, identifying and extracting rules from the language of the source material so that they can be expressed using conditional and deontic rules. Their rule-based approach Stanford Parser Using GATE

US Federal Code of Regulations

The paper presents a linguistically-oriented, rule-based approach It outlines use cases, discusses the source materials, reviews the methodology, then provides initial results and future steps.

Mercatali et al. [16] tackle the automatic translation of textual representations of laws to a formal model, in their case UML. This underlying formalism is of course different, where they are mainly interested in the hierarchical structure of the documents rather than the norms themselves. Their method does not use dependency or phrase-structure trees but shallow syntactic chunks.

Cheng et al. [7] also describe a system for extracting structured information for texts in a specific legal domain. Their method combines surface-level methods like tagging and named entity recognition (NER) with semantic analysis rules which were hand-crafted for their domain and output data format.

An alternative approach to bridging the gap between natural and formal languages is to introduce a *controlled natural language* (CNL)—a reduced subset of a natural language which is in fact formal. This may be done using either a general-purpose CNL such as Attempto Controlled English (ACE) [9] which comes with a parser to discourse representation structures, or a custom CNL.

In their work, Camilleri et al. [5] have designed a custom CNL which can be parsed directly into a *C-O Diagram*. The issue here is that the original natural language text may be quite a distance away from its CNL representation, and the translation from NL to CNL is still a manual step.

The FrameNet-CNL framework [2, 10] proposes an approach to information extraction problem by combining CNL with FrameNet—a lexicographic database describing word meanings based on the principles of frame semantics. Such a system would encompass a powerful abstract knowledge representation paradigm along with a real-world information extraction system, based on frame-semantic parsing. We consider the incorporation of semantic frames into our method as a direction for future work.

6   Conclusions and Future Work

As already mentioned in the introduction and Section 4, the main motivation of our work is to perform complex formal analyses of normative texts through syntactic and (timed) semantic queries through model checking. We are in this paper helping to bridge the gap between natural language texts and formal

analysis tools, taking the burden from the user to have to deal directly with formal languages. The approach presented here is thus not positioned as an eventual replacement for a human; rather it is to be seen as an aid for a semi-automatic transition from natural language to *C-O Diagrams*.

Though the results of the experiments reported here are indicative at best, because of the small test corpus, the application of our technique to the case studies reported here has significantly increased the efficiency in the task of "encoding" the documents into *C-O Diagrams*. Moreover, they are promising enough to warrant further work in this direction.

While the evaluation performed here measures the accuracy of the tool in terms of precision and recall, another relevant metric would be to measure the time spent building a model from scratch versus the time spent on post-editing the output from our tool.

It is common that some paraphrasing is needed during the post-editing phase. This may be on the syntactic level, e.g. by fixing adverbial attachment. However it may also require more in depth understanding, and often involves using related or opposite concepts which cannot be determined without more elaborated processing on the semantic level.

The *C-O Diagram* formalism is essentially *action-based*, where clauses prescribe what an agent should or should not *do*. However in the texts from our experiments we have found that it is very common to describe what should or should not *be*, i.e. referring to states of affairs. Handling these kinds of sentences will require more effective paraphrasing patterns for such cases.

References

[1] Alur, R., Dill, D.L.: A Theory of Timed Automata. Theoretical Computer Science 126(2), 183–235 (1994)

[2] Barzdins, G.: FrameNet CNL: A Knowledge Representation and Information Extraction Language. In: Controlled Natural Language, LNCS, vol. 8625, pp. 90–101. Springer (2014), `http://dx.doi.org/10.1007/978-3-319-10223-8_9`

[3] Behrmann, G., David, A., Larsen, K.G.: A Tutorial on UPPAAL 4.0. Tech. rep., Department of Computer Science, Aalborg University, Aalborg, Denmark (2006)

[4] Bengtsson, J., Yi, W.: Timed Automata: Semantics, Algorithms and Tools. In: Lectures on Concurrency and Petri Nets, LNCS, vol. 3098, pp. 87–124. Springer (2004), `http://www.springerlink.com/index/10.1007/b98282`

[5] Camilleri, J.J., Paganelli, G., Schneider, G.: A CNL for Contract-Oriented Diagrams. In: Controlled Natural Language, LNCS, vol. 8625, pp. 135–146. Springer (2014)

[6] Chang, P.C., Tseng, H., Jurafsky, D., Manning, C.D.: Discriminative re-ordering with chinese grammatical relations features. In: Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation. pp. 51–59. SSST '09, Association for Computational Linguistics, Stroudsburg, PA, USA (2009), http://dl.acm.org/citation.cfm?id=1626344.1626351

[7] Cheng, T.T., Cua, J., Tan, M., Yao, K., Roxas, R.: Information extraction from legal documents. In: Natural Language Processing, 2009. SNLP '09. Eighth International Symposium on. pp. 157–162 (Oct 2009)

[8] Díaz, G., Cambronero, M.E., Martínez, E., Schneider, G.: Specification and Verification of Normative Texts using C-O Diagrams. IEEE Transactions on Software Engineering 40(8), 795–817 (2014)

[9] Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for Knowledge Representation. In: Reasoning Web. LNCS, vol. 5224, pp. 104–124. Springer (2008)

[10] Gruzitis, N., Dannells, D.: A multilingual FrameNet-based grammar and lexicon for Controlled Natural Language. Language Resources and Evaluation (2015), http://arxiv.org/abs/1511.03924

[11] Haverinen, K., Nyblom, J., Viljanen, T., Laippala, V., Kohonen, S., Missil, A., Ojala, S., Salakoski, T., Ginter, F.: Building the essential resources for finnish: the turku dependency treebank. Language Resources and Evaluation 48(3), 493–531 (2014), http://dx.doi.org/10.1007/s10579-013-9244-1

[12] Klein, D., Manning, C.D.: Accurate Unlexicalized Parsing. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL). pp. 423–430 (2003)

[13] de Marneffe, M.C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., D. Manning, C.: Universal Stanford Dependencies: A cross-linguistic typology. In: Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC). pp. 4585–4592 (2014)

[14] de Marneffe, M.C., Manning, C.D.: The Stanford Typed Dependencies Representation. In: Proceedings of the COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation. pp. 1–8 (2008)

[15] Martinez, E., Cambronero, E., Diaz, G., Schneider, G.: A Model for Visual Specification of e-Contracts. In: Proceedings of the 7th International Conference on Services Computing (IEEE SCC). pp. 1–8. IEEE Computer Society (2010)

[16] Mercatali, P., Romano, F., Boschi, L., Spinicci, E.: Automatic Translation from Textual Representations of Laws to Formal Models through UML. In: Proceedings of the 18th Annual Conference on Legal Knowledge and Information Systems (JURIX). pp. 71–80. IOS Press (2005)

[17] Seraji, M., Jahani, C., Megyesi, B., Nivre, J.: A persian treebank with stanford typed dependencies. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14). European Language Resources Association (ELRA), Reykjavik, Iceland (may 2014)

[18] Socher, R., Bauer, J., Manning, C.D., Ng, A.Y.: Parsing With Compositional Vector Grammars. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL). pp. 455–465 (2013)

[19] Wyner, A., Peters, W.: On rule extraction from regulations. In: Atkinson, K. (ed.) The twenty-fourth annual conference on Legal knowledge and information systems (JURIX 2011), pp. 113–122. IOS Press, Amsterdam (2011)