Logic-Based Regulation Compliance-Assistance

Shawn Kerrigan
Stanford University
Dept. of Civil & Environmental Eng
Stanford, CA 94305-4020
kerrigan@stanford.edu

Kincho H. Law
Stanford University
Dept. of Civil & Environmental Eng
Stanford, CA 94305-4020
law@stanford.edu

ABSTRACT
This paper focuses on the creation of a first order predicate calculus based regulation compliance-assistance system built upon an XML framework. Two areas of research that support the development of the compliance assistance system are discussed. The first is a document repository containing federal and state regulations and supplemental documents. The second is an XML framework for representing regulations and associated metadata. The prototype effort for the regulation assistance system has been focused on environmental regulations and related documents. The compliance assistance system is illustrated in the domain of used oil management. The objective of this research is to develop a formal information infrastructure for regulatory information management and compliance assistance.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval; [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods – *Predicate logic*; J.1 [**Administrative Data Processing**]: Law.

General Terms
Design, Performance, Theory.

Keywords
Regulations, Legal Informatics, Compliance Assistance.

1. INTRODUCTION
Industrial production activities in the United States that produce byproducts classified as hazardous waste must comply with regulations regarding the handling and fate of such materials. In the United States, both federal and state, as well as local governments, have strict regulations imposed on the treatment and disposal of such chemical wastes. Environmental regulations are complex and voluminous. The largest of the generators have the staffs to deal with the regulatory agencies and, often, the resources to contract with specialized treatment, storage and disposal facility (TSDF) companies to manage their waste. However, environmental regulations can be disproportionately burdensome on small businesses, since these businesses often do not have the resources to staff personnel trained to deal with these complicated regulations and procedures [1, 2]. Many government regulations are now available online. However, most of current online portals are primarily designed for displaying the information for experienced users and are difficult to use for further processing. Information technology (IT), if properly designed and developed, has the potential to mitigate and help solve many of these complicated issues. Through the application of advanced information technologies and development of new methodologies, the REGNET research project aims to develop a formal information infrastructure for regulatory information management and compliance assistance.

This paper focuses on the creation of a first order predicate calculus based regulation compliance-assistance system built upon an XML framework. We first briefly describe two areas of research that support the development of the compliance assistance system. The first is a document repository containing federal and state regulations and supplemental documents. The second is an XML framework for representing regulations and associated metadata. We then describe in detail the prototype effort for the regulation assistance system. The regulation assistance system is illustrated in the domain of used oil management.

2. DOCUMENT REPOSITORY
An objective for the REGNET information infrastructure has been the development of a document repository for environmental regulations. The scope of our current prototype development covers US Code of Federal Regulations Title 40 (40 CFR): Protection of the Environment, along with selected supplementary and supportive documents that focus on regulations covering hazardous waste and the management of used oil. Supplemental documents are important because they often contain information that is necessary for the accurate interpretation of the federal regulation(s) to which they refer [3]. Supplementary documents may come in the form of administrative decisions, guidance documents, court cases, letters from the general counsel and letters of interpretation from the Environmental Protection Agency (EPA). The REGNET document repository is designed to make these important documents more accessible. The contents of the repository are available through the mediation of one or more searchable concept hierarchies, or through a regulation assistance system described later in this paper.

ICAIL '03, June 24-28, 2003, Edinburgh, Scotland, UK.

3. XML REGULATION FRAMEWORK AND METADATA

We have developed an XML framework for environmental regulations. The framework is document centric and includes XML tags for each level of regulation text – for example part, subpart, section or subsection – that mirrors the standard structure of regulations. This framework results in a hierarchical structure for the regulations, with regulation text attached throughout. Figure 1 shows how a regulation can be decomposed into a hierarchical tree structure. Figure 2 shows an abbreviated sample of how we represent this hierarchical structure in XML. Parsing systems have been built to transform federal and state environmental regulations from Portable Document Format (PDF) and HTML into the REGNET XML framework. With XML, it is possible to augment a regulation with various types of annotation and regulation-specific metadata rather than to simply structure the regulation according to how it should be displayed. With respect to the document repository, the metadata types currently added to the regulation framework include concept tags, reference tags and definition tags.

The concept tags allow dynamically generating links to related supporting documents in the document repository. This is useful because supporting documents and regulations may not directly reference each other even when they address the same topic. The automatic application of concept tags to the XML framework means that as new supporting documents are added to the document repository, regulations stored in the framework can automatically be linked to them via the terms that they share in common. Concept tags can be generated "semi-automatically" using existing text mining and information retrieval tools [4]. Currently, we use software from Semio Corp. to extract, clean and define over 65,000 concepts for 40 CFR regulations and to categorize the concepts according to different interests and applications.

Regulation provisions tend to contain a large number of casual English references to other provisions. These references are cumbersome to look up manually, and they reduce the readability of the regulation text itself. Simple references (for example, "as stated in 40 CFR section 262.14(a)(2)") and complex references (for example, "the requirements in subparts G through I of this part") exist throughout regulations. Given the large volume of federal and state environmental regulations, a manual translation of references would be too time consuming. A parsing system was developed using a context-free grammar and a semantic representation/interpretation system that is capable of tagging regulation provisions with the list of references they contain. Instead of building hyperlinks, which tie the reference to a particular source for the referred document, the reference tags provide a complete specification for *what* regulation provision is referenced. *Where* the regulation is located is not specified so that a viewing system may select from any document repository of regulations to retrieve the referenced provision. This gives better flexibility than a rigid hyperlink structure for maintenance and scalability.

The large number of domain-specific terms and acronyms that appear in regulations can make regulation text difficult for novices to understand. We standardize all definitions with XML

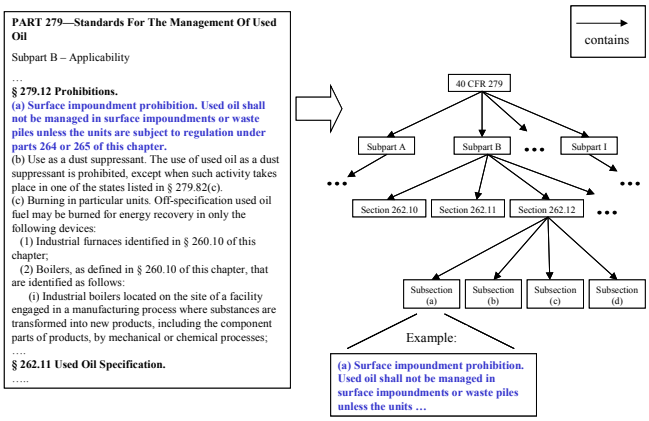

Figure 1. Decomposition of regulation into a tree structure

```
<regulation id="40.cfr.279" name="Standards For The Management Of Used Oil" type="federal">
  <regElement id = "40.cfr.279.A" title = "Subpart A">
    < regElement id = "40.cfr.279.12" title = "Prohibitions">
      < regElement id = "40.cfr.279.12.a" title = "Surface Impoundment prohibition">
        <regText>
          <paragraph>Used oil shall not be managed in surface impoundments or waste
            piles…</paragraph>
        </regText>
      </regulation>
      <regulation id = "40.cfr.279.12.b" title = "Use as a dust suppressant">
      </regulation>
    </regulation>
  </regulation>
</regulation>
```

Figure 2. Abbreviated XML representation of regulation tree structure

elements, which allow regulation viewing systems to incorporate explicit definitions of terms and acronyms into their user interfaces.

4. REGULATION ASSISTANCE SYSTEM

4.1 Overview

There has been a push in the United States by the executive office for government agencies to put more emphasis on compliance assistance in lieu of enforcement to encourage companies to comply with regulations [5, 6]. Towards this end, specialized modules using expert system technologies have been built to assist specific applications and business types in understanding regulation requirements [7]. In these systems, references to the regulations are not explicitly linked. Our research on developing a compliance assistance infrastructure builds upon the XML regulation framework and takes advantage of the regulation metadata described in Section 3.

Besides the concept, reference and definition tags, we add logic and control processing metadata to the REGNET regulation framework. Logic metadata comes in two variations. There is only one form of control processing metadata. Regulation logic metadata represents a rule or concept from a regulation using First Order Predicate Calculus (FOPC) logic sentences. These logic sentences are used to represent the rules that must be followed for an entity to be in compliance with the regulations. User interface logic metadata uses FOPC logic sentences to represent compliance questions and a list of possible user answers to those

questions. Control processing metadata provides information about what provisions of a regulation need to be checked for compliance. Each type of logic or control processing metadata can be associated with any regulation provision in the document. In the REGNET framework, these three types of metadata are necessary for the system to be able to verify compliance with a regulation. However, they must be specified by a domain expert as they cannot be generated automatically. For the purposes of demonstration, a federal used oil regulation (40 CFR 279) has been manually tagged with regulation logic metadata, with user-interface logic metadata, and with control processing metadata.

We built a regulation assistance system (RAS) to demonstrate how the regulation meta-data can be used. The RAS functionality is implemented by a web interface that communicates with a compliance checking system. The compliance checking system interacts with a theorem prover component. The structure of this system is shown in Figure 3. The compliance checking system controls the process used to check for violations. First, it parses the XML-structured regulation to extract the information necessary to run a compliance check. The XML structure allows the system to properly scope the meta-data and reduce the amount of extraneous data passed to the reasoning system. Only the logic and control processing metadata necessary for the compliance check are acquired and dynamically loaded into the reasoning system. This is important because the performance of FOPC theorem provers decreases rapidly as the number of logic sentences used for reasoning increases. The system design is such that any FOPC theorem prover can be used to perform the logic checks. Presently, we employ Otter, a publicly available theorem prover developed at the Argonne National Laboratory [8].

The primary feature of this web-based compliance assistance system is that it helps guide the user through regulations. In order to facilitate greater understanding of the regulations, the system

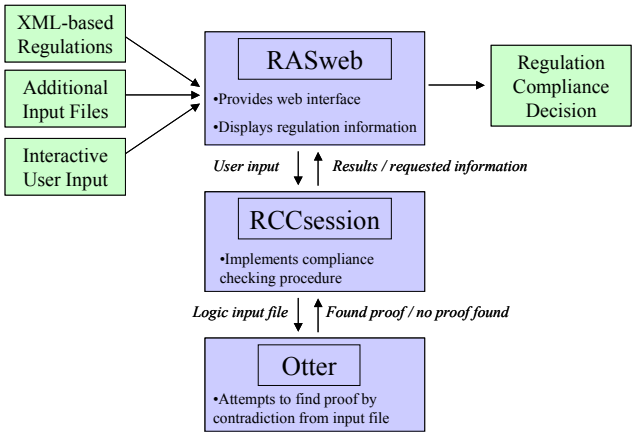

Figure 3. Diagram of the Regulation Assistance System's structure

makes available a number of enhancements while guiding the user through a compliance check, utilizing the metadata with which the regulations are tagged. The system can automatically insert links to any referenced regulation provisions and display terms and definitions. Key conceptual phrases for the provision are displayed and linked, enabling instant access to repository documents related to the provision. The use of concepts, definitions, and references is shown in Figure 4.

A web interface asks users questions based on information in the XML logic metadata. Users may select a response from a menu of possible responses, including an "I don't know" option that forks the compliance-checking process along all possible answers. The ability to allow users to fork the compliance process along all possible paths at any time is useful for exploring different scenarios. When the system completes a check against the

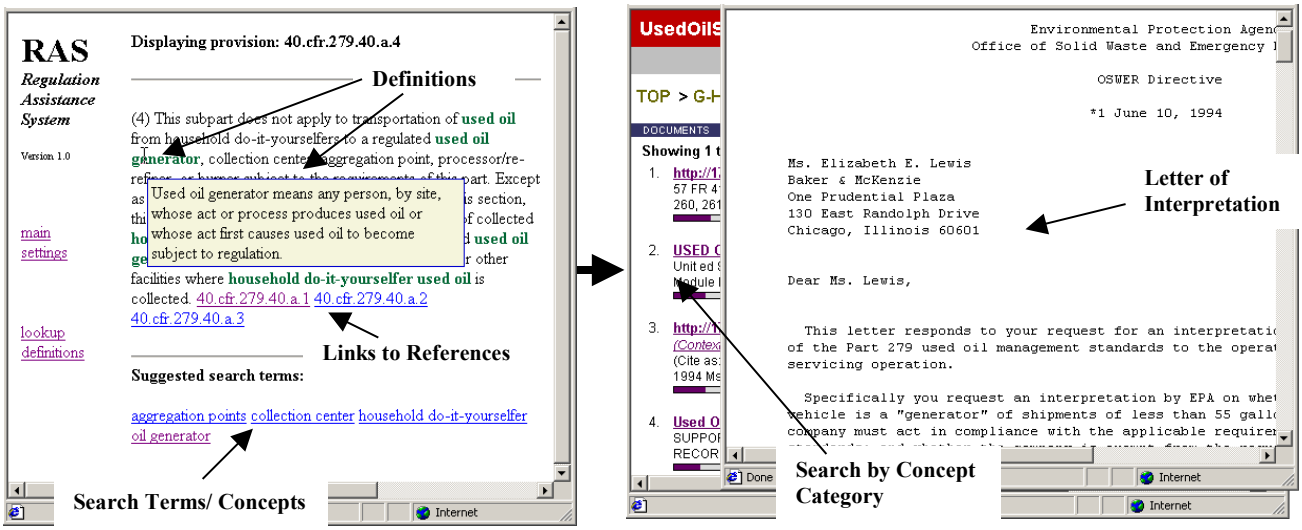

Figure 4. Definition, reference and concept usage

regulation provisions or detects a conflict between the user's answers and the regulation, it displays a summary of the question-and-answer history as well as the results of the compliance check. The use of and results produced by the system are illustrated in Figure 5 below. Downloadable logs of completed compliance checks allow users to maintain detailed records of their compliance checks, a feature companies we met with felt would be extremely valuable for record keeping or when revisiting the regulations at a later date. The logs of compliance checks can also be uploaded and edited for future compliance checks against the same or updated regulations.

4.2 Logic and control processing meta-data

In order to facilitate the development of compliance assistance systems, we developed XML elements to provide processing information for systems interpreting the regulation document. We developed control elements to specify what regulation provisions need to be processed, and logic elements to represent compliance information.

4.2.1 Control processing elements

The control element is a wrapper element that may contain one or more instructions within it in the form of one or more of three sub-elements: goto, switchTo, or end. These control elements allow regulation designers to specify what regulation provisions may or may not need to be investigated. While not FOPC in nature, control elements provide processing logic and therefore may be used within the logic XML elements that are discussed in Section 4.2.2. These control elements are currently added manually, but it would be possible to take advantage of the automatically-generated reference elements to partially-automate the process of adding control elements.

The goto control element specifies a regulation provision that the system should process next; returning to the current provision once the specified provision has completed its check. The goto element is useful when it is necessary to check an additional regulation provision without abandoning the current line of processing. For example, frequently a regulation provision will refer readers to another regulation provision that should be read before continuing. The goto element instructs a system to temporarily go to the specified regulation provision, but to return to the currently provision eventually.

Similarly, the switchTo element specifies a regulation provision to process next, but processing should not return to the current provision once the specified provision has completed its check. This is useful when a regulation provision specifies some conditions under which a different regulation provision will apply. The switchTo element instructs the system that the check against the current regulation provision is complete, and that processing should continue starting with the regulation provision specified by the switchTo element.

Figure 6 demonstrates the usage of the goto and switchTo elements. This example illustrates an instruction to process section 279.65, and once that section is complete to switch processing to section 279.73. Note that in order to specify processing control to move to a reference within the regulation the control attribute "target" is used. For example, target = "40.cfr.279.65", refers the compliance processing system to section 279.65 in 40 CFR.

The end element signals that the check of the specified provision is complete. This is useful when the regulation specifies that under certain conditions the check against the current provision need not go any further. Since regulation checks may be done at any level of the regulation document, it is important to specify a target reference for the end element. For example, if during a check of the regulation Section 40.cfr.279.12 an end element is encountered that specifies that 40.cfr.279.12.a is complete, it is important to realize that the check against 40.cfr.279.12 is not finished and should continue. On the other hand, if section 40.cfr.279.12.a is being checked and an end element is encountered that specifies 40.cfr.279 is complete, processing of the current provision can stop. Figure 7 demonstrates the usage of the end element. This example instructs the system that the compliance checking for provision 40 CFR 279.12 is complete.

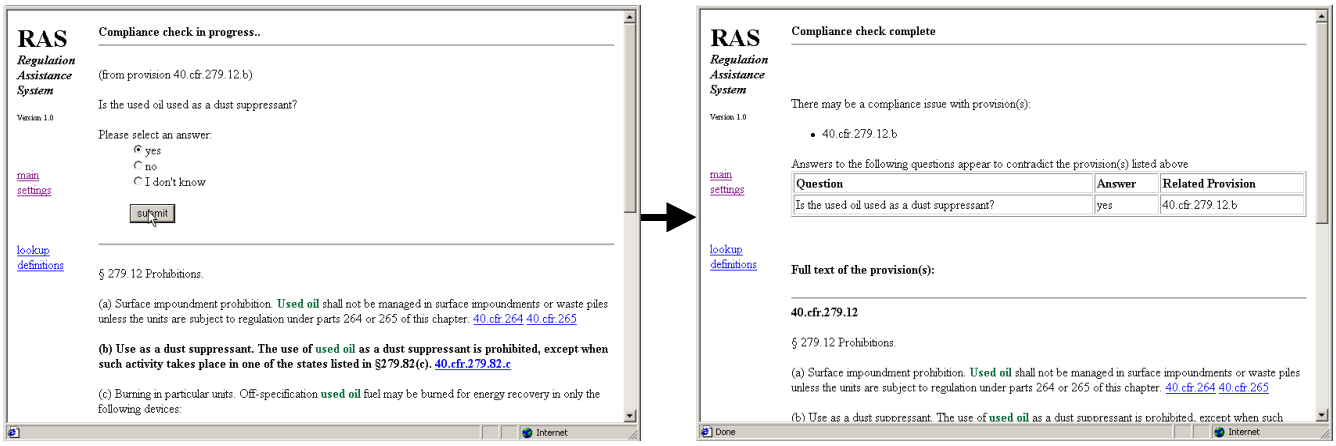

Figure 5. Example compliance-checking session

```
<control>
        <goto target = “40.cfr.279.65” />
        <switchTo target = “40.cfr.279.73” />
</control>
```

Figure 6. Goto and switchTo element usage

```
<control>
        <end target = “40.cfr.279.12” />
</control>
```

Figure 7. End element usage

4.2.2 Adding Logic to XML regulations

Logic can be added to the XML-based regulation document to facilitate manipulation and interpretation of the document by logic-based systems. Internal contradictions within the regulation can be checked for, contradictions between regulation documents can be identified, and compliance checking systems can be built to verify that a user is in compliance with the regulation. This section discusses the tagging of environmental regulations with logic, and will be followed up by a discussion of the algorithm used by a prototype system for checking compliance with the regulations.

The approach of tagging XML structured regulations with FOPC introduces an open platform consisting of structured text and embedded logic that is a significant improvement over unstructured text variants. The use of FOPC makes the work accessible and usable by a wide variety of automated deduction systems.

Logic elements can be added to the XML structure within regElement XML elements. The logic elements are denoted by “logic” tags, and may contain either logicSentence or logicOption. These elements are described in detail below. All of the logic added to regulations in the current implementation is done manually.

The logic sentences are written in FOPC, which provides the expressive power with which to model the meaning of environmental regulations. While the exceptions in the regulation rules can introduce an element of non-monotonicity, the closed domain of the regulation scope may make this a tractable problem in FOPC. The choice of FOPC also introduces a great deal of flexibility for choosing a reasoning system, since there are many reasoners available for working with FOPC.

Logic sentences representing the ideas laid out by a regulation are added to an XML regulation document in logicSentence elements that may be placed within “logic” elements. These logicSentence elements may be used to tag regulation provisions throughout the document with their logical meaning. The flexible placement of the logicSentence element enables the tagging of any provision within the document with a meaning. For example, tagging the root regulation element with a logicSentence element specifies that the logic sentence should be applied to the entire document. The logicSentence elements are generally used to define the rules and concepts expressed in a regulation.

Figure 8 illustrates the usage of a logicSentence element. The logicSentence element describes a rule that used oil may not be used as a dust suppressant. The rule states that for all objects “o”, if “o” is used oil then “o” cannot be a dust suppressant. The use of “ForwardImplies” instead of the more common logic syntax “->” is necessitated by the XML standard, and is described in greater detail in Section 4.2.3.

Another logic element was introduced to handle the issue of user input. The logicOption element can be used to build a structured question and answer system that constructs logic sentences based on the user’s input. Without the “logicOption” elements, interacting with the system would require the user to work with FOPC sentences.

The logicOption element contains one question element and one or more answer elements. The question specifies the text that can be used to prompt a user for input. The answer element contains a possible answer to the question and the logic that should be associated with that answer. Since answers are tied to logic statements, the user can interact with the system in plain English, but the answers are mapped to logic statements so that they can be used for compliance checking. The logicOption element allows logic statements to be specified for compliance checking without requiring the user to construct FOPC sentences on their own.

Figure 8 illustrates the usage of a logicOption element that assists with gathering user input. This particular element maps the user’s response to a question about the use of used oil to logic statements that reflect the user’s answer. For example, a compliance assistance system might ask the question "Is the used oil used as a dust suppressant?", and provide the option of answering "yes" or "no". If a user selects the "yes" answer, the system would know to match the response to the logic sentence "(usedOil(oil1) AND dustSuppressant(oil1)).".

```
<logicSentence>
  all _o  (usedOil(_o) ForwardImplies -(dustSuppressant(_o))).
</logicSentence>

<logicOption>
  <question>
    Is the used oil used as a dust suppressant?
  </question>
  <logicOpt answer = "yes">
    <logic_ans>
      (usedOil(oil1) AND dustSuppressant(oil1)).
    </logic_ans>
  </logicOpt>
  <logicOpt answer = "no">
    <logic_ans>
      (usedOil(oil1) AND (-(dustSuppressant(oil1)))).
    </logic_ans>
  </logicOpt>
</logicOption>
```

Figure 8. Usage of logicSentence and logicOption elements

4.2.3 Standard logic syntax and XML standards

One drawback to the use of XML for storing logic representations of regulations is that there are syntactic limitations that must be met to comply with the XML standard. For example, XML elements are defined by the XML standard to start with "<", as in "<regText>". This conflicts with the standard logic syntax used for reverse implications, "<-", and equivalences, "<->". A simple substitution of text provides the solution for this problem, where the illegal XML character sequences are replaced with legal ones.

The substitutions currently being used to represent FOPC in an XML compliant syntax are shown in Table 1. Note that the substitutions for "->" and "|" are not necessitated by XML standards, but are done so that the XML logic uses a consistent representation formalism. The substitutions for "<-", "<->", and "&" are required by XML standards. The substitutions must be reversed by logic processing systems reading the XML regulation so that the standard syntax is used when providing the data to a logic reasoner. The XML compliant substitutions also become reserved words in the logic representation language. Since the words in the right column will be substituted with the logic symbols in the left column, words in the right-hand side of the table are reserved words that cannot be used for logic predicates or function names.

4.3 Compliance-Checking Process

The RAS compliance-assistance processing algorithm proceeds in three stages. First, the XML regulation is verified. Second, interactive processing is done as the RAS system moves dynamically through the regulation. Third, results of the analysis are compiled and presented to the user. Each of the stages is discussed in detail below.

4.3.1 XML Regulation Verification

The RAS system performs two layers of verification checks on an XML regulation before it is used to assist with compliance checking procedures. The first step in verifying an XML regulation is to perform structural verification of the XML. This is done by verifying the regulation against a regulation DTD we developed.

The second step in verifying the XML regulations is to verify that all the logic rules contained in logicSentence elements are consistent. The system extracts all the logicSentence elements from the target regulation and builds an appropriate input file for the theorem prover, Otter. If the theorem prover does not find a contradiction in the logic sentences within a given time period, the core regulation logic is assumed to be consistent. This check attempts to ensure that the set of logic rules embedded in the XML regulation, which were specified by a domain expert, do not contain a contradiction. The initial check for contradictions in regulation rules does not guarantee that there are no contradictions in the rules, since the theorem prover, Otter, is not guaranteed to find a contradiction if one exists. In practice, however, this initial logic check has been fairly robust.

The initial check for problems with the logic rules is important, since if the rules contained a contradiction the logic system would be unable to assist with compliance checking against the regulation. The RAS system identifies potential conflicts with regulation rules by identifying logical contradictions between user input and regulation rules. Therefore, if the regulation rules themselves contain a contradiction the algorithm used by the RAS system will not work.

Table 1. Substitutions for XML compliant logic sentences

Standard logic syntax	XML compliant substitution
->	ForwardImplies
<-	ReverseImplies
<->	EquivalentTo
&	AND
\| | OR |

4.3.2 Compliance Processing Algorithm

Processing of the regulation document is done by a depth-first tree traversal of the XML structure starting at a selected provision. A provisions-to-process (PTP) stack maintains a list of regulation provisions that need to be investigated, and an already-processed-provisions (APP) list keeps track of provisions for which processing is complete. Any XML "control" elements encountered while traversing the regulation tree-structure redirect the flow of processing. The effect of the three control elements on the PTP stack and APP list will be discussed next.

The effect of the goto control element on the PTP stack and the APP list is shown in Figure 9. The initial PTP stack is shown on the left and the resulting stack after taking the control elements into account is shown on the right side of the figure. As the first example shows, in the simplest case the goto element adds the provision specified to the PTP stack. As the second example shows, only a single call to a particular regulation provision may be in the PTP stack at a time. Additional attempts to add the same provision to the stack are ignored so as to prevent infinite loops. The third example illustrates the idea that provisions in the APP list cannot be added to the PTP stack, since they have already been processed. The fourth example demonstrates that even if the system is processing a sub-provision of the top PTP provision, the goto element operates as would be expected.

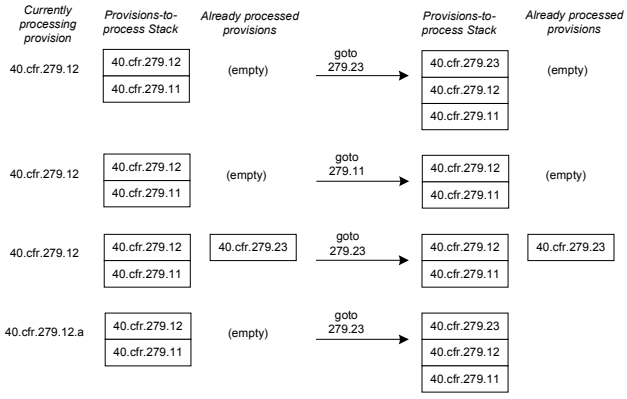

Figure 9. The goto element

The effect of the end control element on the PTP stack and the APP list is shown in Figure 10. Control elements of end type result in the targeted provision being removed from the PTP stack if it exists there, and added to the APP list. The first example in Figure 10 shows the element's basic effect. The end element has no effect if the specified provision is already in the APP list, as shown in the second example. The third example illustrates how sub-provisions of provision in the PTP stack can be added to the APP list. If provisions in the PTP stack are sub-provisions of a provision targeted by an end element, the sub-provisions will be removed from the PTP stack.

The effect of the switchTo control element on the PTP stack and the APP list is shown in Figure 11. The first example in Figure 11 show how in the basic case the top provision is removed from the PTP stack and added to the APP list, and the provision specified by the switchTo element is added to the PTP stack. The second example demonstrates how in cases where the switchTo element refers to previously processed provisions the referenced provision is not added to the PTP stack. The third example shows that if the system is processing a sub-provision of the top PTP provision, the switchTo element adds the provision currently being processed to the APP list and the provision specified by the switchTo element to the PTP stack. This illustrates how the switchTo element is provided for convenience, since it has the same effect as a goto element combined with an implied end element for the current provision.

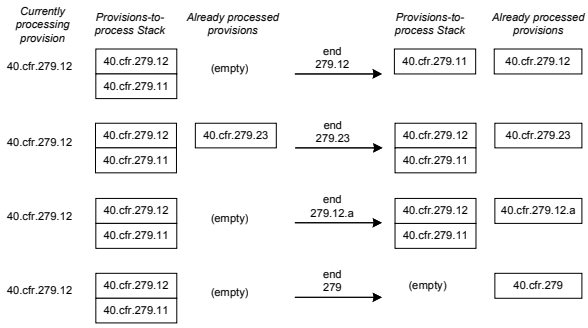

Figure 10. The end element

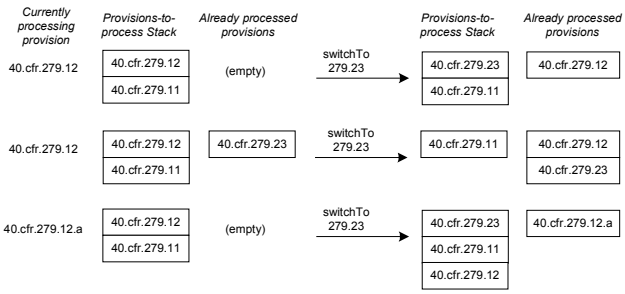

Figure 11. The switchTo element

As logicOption XML elements are encountered during processing, these elements are used to prompt the user for answers to the questions they contain. The logicOption elements provide a mapping from user responses to logic sentences, which can then be verified against logic rules in the regulation. After each question is answered, the logic associated with the selected answer is recorded and any control elements associated with the answer are noted. Otter is then used to check for a contradiction between the logic associated with the user's answers, called response logic, and the rules specified by the regulation.

Figure 12 shows a flowchart diagramming the procedure for identifying conflicts between user response logic and regulation rule logic. First, an input file is prepared for the theorem prover, Otter, with all the regulation rules encountered during processing along with all the logic sentences selected by users in response to questions. Second, Otter is run using the input file. Third, RAS reads the output file from Otter to see if the theorem prover was able to find a contradiction in the input logic sentences. If no proof was found, the logic sentences are assumed to be consistent. If a proof was found, the proof steps are read to find the input logic sentences that contributed to the contradiction. These input logic sentences are then mapped back to the provision rules or the user responses from which they originated. This allows the system to identify what is contributing to the logical contradiction, i.e. non-compliance with the regulation.

Answers to the questions contained in logicOption elements are also recorded in a file. This file enables the system to automatically process questions that have been answered in the past. These answers form a detailed log that can be provided to the user as described in Section 4.1.

The questioning procedure terminates when either a logical contradiction is found or the PTP stack is empty. When the questioning procedure ends due to an empty PTP stack, the system returns a result stating that it appears the user is in compliance with the regulation. When the questioning procedure ends due to a logical contradiction being found, the system returns a result stating that there is a compliance problem and a detailed report is returned to the user to help identify the problem. All the questions, answers and relevant provisions that contributed to the logical contradiction are then displayed for the user. An example screen shot of this process is shown in Figure 5.

Annotating XML regulations with logic elements and processing them in the manner described above has a performance advantage over simply building a large Knowledge Base (KB) of logic sentences. The primary advantage of the approach described is that the number of logic sentences that need to be handled by the reasoning subsystem (i.e. Otter) is reduced. Doing logic proofs is computationally intense, and significantly reducing the number of extraneous logic sentences greatly reduces the processing time for proofs and increases the complexity of problems that can be handled. If a RAS compliance-checking session were to trace over and dynamically load the logic from every provision in a particular regulation, the performance of the RAS system would not be better than that of a system that simply used a complete KB of all the logic sentences in a regulation for reasoning. The expectation of the RAS system is that in general not all provisions in a regulation would need to be processed, so the corresponding number of logic sentences needed for the compliance-checking

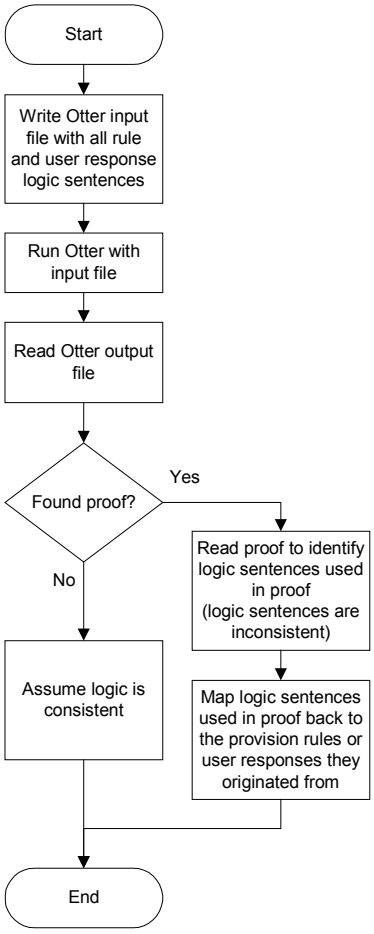

Figure 12. Processing FOPC with Otter

logic proofs will be smaller than in the case of a system that simply uses a complete KB for all logic checks.

This section has described the logic and control elements added to the XML regulations, along with the algorithm that uses these elements for compliance-assistance purposes. A flowchart of the overall processing algorithm is shown in Figure 13.

4.4 The Broader Compliance Perspective

The compliance problem from the perspective of the regulated community can be broken-down into two parts. First, one must determine which sets of regulations one must comply with. Second, one must determine what needs to be done to comply with those regulations. The RAS system primarily addresses the second of these two steps by guiding users through regulations. The RAS system was designed, however, such that it could be used as a component in a larger system that would first assist a user in identifying regulations that need to be investigated. The RAS system can initiate compliance checks at any point within a regulation, and a compliance check can by started by connecting to the RAS system with a target regulation in the URL.

To demonstrate how straightforward it can be to build a compliance guide for a specific application utilizing the RAS

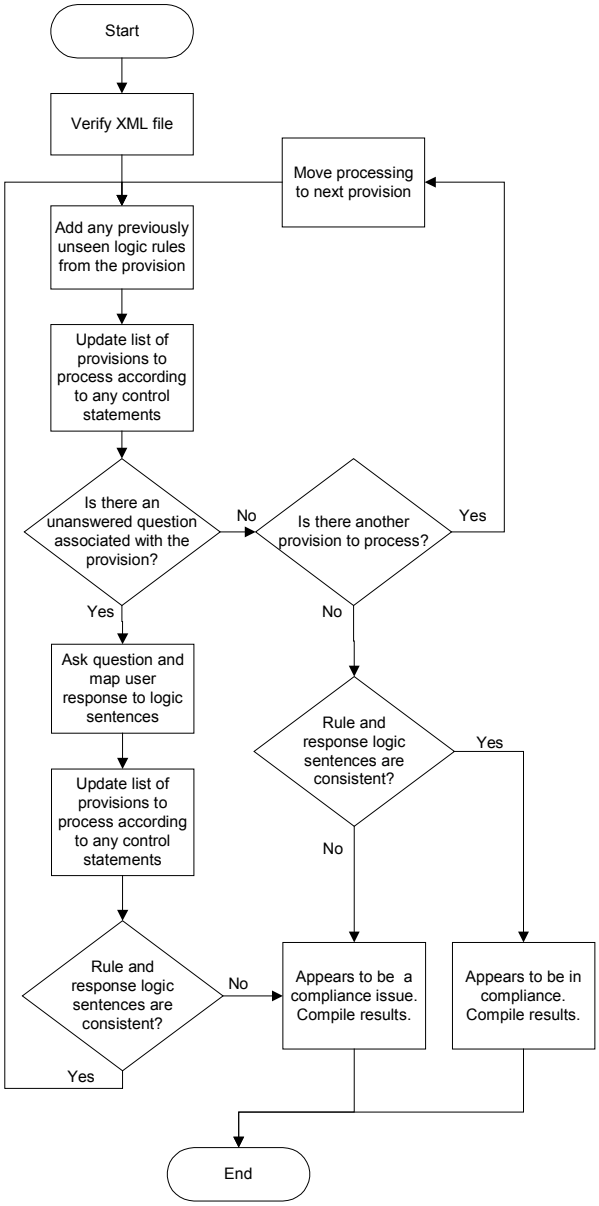

Figure 13. Determining compliance with a regulation

system and the document repository as a back end, a sample online guide was built for vehicle maintenance shops. The vehicle maintenance shop online guide was adapted from a paper-based guide developed by the New York State Department of Environmental Conservation Pollution Prevention Unit [9]. Our adaptation is for demonstration purposes only since it combines some federal and state regulations. Figure 14 illustrates how the demonstration system links into the RAS system to make use of the used oil regulations. The guide targets vehicle maintenance shops and explains what regulations apply to typical work done in that industry. While the original paper-based guide explains requirements and references applicable regulations, our online adaptation provides the additional feature of enabling users to click on referenced regulations and check for compliance

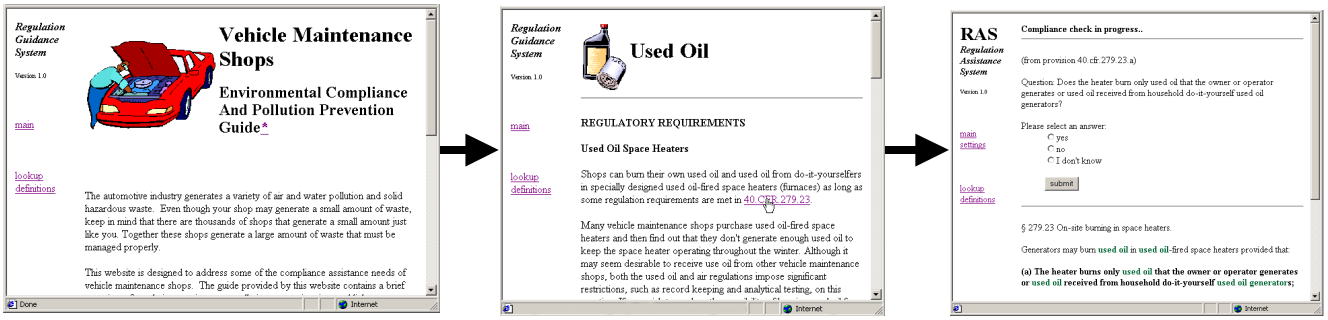

Figure 14. Linking industry-specific guides to the regulation assistance system

by stepping through the regulation itself. Online regulation guides such as the vehicles maintenance shop example located anywhere on the Internet can build upon the compliance-checking capabilities of the RAS system simply by passing target regulations in the URL.

5. USER EVALUATION

In order to gather feedback we have organized a small workshop at Stanford University attended by our key contacts from EPA Region IX office, HP, Palo Alto's environmental compliance office, and legal professionals in the Bay Area. We have also shown demos and have provided access to the compliance assistance system for a number of individuals, government agencies, nonprofit organizations, and large and small companies. Feedback from these interactions has indicated that our approach, if scalable, would be useful to many parties affected by regulations.

6. RELATED WORK

Representation of regulations and laws has been an active research area for decades. There has been a great deal of work done on building expert systems for law [10, 11]. T. Bench-Capon provided a review on the applications of knowledge-based systems for legal applications, particularly the research and development efforts related to the Alvey DHSS Demonstrator project in U.K. [12]. The reference includes several hundreds of citations that appeared before 1990 which are related to logic and rule based approaches and their application in legal systems. Much of the earlier work in IT and law focused on building systems to optimize decisions with respect to laws, particularly tax law [13]. Some of the recent work has focused on investigations into case-based reasoning and information retrieval [14, 15]. Methodologies on tailoring legal documents to users' needs have also been studied [16]. While legal knowledge representation and reasoning has been an active research topic [17, 18], an integrated approach covering the management of regulations, efficient access and retrieval of documents and tools for compliance checking is missing. Wang [19], in his thesis, proposes an integrated and distributed information management infrastructure to support hazardous waste compliance, research work that was a precursor to this work and laid much of the groundwork for the research described in this paper. This research focuses on the issues related to the development of a regulatory information management infrastructure that can also support compliance assistance.

7. SUMMARY

The goal of the REGNET research Project is to develop an information infrastructure for regulatory information management and compliance assistance. This paper describes some of the work that has been done to date on creating a logic-based regulation assistance system, which builds upon an XML-based framework for representing regulations and a document repository.

8. ACKNOWLEDGMENTS

This research project is sponsored by the National Science Foundation, Contract Numbers EIA-9983368 and EIA-0085998. The authors would like to acknowledge a "Technology for Education 2000" equipment grant from Intel Corporation and the support by Semio Corporation in providing the software for this research. The authors would like to thank professors Gio Wiederhold of the Computer Science Department, Barton Thompson of the Law School, and Jim Leckie of the Department of Civil and Environmental Engineering for their valuable suggestions in this project. Last but not least, the authors would like to acknowledge the contributions by the REGNET research team members, Charles Heenan, Haoyi Wang, Gloria Lau and Jie Wang.

9. REFERENCES

[1] Rechtschaffen, C., "Competing Visions: EPA And The States Battle For The Future Of Environmental Enforcement," *Environmental Law Reporter*, 2000.

[2] Romine, M., "Politics, The Environment, And Regulatory Reform At The Environmental Protection Agency," *Environmental Lawyer*, 1999.

[3] Heffron, F.A. and N. McFeeley, *The Administrative Regulatory Process*, Longman, 1983.

[4] Heenan, C., *Manual and Technology-Based Approaches to Using Classification for the Facilitation of Access to Unstructured Text* (Unpublished Manuscript), Engineering Informatics Group, Stanford University, January, 2002. (available at http://eil.stanford.edu/regnet).

[5] *Business Compliance One Stop Workshop*, Small Business Administration, Queenstown, MD, July 24-26th, 2002.

[6] *National Compliance Assistance Providers Forum*. co-sponsored by the U.S. Environmental Protection Agency and Texas Commission on Environmental Quality: San Antonio, TX., Dec., 2002.

[7] Botkin, A., "Wizards, Advisors and Websites, Oh My! Interactive Electronic Tools for Compliance Assistance," presented at the *National Compliance Assistance Providers Forum*, co-sponsored by U.S. Environmental Protection Agency and Texas Commission on Environmental Quality, San Antonio, December, 2002.

[8] McCune, W.W., *Otter 3.0 Reference Manual and Guide*. ANL-94/6, Mathematics and Computer Science Division, Argonne National Laboratory, 1994.

[9] *Environmental Compliance and Pollution Prevention Guide for Vehicle Maintenance Shops*, New York State Department of Environmental Conservation Pollution Prevention Unit, 2002.

[10] Wahlgren, P., *Automation of Legal Reasoning*, Kluwer Law and Taxation Publishers, 1992.

[11] Zeleznikow, J. and D. Hunter, *Building Intelligent Legal Information Systems: Representation and Reasoning in Law*, Kluwer Law and Taxation Publishers, 1994.

[12] Bench-Capon, T.J.M., *Knowledge Based Systems and Legal Applications*. The APIC Series 36, Academic Press, 1991.

[13] McCarty, T., "Reflections on Taxman: An Experiment in Artificial Intelligence and Legal Reasoning," *Harvard Law Review*, 1977.

[14] Stranieri, A. and J. Zeleznikow. "The Evaluation of Legal Knowledge Based Systems," *Proceedings of the Seventh International Conference on Artificial Intelligence and Law*, pp. 18-24, 1999.

[15] Brüninghaus, S. and K.D. Ashley, "Finding factors: learning to classify case opinions under abstract fact categories," *Sixth International Conference on Artificial Intelligence and Law*, Melbourne, Australia, ACM Press, 1997.

[16] Royles, C.A. and T.J.M. Bench-Capon, "Dynamic Tailoring of Law Related Documents to User Needs," *9th International Workshop on Database and Expert System Applications*, IEEE, 1998.

[17] *Proceedings of the 7th International Conference on Artificial Intelligence and Law*. Oslo, Norway, ACM Press, 1999.

[18] *Proceedings of the 8th International Conference on Artificial Intelligence and Law*. St. Louis, Missouri, ACM Press, 2000.

[19] Wang, J., *Distributed Information Organization and Management for Hazardous Waste Regulation Compliance Checking, in Department of Civil and Environmental Engineering*. (under preparation), Stanford University.