9-13-2007

Logic-based Regulatory Conformance Checking

Nikhil Dinesh
University of Pennsylvania, nikhild@seas.upenn.edu

Aravind K. Joshi
University of pennsylvania, joshi@seas.upenn.edu

Insup Lee
University of Pennsylvania, lee@cis.upenn.edu

Oleg Sokolsky
University of Pennsylvania, sokolsky@cis.upenn.edu

14th Monterey Workshop (2007), Monterey, CA, September 10-13, 2007.

Logic-based Regulatory Conformance Checking

Abstract
In this paper, we describe an approach to formally assess whether an organization conforms to a body of regulation. Conformance is cast as a model checking question where the regulation is represented in a logic that is evaluated against an abstract model representing the operations of an organization. Regulatory bases are large and complex, and the long term goal of our work is to be able to use natural language processing (NLP) to assist in the translation of regulation to logic. We argue that the translation of regulation to logic should proceed one sentence at a time. A challenge in taking this approach arises from the fact that sentences is regulation often refer to others. We motivate the need for a formal representation of regulation to accommodate references between statements. We briefly describe a logic in which statements can refer to and reason about others. We then discuss preliminary work on using NLP to assist in the translation of regulatory sentences into logic.

Comments
14th Monterey Workshop (2007), Monterey, CA, September 10-13, 2007.

Logic-based Regulatory Conformance Checking

Nikhil Dinesh, Aravind Joshi, Insup Lee, and Oleg Sokolsky

Department of Computer Science
University of Pennsylvania
Philadelphia, PA - 19104, USA
{nikhild,joshi,lee,sokolsky}@seas.upenn.edu

Abstract. In this paper, we describe an approach to formally assess whether an organization conforms to a body of regulation. Conformance is cast as a model checking question where the regulation is represented in a logic that is evaluated against an abstract model representing the operations of an organization. Regulatory bases are large and complex, and the long term goal of our work is to be able to use natural language processing (NLP) to assist in the translation of regulation to logic.
We argue that the translation of regulation to logic should proceed one sentence at a time. A challenge in taking this approach arises from the fact that sentences is regulation often refer to others. We motivate the need for a formal representation of regulation to accomodate references between statements. We briefy describe a logic in which statements can refer to and reason about others. We then discuss preliminary work on using NLP to assist in the translation of regulatory sentences into logic.

1 Introduction

Regulations, laws and policies that affect many aspects of our lives are represented predominantly as documents in natural language. For example, the Food and Drug Administration's Code of Federal Regulations[1] (FDA CFR) governs the operations of American bloodbanks. The CFR is framed by experts in the field of medicine, and regulates the tests that need to be performed on donations of blood before they are used. In such safety-critical scenarios, it is desirable to assess formally whether an organization (bloodbank) conforms to the regulation (CFR).

Conformance checking is a relatively new problem in requirements engineering, which has been gaining attention in industry and academia [1]. A key difference between regulations and other sources of informal requirements is in determining the source of a requirement. The requirements used to design a system often arise from varied places, such as interviews with customers and discussions with domain experts. This makes the identification of requirements a difficult problem. However, since there are consequences associated with disobeying the law, law-makers spend considerable effort in articulating the requirements (as

[1] http://www.gpoaccess.gov/cfr/index.html

normative sentences). As a result, one can informally associate a requirement with a sentence or discourse.

The challenge in conformance checking is that the task of formalizing the requirements is difficult, due to the large size and complexity of regulations. The long term goal of our work is to use natural language processing (NLP) techniques to aid in the formalization of regulation. From the perspective of using NLP for requirements engineering, this area is epecially interesting due to the availability of large corpora of regulations that can serve as a test-bed for NLP techniques.

We approach the problem of formally determining conformance to regulation as a model-checking question. The regulation is translated to statements in a logic which are evaluated against a model representing the operations of an organization. The result of evaluation is either an affirmative answer to conformance, or a counterexample representing a subset of the operations of the organization and the specific law that is violated. A similar approach is adopted by several systems [1–3].

When a violation is detected, the problem could be in one of three places: (a) the organization's operations, (b) the regulation or (c) the translation of the regulation to the logic. To aid in determining the source of the problem, there needs to be a notion of correspondence between the sentences of regulation in natural language and logic. We attempt to maintain a correspondence by translating regulation to logic one sentence at a time. An added benefit of doing this is to be able to focus our NLP efforts at the sentence level.

In this paper, we discuss two related parts of our approach. The first part deals with the issue of designing a logic into which we can translate regulation one sentence at a time. The main difficulty that we encountered in doing this is the problem of *references to other laws*. A common phenomenon in regulatory texts is for sentences to function as conditions or exceptions to others. This function of sentences makes them dependent on others for their interpretation, and makes the translation to logic difficult. In Section 2, we argue (using examples and lexical occurence statistics) that a logic to represent regulation should provide mechanisms for statements to refer to others, and to make inferences from the sentences referred to.[2] In Section 3, we briefly describe the logic that we use to represent regulation.

In the second part of the paper (Section 4), we turn our attention to the problem of using NLP to assist in the translation of sentences of regulation into logic. Section 5 concludes.

2 The Problem of References to Other Laws

In this section, we argue that a logic to represent regulation should provide a mechanism for sentences to refer to others. The discussion is divided into two parts. In Section 2.1, we discuss examples of the phenomenon that we are

[2] A study in [1] suggests that such references between sentences are common in privacy regulation as well.

interested in and how they may be represented in a logic with no mechanism for sentences to refer to others. We then contrast the distribution of some lexical categories in the CFR with newspaper text, which suggest that references to sentences are an important way of expressing relationships between sentences in regulation (Section 2.2).

2.1 Examples

The examples in this section are shortened versions of sentences from the CFR Section 610.40, which we will use through the course of the paper. Consider the following sentences:

(1) Except as specified in (2), every donation of blood or blood component must be tested for evidence of infection due to Hepatitis B.

(2) You are not required to test donations of source plasma for evidence of infection due to Hepatitis B.

(1) conveys an obligation to test donations of blood or blood component for Hepatitis B, and (2) conveys a permission not to test a donation of source plasma (a blood component) for Hepatitis B. To assess an organization's conformance to (1) and (2), it suffices to check whether "all non-source plasma donations are tested for Hepatitis B". In other words, (1) and (2) imply the following obligation:

(3) Every non-source plasma donation must be tested for evidence of infection due to Hepatitis B.

There are a variety of logics in which one can capture the interpretation of (3), as needed for conformance. For example, in first-order logic, one can write $\forall x : (d(x) \wedge \neg sp(x)) \Rightarrow test(x)$, where $d(x)$ is true iff x is a donation, $sp(x)$ is true iff x is a source plasma donation, and $test(x)$ is true iff x is tested for Hepatitis B. Thus, to represent (1) and (2) formally, we inferred that they implied (3) and (3) could be represented more directly in a logic.

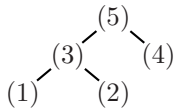
Now suppose we have a sentence that refers to (1):

(4) To test for Hepatitis B, you must use a screening test that the FDA has approved for such use.

The reference is more indirect here, but the interpretation is: "if (1) requires a test, then the test must be performed using an appropriate screening test kit". A bloodbank is not prevented from using a different kind of test for source plasma donations. (4) can be represented by first producing (3), and then inferring that (3) and (4) imply the following:

(5) Every non-source plasma donation must be tested for evidence of infection due to Hepatitis B using a screening test that the FDA has approved for such use.

It is easy to represent the interpretation of (5) directly in a logic. However, (5) has a complex relationship to the sentences from which it was derived, i.e., (1), (2) and (4). The derivation takes the form of a tree:



The examples we have considered are simplified versions of the sentences in the CFR 610.40. In the CFR, (1) has a total of six exceptions, and the exceptions have statements that qualify them further. This process of producing a derived obligation and translating it becomes extremely difficult.

References to other laws are not always hierarchical or acyclic. There are two kinds of circularities that can arise. The first is a syntactic circularity which arises due to vague references. For example, two occurences of the phrase "required testing under this section" can give rise to a cycle if one interprets "this section" as "all the other sentences in this section". However, such phrases typically appear in paragraphs where no tests are required and the cycle can be broken by restricting the references to paragraphs where tests are required. The second kind of circularity is a semantic circularity which can make the regulation paradoxical, e.g., with self referential sentences. Fortunately, we have not observed such circularities.

To summarize, if one wishes to use a logic with no support for referring to other sentences, translating regulation to the logic would involve the following steps: (a) resolving circularites to construct a hierarchy of references, (b) creating derived obligations by moving up the hierarchy, until a set of derived obligations with no references are obtained, and (c) translating the final set of derived obligations to logic.

This procedure would not be problematic if there are few cases of references. In the following section, we discuss the distribution of some lexical categories in the CFR which suggest that this is a very common case. This makes the procedure impractical in terms of the effort that would be involved. The logic that we describe in later sections lets one express references directly, and the resolution of cirularites and creation of derived obligations happen as part of the semantics.

2.2 Distribution of Lexical Categories

In the previous section, we saw several examples of how sentences in regulation refer to others. Natural language offers a variety of devices to relate sentences to others. A large class of such devices fall under the rubric of *anaphora*, which is a means of linking a sentence to the prior discourse. Common examples of such

anaphoric items are pronouns and adverbial connectives, e.g., however, instead, furthermore, etc.[3]

Table 1. Differences in the distribution of some anaphoric lexical items in the Wall Street Journal (WSJ) corpus and the CFR. Both the WSJ and the CFR have approximately 1M words.

Lexical Item	WSJ	CFR
he, she, him, her	8564	297
it, its	15168	2502
they, their	4500	862
ADV1	3162	2402
ADV2	2453	349
such	662	3028
References to other laws	_	18509

Table 1 contrasts the distribution of potentially anaphoric items in the Wall Street Journal (WSJ) corpus, with the CFR. The first three rows show counts of pronouns, and the CFR has a markedly lower number of pronouns than the WSJ. The next two rows show counts of adverbial connectives. ADV1 comprises of the connectives *also, however, in addition, otherwise, for example, therefore, previously, later, earlier, until* and *still*. These connectives have specialized uses in the CFR and tend to be quite frequent, with *otherwise* being the most frequent in the CFR (517 cases). ADV2 is a set of 48 adverbial connectives annotated by the Penn Discourse Treebank [4] excluding those in ADV1, e.g., *instead, as a result, nevertheless*. The connectives in ADV2 are significantly more frequent in the WSJ than in the CFR.

The last two rows in Table 1 show two common ways of establishing relationships between sentences in the CFR. The adjective *such* is a common way of refering to a set discussed in an immediately preceding law, e.g., *such tests*. The last row counts explicit references to other law, by searching for phrases like *this section*, or references to section and paragraph identifiers. Of the categories we considered this is by far the most frequent in the CFR.

We now describe the logic that we use to handle references. The other frequently occuring anaphora (ADV1 and *such*) are typically accompanied by references (e.g., *however* and *otherwise* give exceptions to other laws), and similar mechanisms can be used to express them. Formalizing the remaining anaphora is a subject of future work.

[3] Not all uses of pronouns are anaphoric. Some pronouns are bound by quantifiers, e.g., *every one loves their mother*. We report counts based on occurence of strings and do not distinguinsh between different uses.

3 A Logic that Allows References Between Laws

In this section, we describe the logic that we attempt to translate the regulation into. The description in this section is brief and informal, and introduces only the machinery needed to clarify the discussion in Section 4. We refer the reader to [5, 6] for a formal account of the semantics and the computational issues. Consider our examples again:

(6) Except as specified in (7), every donation of blood or blood component must be tested for evidence of infection due to Hepatitis B.

(7) You are not required to test donations of source plasma for evidence of infection due to Hepatitis B.

(6) and (7) are represented as follows:

- 6.**o**: $d(x) \wedge \neg \text{by}_7(\neg\Diamond test(x)) \rightsquigarrow \Diamond test(x)$ and
- 7.**p**: $d(y) \wedge sp(y) \rightsquigarrow \neg\Diamond test(y)$

First, consider the formula 7.**p**: $d(y) \wedge sp(y) \rightsquigarrow \neg\Diamond test(y)$. This is read as "It is permitted that if y is a donation of source plasma, then it is not tested eventually". The letter **p** denotes permission, $d(y)$ asserts that y is a donation, $sp(y)$ asserts that y consists of source plasma, $test(y)$ asserts that y is tested, and $\Diamond$is the linear temporal logic (LTL) operator eventually. The connective $\rightsquigarrow$ is a variant of implication which we will discuss in what follows.

Now consider the subformula $\text{by}_7(\neg\Diamond test(x))$. This is read as "By the law (7), x is not tested eventually". We note that this subformula should hold iff y is a donation of source plasma. And finally, 6.**o**: $d(x) \wedge \neg \text{by}_7(\neg\Diamond test(x)) \rightsquigarrow \Diamond test(x)$ can be paraphrased as "It is obligated that if x is a donation and it is not the case (7) doesn't permit that x is not tested eventually, then x must be tested eventually". The letter **o** denotes obligation. Formulas in the logic are evaluated with respect to sequences of states of an implementation (in a manner similar to LTL). Each state is associated with a set of objects and a way of evaluating predicates.

Table 2. A run and its annotations

Time	Objects	Predicates	Annotations
1	o_1	$d(o_1)$, $sp(o_1)$, $\neg test(o_1)$	2: $\neg\Diamond test(o_1)$
2	o_1	$d(o_1)$, $sp(o_1)$, $\neg test(o_1)$	2: $\neg\Diamond test(o_1)$
	o_2	$d(o_2)$, $\neg sp(o_2)$, $\neg test(o_2)$	1: $\Diamond test(o_2)$
3	o_1	$d(o_1)$, $sp(o_1)$, $test(o_1)$	2: $\neg\Diamond test(o_1)$
	o_2	$d(o_2)$, $\neg sp(o_2)$, $\neg test(o_2)$	1: $\Diamond test(o_2)$

Table 2 shows a possible run of a bloodbank. First, an object o_1 is entered into the system. o_1 is a donation of source plasma ($d(o_1)$ and $sp(o_1)$ are true).

When a donation is added, its test predicate is initially false. Then, an object o_2 is added, which is a donation but not of source plasma. In the third step, the object o_1 is tested. Unless the run is extended to test o_2, the bloodbank doesn't conform to the statements (6) and (7). We now discuss how the annotations are arrived at, and used to assess the regulation.

We first evaluate 7.**p**: $d(y) \wedge sp(y) \rightsquigarrow \neg\Diamond test(y)$ with respect to all variable assignments. When y is assigned the value o_1, the precondition $d(y) \wedge sp(y)$ is true, and we *annotate* the state with 7: $\neg\Diamond test(o_1)$. This annotation happens regardless of whether $\neg\Diamond test(o_1)$ is true or false under the variable assignment.

Next, we evaluate 6.**o**: $d(x) \wedge \neg \text{by}_7(\neg\Diamond test(x)) \rightsquigarrow \Diamond test(x)$. When x is assigned the value o_1, $d(x)$ is true. To evaluate $\text{by}_7(\neg\Diamond test(x))$ we check if there is an annotation (ψ) on the state such that $\psi \Rightarrow \neg\Diamond test(o_1)$ is valid, i.e., a theorem in LTL. Since, $\neg\Diamond test(o_1)$ is an annotation this is an appropriate candidate for ψ and we conclude that $\text{by}_7(\neg\Diamond test(x))$ is true. Hence the precondition $d(x) \wedge \neg \text{by}_7(\neg\Diamond test(x))$ is false, and the obligation is vacuously satisfied.

When considering a non-source plasma donation (o_2), no annotation is provided by 7.**p**: $d(y) \wedge sp(y) \rightsquigarrow \neg\Diamond test(y)$. We will not be able to find ψ such that $\psi \Rightarrow \neg\Diamond test(o_2)$ is valid. This will make the precondition $d(x) \wedge \neg \text{by}_7(\neg\Diamond test(x))$ true, and if a test is not performed eventually, a violation will be detected. Violations are detected only with respect to obligations. Permissons do not produce violations and are relevant to conformance only via references from an obligation.

Complexity: In [5], we show that conformance checking is hard for EXPTIME. The high complexity is due to the satisfiability tests that are needed to evaluate references. A case study of the FDA CFR motivated a restriction, called *the single copy property*, which allows us to compile out the satisfiability tests [6]. For acyclic regulations, the compilation procedure yields first-order temporal logic statements. Conformance checking with first-order logic is PSPACE-complete. However, the exponential factor is determined by the maximum predicate-arity, which tends to be small. [6] describes algorithms for checking conformance at runtime, and an evaluation using a prototype implementation.

Related Work: The logic describe here is a starting point in adding references to systemes such as [2, 3]. [2] represents business contracts as SQL queries, and [3] uses first-order logic augmented with real time operators. References can be added to these systems, provided that the existential quantification is relativized to either the preconditions or the postconditions. However, restrictions are needed to ensure that the satisfiability tests remain decidable. [1] discusses the importance of anlayzing references, but do not provide a formalization.

4 NLP as an Aid in Formalizing Regulation

In this section, we discuss preliminary work on using natural language processing (NLP) to aid in creating a logic-based representation of regulation. We emphasize

that we use NLP purely as an assistive technology and do not attempt to replace the human user.[4]

We approach the problem using the supervised learning methodology, which has been used for a variety of tasks in NLP, e.g., parsing, computing predicate-argument structure, named-entity recognition etc. The supervised learning methodology proceeds as follows:

1. Define a representation (logic) to be computed from the text
2. Manually describe/annotate how units of text correspond to units of the desired representation. The number of examples manually annotated depends on the needs of the application.
3. Train a (statistical) learning algorithm to compute the representation.

Our focus upto this point has been on Steps 1 and 2, i.e., designing the logic and formulating an annotation scheme to associate natural language and logic. In order for the methodology to be successful, it should be possible for a human to describe how she went from natural language to logic. Such a description would take the form of an annotation guideline. For example, [7] gives guidelines for annotating phrase structure on sentences, and [8] gives guidelines for annotating discourse relations. The process of formulating guidelines is typically one of iterative refinement. We begin by fixing a representation, and then annotating a few sentences with this representation. The problematic cases are analysed, resulting in revisions to the guidelines, and the process repeats.

We have made three annotation passes over 100 sentences and are in the process of refining guidelines. The rest of this section describes what we are attempting to annotate and the difficulties encountered. In Section 4.1, we decompose the annotation process into three steps. Some of the key problems encountered are discussed in Sections 4.2 and 4.3. Section 4.4 discusses related work. A preliminary discussion of our approach appears in [9]. We have since extended the logic and annotation guidelines.

4.1 Translating Regulatory Sentences to Logic

Many logics are semantically adequate for the application of conformance checking. However, to be able to describe or annotate how a statement in logic is obtained from natural language, the logic and natural language need to be syntactically isomorphic. (8) and (9) are examples of what we mean by syntactically isomorphic statements in natural language and logic.

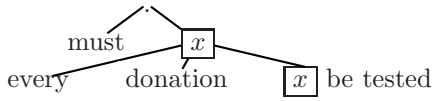
(8) Every donation must be tested

(9) 8.**o**: $donation(x) \rightsquigarrow tested(x)$

We note that predicates such as $tested(x)$ need to be refined to accomodate references between laws, and we discuss this issue in what follows. We now sketch the procedure for associating (8) and (9).

[4] The intended users of our system are designers of software in the organization being regulated.

First, (8) is mapped to the *abstract syntax tree* (AST):



The AST is obtained from (8) by moving the modal *must*, followed by moving the phrase *every donation* to the front of the sentence. While moving a word or phrase, a variable is optionally left behind as a placeholder.

The second step is to associate leaf nodes which are not the leftmost child of their parent with components of the formula. *donation* is associated with the predicate $donation(x)$, and *x must be tested* is associated with the predicate $tested(x)$.

Given the associations for non-leftmost leaves, the leftmost leaves are associated with operations that combine the associations of their siblings to create an association for the parent. *every* is associated with an operation that combines the associations of its siblings to associate $donation(x) \rightsquigarrow tested(x)$ with its parent. Finally, *must* is associated with an operation that takes $donation(x) \rightsquigarrow tested(x)$ and associates 8.**o**: $donation(x) \rightsquigarrow tested(x)$ with its parent. This procedure for translating natural language to logic can be broken into three steps:

1. Converting a sentence to an AST
2. Associating the non-leftmost leaves of the AST with components of the logic
3. Associating the leftmost leaves with combination operations

This decomposition of the problem is the one adopted (modulo terminology) in theoretical linguistics [10]. Of these steps, our goal is to achieve automation with a good level of accuracy for Step 1. For Steps 2 and 3, we can only envision partial automation in the immediate future. The goal is to design appropriate interfaces to assist the user in performing these steps. We now discuss some of the challenges in associating regulatory sentences with ASTs (Section 4.2). We then turn to a discussion of some issues related to Steps 2 and 3 in Section 4.3.

4.2 Annotating sentences with ASTs

The AST produced from a sentence is a resolution of scope ambiguities. The sentence (8) above is simple in comparison to the sentences that one encounters in regulatory text, where a sentence has multiple noun phrases and modalities. Consider the following sentence from CFR 610.1 (the AST is shown in Figure 1):

(10) No lot of any licensed product shall be released by the manufacturer prior to the completion of tests for conformity with standards applicable to such product.

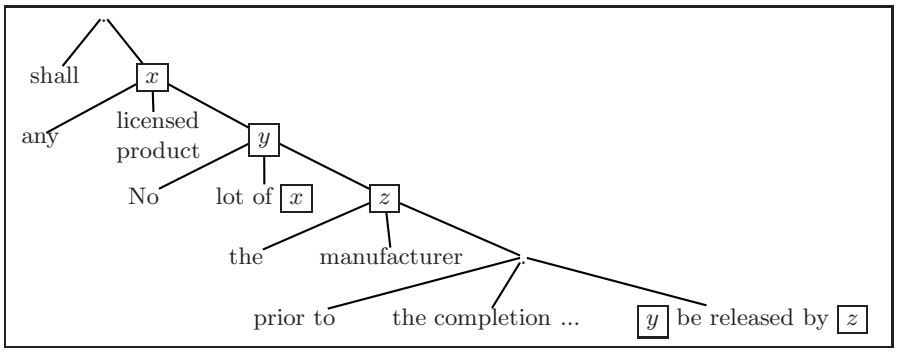


Fig. 1. AST for (10). The structure for the noun phrase *the completion of tests for conformity with standards applicable to such product* is not shown.

(11) 10.**o**: $licensedProduct(x) \wedge lotOf(y,x) \wedge priorTo(\varphi) \wedge manufacturer(z) \rightsquigarrow \neg releasedBy(y,z)$

For simplicity, we omit some details from (11). The phrase *the completion of tests for conformity with standards applicable to such product* involves a reference to other laws, i.e., the applicable standards appear in various places in Part 610. The subformula $priorTo(\varphi)$ in (11) can be formalized using a variant of the technique discussed in Section 3. We now discuss some issues related to (10), (11) and the AST in Figure 1.

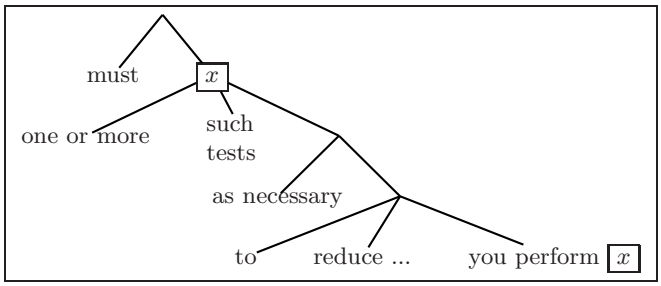


Fig. 2. AST for (12). The structure for the clause *reduce adequately and appropriately the risk of transmission of communciable disease* is not shown.

Consider again the phrase *the completion of tests for conformity with standards applicable to such product*. While we can give this phrase an internal stucture in the AST, we do not know how to associate it structurally to its formal interpretation. In other words, from the perspective of translation, the phrase has to be treated as an idiom of sorts. In annotating a sentence with its AST,

we give such phrases an internal structure and leave the problem of treating it as an idiom to subsequent steps.

Another issue is the question of what to move. In many linguistic theories, only quantificational noun phrases, e.g., *any product*, are treated as candidates for movement. In our annotation scheme, the constructs that are moved are noun phrases, coordinated and subordinated phrases/clauses, relative clauses, and some modals and adverbs. This lets use describe the scopal interaction of these constructions without having to construct a separate phrase structure tree, thus saving annotation effort.

A difficulty in annotating ASTs is that there are many constructions in natural language which we do not know how to formalize. Consider the following statement:

(12) You must perform one or more such tests, as necessary, to reduce adequately and appropriately the risk of transmission of communciable disease.

In (12) it is unclear how to order *one or more such tests*, *as necessary* and *to reduce adequately and appropriately the risk of transmission of communciable disease*. The intended interpretation of this sentence (which is also unclear) is "if a test is required, then it must be performed repeatedly until a conclusive result is obtained".[5] In such cases, we construct an AST following the surface order of the phrases, as shown in Figure 2.

The point to take from this discussion is that not all the structure provided in an AST can be mapped directly to logic. On occasion one has to "undo" some of the movements in order to perform the association. An analogous situation arises in the problem of alignment in machine translation (between natural languages). One cannot always find a syntactically isomorphic translation of an English sentence into French. Certain constructions have to be treated idiomatically. In translating to logic, the number of constructions that we have to treat idiomatically give us a way to evaluate the syntactic expressive power of the logic. If there are many such constructions, it would suggest that the logic needs to be extended. We now discuss issues in associating the leaves of the AST with formulas in logic.

4.3 Associating the leaves of ASTs with logic

In Section 4.1, we gave the AST for the sentence "every donation must be tested". The leaf node "x be tested" was associated with the predicate $tested(x)$. In order to accomodate references between laws, we need to be able to infer, for example, that "if no tests are required, then a test for Hepatitis B is not required". Such inferences would not succeed if we used predicates such as $tested(x)$ and $testedForHepatitisB(x)$ (we need $\neg tested(x) \Rightarrow \neg testedForHepatitisB(x)$ to be valid). To handle such cases, we approach the definition of the set of predicates in two steps, which we describe below.

[5] The intended interpretation was clarified in a memo released by the FDA.

The first step is creating a schema. A schema is a set of class definitions. A class definition consists of a set of attribute definitions, and an attribute definition is a name associated with a type. The types of attributes are taken to be either atomic values (numbers or strings), references or sets of references. For example, the class *Donation* has an attribute named *tests* which is a set of references to objects in the class *Test*.

Predicates are treated as assertions over instances of the schema and are defined using a description logic. The logic that we use combines graded modal logic (modal logic with counting quantifiers), and hybrid logic (which allows one to refer to particular objects). The predicate "$testedForHepatitisB(x)$" is formalized as: $@_x(\exists tests : (purpose = \text{Hepatitis B}))$, read as "at the object referred to by x, there is an object referenced in the *tests* attribute and the *purpose* attribute of the test object has the value "Hepatitis B".

Given a set of documents, there are many ways one could create a schema, depending on domain knowledge and taste. Designing NLP-based interfaces to aid in the extraction of schemas has been explored in the past [11, 12]. Our goal is to adapt previous work to the regulatory domain. Both creating the schema and defining the predicates will require significant manual intervention.

The key challenge in translating natural language to logic (for our application) is being able to decompose the problem into steps that depend mostly on the text, and steps that depend mostly on domain knowledge. We believe that the computation of ASTs and the creation of schemas can be tied closely to the text, and as more documents are formalized better accuracy can be achieved. The problem of creating predicate definitions would benefit from further decomposition, and to our knowledge, this is an open problem.

4.4 Related Work

The problem of translating natural language to logic has received much attention in theoretical linguistics [13]. There are many problems both in the design of logic and the translation procedure that have yet to be resolved. More recently, there have been efforts in NLP to automate this translation for some applications.

[14, 15] show that a good degree of automation can be achieved when the text is constrained. The sentences considered are queries to geographical database, e.g., "Which states does the river Delaware run through?". The specific corpus considered associates each sentence to logic. The associations between components of a sentence and logic are computed during the learning phase. While this approach reduces the annotation effort, the inference of associations during the learning step becomes more difficult.

[16] describes a corpus annotated using a manually crafted Head-driven Phrase Structure Grammar (HPSG). In addition to parse trees, a translation to logic is associated. The logic produced is similar in spirit to the ASTs that we annotate. We do not adopt this approach directly for two reasons. First, the annotation of ASTs avoids the overhead of creating phrase structure trees. Second, the logic produced in [16] introduces a large number of predicates (approximately one per word), and this makes the formulas large and difficult to

refine. The leaves of the AST are typically phrases, and we have found in case studies that it is easier to define predicates at this level of granularity.

[17] discusses an approach to computing wide-coverage semantic interpretation. The goal is to be able to produce approximate translations in first-order logic and carry out inferences. Similar problems arise in the definition of predicates. The envisioned applications are those for which some errors in the logic produced are tolerable. For our application, while it may be impossible to avoid errors, the goal is to provide a correct translation of a sentence. This involves a careful analysis of modalities, which is not possible in current wide-coverage techniques.

5 Conclusions and Future Work

We have motivated the need for a formal representation of regulation to accomodate references between laws (Section 2). We described, in Section 3, a logic that accomodates certain kinds of references, i.e., those appearing in preconditions. There is also the need for reference in postconditions, to express naturally cases where one law cancels obligations and permissions given by another. We are currently working on extending the logic to allow such references.

In Section 4, we described preliminary work on using NLP to assist in creating the formal representation of regulation. In NLP, the focus has been on computing information tied to the surface structure of the sentence, such as parse trees and predicate-argument structure. However, in formalizing requirements, we are often interested in inferences drawn from sentences and the context. Relating these inferences back to the surface structure of a sentence poses interesting challenges to both NLP and formal methods.

We have focussed entirely on regulatory requirements in this paper, and designed machinery to accomodate its peculiarities. The logic that we have developed is useful for expressing rules with a large number of exceptions. Since the logic and the annotation of ASTs are not independent, there will be challenges in adapting the approach to different kinds of requirements. A particularly challenging aspect is the large number of modalities in natural language. In the regulatory texts that we have examined, *time* and *obligation* are the salient modalities, but this may not be the case for other kinds of requirements. A study of requirements in different domains is a topic for further research.

References

1. Breaux, T.D., Vail, M.W., Anton, A.I.: Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In: Proceedings of the 14th IEEE International Requirements Engineering Conference. (2006)
2. Abrahams, A.: Developing and Executing Electronic Commerce Applications with Occurrences. PhD thesis, Univeristy of Cambridge (2002)
3. Giblin, C., Liu, A., Muller, S., Pfitzmann, B., Zhou, X.: Regulations Expressed as Logical Models (REALM). In Moens, M.F., Spyns, P., eds.: Legal Knowledge and Information Systems. (2005)

4. Miltsakaki, E., Prasad, R., Joshi, A., B.Webber: The Penn Discourse Treebank. In: LREC. (2004)
5. Dinesh, N., Joshi, A., Lee, I., Sokolsky, O.: Reasoning about conditions and exceptions to laws in regulatory conformance checking. In Submission: http://www.cis.upenn.edu/~nikhild/reasoning.pdf (2008)
6. Dinesh, N., Joshi, A., Lee, I., Sokolsky, O.: Checking traces for regulatory conformance. In: Proceedings of the Workshop on Runtime Verification (to appear). (2008)
7. Bies, A., Ferguson, M., Katz, K., MacIntyre, R.: Bracketing guidelines for Treebank II style Penn Treebank Project. ftp://ftp.cis.upenn.edu/pub/treebank/doc/manual/root.ps.gz (1995)
8. The PDTB Group: The Penn Discourse Treebank 1.0 Annotation Manual. Technical Report IRCS-06-01, IRCS (2006)
9. Dinesh, N., Joshi, A.K., Lee, I., Webber, B.: Extracting formal specifications from natural language regulatory documents. In: Proceedings of the Fifth International Workshop on Inference in Computational Semantics. (2006)
10. May, R.: Logical Form: Its structure and derivation. MIT Press (1985)
11. Overmeyer, S.P., Lavoie, B., Rambow, O.: Conceptual modeling through linguistic analysis using lida. In: 23rd International conference on Software Engineering. (2001) 401–410
12. Bryant, B.R.: Object-oriented natural language requirements specification. In: ACSC 2000, The 23rd Australasian Computer Science Conference. (Jan 2000)
13. Heim, I., Kratzer, A.: Semantics in Generative Grammar. Blackwell (1998)
14. Zettlemoyer, L.S., Collins, M.: Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In: Proceedings of UAI. (2005)
15. Wong, Y.W., Mooney, R.J.: Learning synchronous grammars for semantic parsing with lambda calculus. In: Proceedings of ACL. (2007)
16. Oepen, S., Flickinger, D., Toutanova, K., Manning, C.: LinGO Redwoods: A rich dynamic treebank for HPSG. In: Proceedings of the workshop on treebanks and linguistic theories. (2002)
17. Bos, J., Clark, S., Steedman, M., Curran, J.R., Hockenmaier, J.: Wide-coverage semantic representations from a CCG parser. In: Proceedings of COLING. (2004)