

Monitoring and Updating Regulations and Policies for Government Services

Soon Ae Chun¹, Edwin Portscher¹, James Geller²

¹ CIMIC, Rutgers University, Newark, NJ 07102

² Computer Science Department, New Jersey Institute of Technology,
Newark, NJ 07102

Abstract. One of the challenges citizens and businesses face in interacting with governments for entitled services or compliance services is to find the right set of regulations and rules that are applicable for them. Very often the regulations and policies that determine the applicability of specific services are implemented and provided by separate government agencies, thus scattered in different Web sites and documents and are hard to be identified. In addition, the regulatory rules and government policies change often, making it harder to maintain the applicable services. In this paper, we present an integrated *policy ontology* of distributed regulatory rules and policies to support the discovery of applicable regulations and services. We present an ontology-guided annotation and extraction of policy rules from Web source documents. To support the dynamic changes in regulatory and policy rules, we developed a method of *automatic updating* the policy ontology by monitoring the source text documents. We describe a prototype eligibility verification system that verifies eligibility for various welfare government services. The eligibility rules and policies are extracted from different government websites to enrich the policy ontology. The eligibility rules and policies are monitored, and automatically updated, should any changes occur.

1 Introduction

The explosive growth of the Internet has changed the way the world does business. Government agencies have been no exception. Many government agencies have put up websites with tremendous amounts of information to facilitate access to various government services. Citizens or business entities are now faced with information overload and often get lost in cyberspace with hyperlinks pointing everywhere. In addition, the services they are entitled to are multi-agency services composed from individual agency's services, requiring the citizens to interact with multiple agencies independently. For example, welfare applicants need to go through document after document on the Web, searching for the appropriate rules and regulations to figure out eligibilities for several services [1]. There are over a dozen welfare programs that are available in New Jersey, ranging from low-income family food stamps to the well being of women and children. These programs are provided by dozens of divisions within one agency (New Jersey Department of Human Services) [2].

Identifying services and eligibilities is time consuming and often confusing. A system that can provide an integrated and customized view of the needed information and services is highly useful for citizens as well as government officials who are evaluating the eligibilities. To address this issue, we use an integrated ontology of regulatory and business policies from different government agencies. The policies are presented as rules that are arranged with semantic categories of concepts and sub-concepts.

To inform an applicant of their eligibility for government services may seem like a trivial task. One could simply hardcode the eligibility criteria into a program. However, any system like that would require a human to constantly monitor these government websites, and update the code as necessary when one of the eligibility criteria changes. This is costly, creates inconsistent data due to human error and delays processing, limiting the timely availability of current rules and policies for determining eligibilities for services. This also hinders the successful inter-organizational (inter-agency) business integration, where the business policies and rules may change frequently to meet the organizational strategic goals. We address the issue of automated identification of changes and updates of regulatory and business policies. These changes are monitored and immediately available. We use a text-based approach for tracking the policy changes that are reflected in the policy ontology.

This paper thus addresses the following issues and provides our solutions for each: (1) how to model regulatory and business policies is addressed by developing a policy ontology, (2) how to identify a set of eligible services is addressed by reasoning capabilities using the policy ontology, and (3) how to monitor and update the policy rule changes by tracking source text pattern changes tied to the policy ontology.

The paper is organized in the following manner. Section 2 describes the background and motivation of this research. Section 3 presents the system overview and Section 4 introduces the ontology modeling of regulatory and business policies, followed by ontology-driven policy rule extraction from Web sources. In Section 5, we introduce the monitoring and update issues and approach. In Section 6, we present the prototype system architecture and its components, followed by related work and conclusions in sections 7 and 8 respectively.

2 Business Rules, Regulations and Eligibility Rules

Business rules or legal (regulatory) rules are embedded in textual documents in many different Web sites. Thus, a company Web site might describe the rules under which a customer may get a refund for a product. A government Web site might describe what kinds of benefits a person is eligible for under which circumstances. A club or organization might describe rules for a person that wants to become a member. A person would need to read the rules, and apply them mentally to a given situation, e.g., the date when an item was purchased, or the current income of an applicant, and make a determination of eligibility or of the steps to be taken. For example, government welfare service eligibility policies can be seen in sites of the New Jersey Department of Human Services [3], NJ Family Care [4], and NJ Supplemental Nutrition Program for Women, Infants and Children [5]. The citizens will have to visit each site

and decide whether the rules are applicable to determine their eligibility for these programs.

Rule-based systems have long been used to incorporate legal or business rules into software systems, which then may make a determination for a given situation based on input data [16,17]. Such rule-based systems are normally implemented on top of an Expert System Shell such as OPS83, JESS or CLIPS [6,14,15]. However, the process of coding and implementing the rules is as difficult as any other programming task and requires a knowledge engineer's proficiency to understand the English language rules first, before translating them into a system of expert rules. These rule-based systems have one additional complication. Published rules often undergo regular changes. Thus, a maximum salary that is allowed while receiving state government child support may change every year to take inflation into account. Thus, the result of building a rule-based system on the basis of the information in a Web page may become invalidated at any time and without notice to the system builder.

What is needed, then, are policy rules that can be automatically extracted from Web documents to build a knowledge repository that can be used as part of a system. Furthermore, this system should continuously (or at least periodically) monitor the original Web documents with the policy rules and automatically incorporate every change of the English text into the system, without any service interruption. For instance, a welfare eligibility determination system should "read" Web documents and automatically construct policy rules from them, then link those rules into a working eligibility determination system. Thus, a person would enter personal data into the eligibility determination system and be told whether or not he/she can still get a refund for a product or whether he/she can get food stamps. When the rules for food stamps change, the eligibility determination system automatically incorporates the changes.

However, natural language extraction of rules from text documents has not advanced to the point where it can be fully automated and reliably used in sensitive areas such as government operations. Any mistake of such a system could make the government liable to law suits. Thus, we extract rules from Web text in a semi-automated fashion, first by a system that uses semantic and syntactic clues to annotate portions of the text, and then by human who validates the annotated text. The system then parses the annotated text based on linguistic keywords, and builds the linguistic knowledge used in expressing the policies that is then used for subsequent annotations for new documents by the system. The extracted rules are semantically annotated using the ontology's concepts and relations. The automatic monitoring and updating of rules is achieved by linking sections of text to the rules that they gave rise to and monitoring those sections of text for changes.

3 System Architecture

We have developed a system, which brings together various government resources into one place. Our system runs based on a regulatory and business policy ontology, which represents the eligibility criteria from several government services. The system monitors the eligibility criteria on these government Websites and updates the rules if

necessary. A Web form is provided to collect an applicant's information. This information is then run against the rules and the applicant is notified of his/her eligibility status. The system consists of the following components: A Web interface to collect information from the applicant is used as a front end; the ontology, which holds the rules; and a back end, which processes the form against the rules and informs the applicant of her/his eligibility. The policy rule annotation and extraction mechanisms identify policy statements in a text, guided by the concepts in the ontology, and extract the rules and enrich the ontology with policy rules. This component has a human verification part to validate whether the identified text fragments are indeed policy statements and whether the extracted rules are valid. A monitoring component is used to monitor the Web site with the English text of the rules, and to update the ontology.

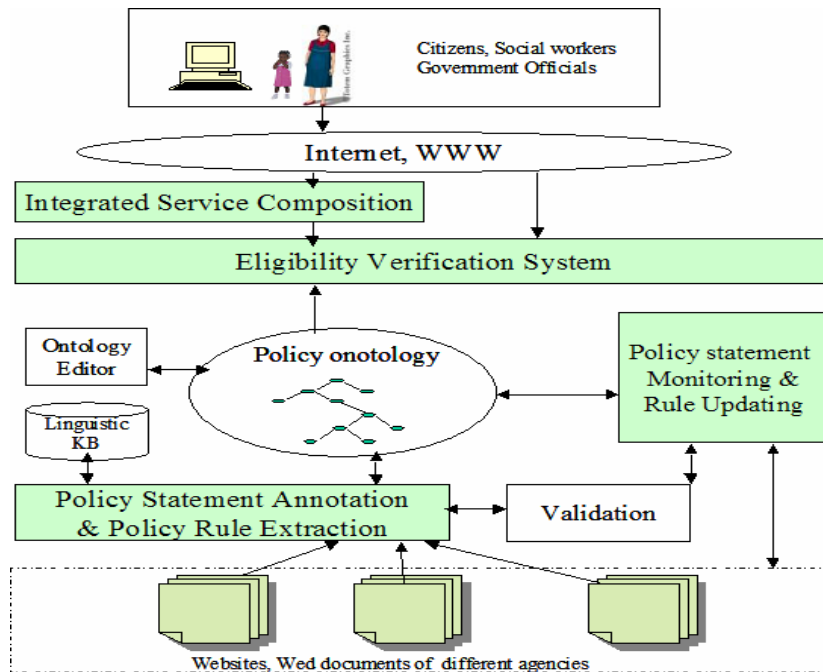


Fig. 1. System Architecture

4 Policy Ontology

The eligibility policy ontology for the New Jersey (NJ) welfare services has been created as shown in Figure 2. The ontology describes the concepts and their relationships from four NJ welfare programs, namely Special Supplemental Nutrition Program for Women, Infants and Children (WIC), Work First New Jersey (WFNJ), NJ FamilyCare and NJ FosterCare. The welfare eligibility policies include rules on family size, income, beneficiary, benefit types, residency, health history, age, citizenship,

drug treatment, year limits, and drug uses. Each welfare program (service) requires a subset of these policy classes (categories), and can specify the rules associated with each class. This ontology was built using Protégé-2000 (Figure 3), where the concepts and their relationships in a welfare service domain are defined and organized.

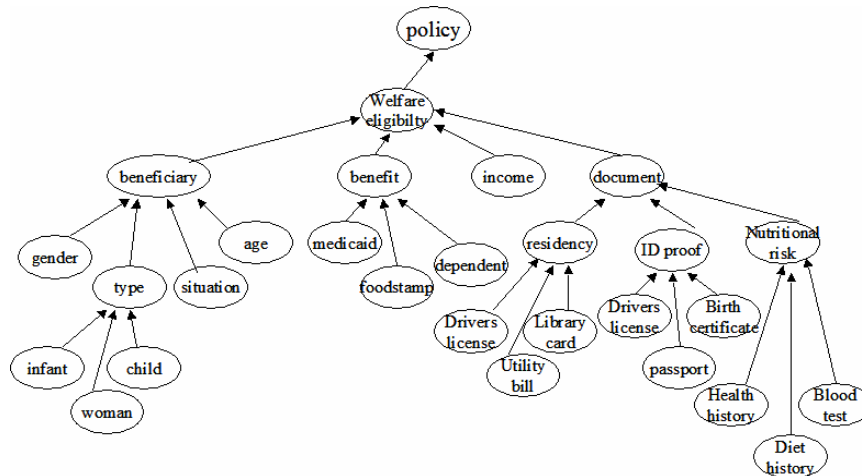


Fig. 2. Welfare Eligibility Policy Ontology

Policy Ontology-driven Rule Extraction: Given the policy ontology of concepts, each policy rule needs to be extracted from text documents. Our approach is to use the concepts defined in the policy ontology to identify and annotate the candidate policy statements in the text. Similar to the Crystal IE system [30], each concept is defined with a concept node (CN) with linguistic and text patterns/cues and semantic constraints. For instance, NJFamilycare has as one of its policy statements that “Foster parents must be at least 21 years old.” This statement is identified as a candidate policy rule related to the concept “age.” We use synonyms of the concept and/or frequently associated phrases with the concept, such as “years old.” The semantics of a concept in the policy ontology and a set of linguistic clues such as deontic verbs, “must, should, ought to, is obliged to, is subject to” that are often used in policy statements are employed to identify a policy statement in a text. Once the fragment of text is considered a policy statement, it is treated as a policy rule candidate statement and is tagged with the related concept. For instance, the above text is annotated as:

<age> Foster parents must be at least 21 years old. </age>

The candidate policy statements are extracted and validated by a human, discarding extracted statements that are not considered policy rules.

Policy Rule Representation: After the validation step, each statement is parsed and converted to a rule formula in JESS [1]. For example, the statement “Foster parents must be at least 21 years old.” is mapped to a JESS rule (defrule rule => (store RESULT (> X 21))). In general, a rule has two parts: the LHS of a rule consists of patterns, which are used to match facts in the knowledge base, while the RHS contains actions and function calls. In this example, the LHS does not contain any

patterns, so the RHS is evaluated immediately. Defrule is a JESS construct, which is used to

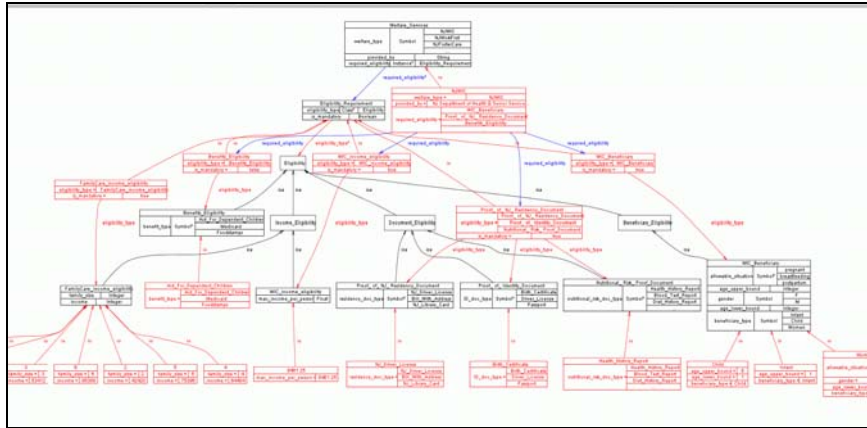


Fig. 3. Protege Implementation of Welfare Eligibility Policy Ontology

define a rule. In the RHS, *RESULT* and *X* are variables. Before the rule is executed *X* will be replaced with a value from the user input. The RHS of the example says that when *X* is greater than 21, it stores the result in variable *RESULT* with the value to be either true or false, depending on whether the condition was satisfied. A JESS rule is similar to an *if... then* statement in a procedural language such as JAVA or C++, but it is not used in a procedural way. While *if... then* statements are evaluated in a specific order, a rule stored in JESS executes the RHS whenever the LHS is satisfied. Rules that are hard-coded in a programming language are very difficult to change. JESS offers a higher degree of flexibility. We have devised an algorithm¹ to recognize the policy statement according to the semantic concepts in the policy ontology and to extract the rules in JESS format.

The policy ontology now can be populated with policy instances corresponding to specific rules extracted from the text. The class defines the topic of each rule attached to it. For the previous example, the class is *age*, because it is a rule having to do with age. The class relates the rule to our form. For example, when creating the form, we have a text box for the user to enter her/his birth date. Birth date relates to age. So we can now calculate the user's age. Each policy class in the ontology can have slots to represent the policy rule information as shown in Figure 4. For each welfare agency we maintain a set of policy rule classes.

- **Policy Statement:** This is the text of the policy rule statement. For example, “Children 15 and younger may apply for the program.”
- **Url:** This is the URL of the Web page where the rule text is located.
- **JessRule** - This is the policy rule in executable JESS format. For example, the rule “Children 15 and younger may apply for the program,” is represented as follows: (defrule rule => (store RESULT (< X 15)))

¹ Not shown in the paper due to the page limitation.

- **Critical Keywords** - This describes critical sections in the rule text that need to be monitored. For example, in the rule “You must be 18 or older to be eligible for this program.” 18 is the critical section. When the age 18 changes, the update of the policy rule is triggered accordingly.
- **Effective period:** The rule can be effective from a start date to an end date.

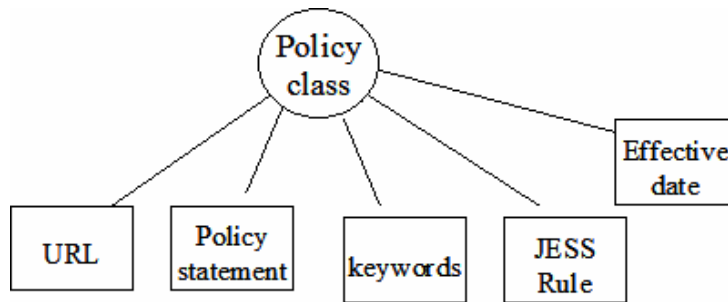


Fig. 4. Policy Rule Representation

5 Rule Monitor and Updater

In order to handle any changes of the text of policy rules, we use a rule monitor that automatically rescans the Web text at regular intervals. Whenever the source text for a policy rule is changed on the Web, this change is automatically propagated to that rule, which is immediately updated. This is done as a hot-swap, so that the eligibility determination system immediately changes its behavior. This is achieved as follows.

Given the rule class and instance in the policy ontology, and the policy’s text statement the system knows exactly which text to monitor. For example, “Children 15 and younger may apply for the program.” From the URL slot of the rule, the system knows which Web page the statement is from. Rule changes are detected when either a critical part of the rule that is being monitored is changed or an entire policy statement text has changed. When changes occur in the critical part of a rule, for example, “21 years old” in “Foster parents must be at least 21 years old,” is changed to 18 years old, then the change is detected. This results in changes in the corresponding JESS rule. In our example, it is changed from (defrule rule => (store RESULT (> X 21))) to (defrule rule => (store RESULT (> X 18))).

If the entire text of a rule is not found, the system checks for the closest match to that sentence. Should our rule change too much, for example if it should change to “Foster parents must be between 18 and 50,” the system needs to determine the semantic category of the new policy and parse the text to generate the new rule. If it is a different semantic category, then it is necessary to create a new rule. If it is the same semantic category, e.g. age, then the system has to revise the existing rule according to the newly parsed rule text. In case of a new rule, human verification is required. The Rule Monitor and Updater detects when a policy rule in a Web page has changed.

Algorithm: Rule Monitoring and Updater

1. FOR (each class C in the ontology)
 - FOR (each rule i at C)
 - Extract the URL C_{U_i} , the text rule C_{T_i} and the JESS rule C_{J_i} ;
 2. Load the page text $T(C_{U_i})$ and remove its HTML tags;
 3. match $T(C_{U_i})$ against C_{T_i}
 4. IF $issubstring(C_{T_i}, T(C_{U_i}))$
 - THEN done; /*Rule has not changed.*/
 5. ELSE FOR (each critical section S_{ij} of C_{T_i})
 - match S_{ij} against $T(C_{U_i})$
 - IF the difference between the old rule and new rule \leq the number of critical sections
 - THEN update the old rule C_{J_i} with new critical sections
 - ELSE report to the user: TOO MANY CHANGES
-

In step 5 we apply a heuristic to decide whether the text has changed to a degree that can be analyzed automatically, or whether human intervention is necessary. We are going from the assumption that if the number of critical sections has stayed the same or has been reduced, we are dealing with parametric changes of the critical sections, such as a change of an age. On the other hand, if new critical sections have been added, the text will need to be reprocessed and presented to a human for approval.

6 Prototype Implementation

The Welfare Eligibility system was built with four simulated government Websites for experimental use in the Welfare project. For instance, the simulated NJFamily-care site covers all the services the real site offers. This includes the eligibility requirements, the services covered, and a detailed description of the applicable policies.

Online application form. One major issue for an eligibility verification system is that it should minimize the information that a user has to enter. A user should be spared entering and reentering her/his personal information over and over again. Rather, he/she should enter all pertinent information once, and the eligibility verification system should route the correct required sections of that information to each government Web site. Our simulated system completely implements this principle. Thus, we maintain a central entry point, the *universal application form* that routes its inputs automatically to the four (simulated) NJ Government Web sites. Only the required sections of the universal application form are passed on to each government Web site.

Web Interface: To determine eligibility for the various government services, welfare applicants fill out one online application form. This online application form includes the questions that will be needed for all four NJ welfare programs that we are supporting. As our system has all the pertinent information, it can make an initial determination whether an applicant is eligible for a welfare service. While this response is not given as a legally binding determination, it is much faster than the gov-

ernment response and can help the applicant by pointing her/him to documents that might have to be obtained for a successful application. When welfare applicants click the SUBMIT button, the Rule Processing Bean will be activated to run the rules and conduct the eligibility verification process.

Fig. 5. Eligibility Verification Web Form

The Java Server Page is responsible for forwarding the form contents from the User Interface to the Rule Processing Bean. It is also responsible for displaying the results from the Rule Processing Bean to the user.

The Rule Processing Bean takes the form input and matches it against the rule types in the ontology. It then runs the rules using the form contents as input, and displays back to the user which eligibility requirements they have met and which eligibility requirements they have not met.

Ontology Construction with Protégé: Protégé is an ontology editor. It is an interface to the ontology and can be used to add new policies for government services and programs. Although the rules in the ontology are updated automatically, the system administrator has to add new eligibility policy concepts when new government services become available.

Policy Rule Extractor: A Java program that converts the Web document into a text format. It applies semantic knowledge (concepts from ontology) and linguistic knowledge (such as key phrases, synonyms, associated or collocated phrases) to extract the candidate policy statements, and converts them into policy rules, which are inserted back into the ontology.

Policy Rule Monitor and Updater: This component monitors the various rules on the various simulated government websites. When a change is detected this component notifies the system administrator by outputting to the screen the time the change occurred, the old rule text, and the new rule text. This component then updates the JessRule and stores it in the ontology accordingly. The administrator can now check

that the rule change was changed correctly, as major changes to the eligibility requirements on the government Web sites would possibly generate an incorrect rule.

The interface to the Rule Monitor and Updater allows the administrator to enter the time intervals when s/he wants the rule monitor to run and check for changes in the Web pages, for example every 5 seconds.

The prototype system has been tested successfully with the simulated Welfare sites' documents and policy rules. Even though our goal is to extract and monitor rules fully automatically, the prototype system can only monitor and update text representations of already identified rules. When the system administrator is notified of a rule change he/she must check to see that the correct rule was generated and improve it if necessary. We are working on algorithms for improved policy annotation and extraction to utilize the linguistic clues gathered from text updates.

7 Related Work

The enormity of the Internet requires effective methods to detect changes. Changes in a data source on the Web require reloading. Due to network limitations these reloads should be limited, however, they should still be performed often enough to ensure freshness of the local copy. To this end, there has been much work done in change detection, such as a sampling based approaches to change detection of networked data sources [18] and frequency-based approaches [19]. In [20] a dynamic queuing model is proposed for search engines to optimize the number of crawling robots in order to handle the updated Web pages and index changes. There are also methods that compare the old document to the new document [27].

With the variety of document types available on the Internet, change detection algorithms for structured and semi-structured documents have been proposed, such as for XML [22, 25]. A format independent change detection algorithm would solve the problem of different document types [25]. There are companies that provide change detection services. For a nominal fee they will monitor websites of one's choice and notify you if they change [23]. Unlike these approaches, we use a continuous (or user-controlled frequency) monitoring approach where the changes are monitored in a flexible time period, and targeted to specific areas in the document, e.g. policy statements.

Approaches for information extraction from Web documents include wrapper induction methods [28, 29], which automatically generate procedures to learn the rules for extraction and use automata to recognize the extractable elements. In [24], a document's structure and layout information (document object model) are used to identify relevant information to be extracted from texts. In contrast, our approach to identifying and extracting policy statements is based on the concepts in the policy ontology.

Attempts have been made to make legal text available on the Web by marking it up with standardized XML tags [31]. In our approach we use concept labels as XML tags. The automated analysis of legal texts for the purpose of summarization has been described in [32]. Legal text analysis and an automated concept extractor, which allows a computer to identify the different concepts that exist in a given legal text, were briefly described in [33].

8 Conclusions and Future Work

In this paper, we have presented an approach to extracting regulatory and business policies from Web documents and to monitoring and updating the changes and modifications in policy rules of different organizations. To identify and extract policy statements from a Web document, a policy ontology that includes concepts and relationships is utilized as well as a linguistic knowledge base. To detect and update the policy rule changes, we monitor policy rule statements in a textual format in a user specified frequency, and automatically extract new rules. To utilize our methods, we have built the service eligibility verification prototype system for various government welfare programs. The policy rules are represented in textual as well as JESS expert system rules so that they can be used to trigger actions in case that the rules satisfy the user input.

We are currently improving our system to handle changes in adding and deleting new and obsolete policies. Our approach for identifying and extracting rules depends on the semantic classes in the policy ontology, thus a new rule that is not in the existing semantic categories is not recognized. We are also looking at the potential uses of Web thesauri and dictionaries for related terms and synonyms. The feedback loop from corrections in the policy statements is being considered. Currently we are looking into the Web source text documents from where the original policy rules are extracted. Often legal changes are posted in separate documents. We will consider extending monitoring of the policy changes by including not only source text documents but also other documents that point to the source text documents. The effective dates for rules should be considered for obsolete versus currently active policies, and the user requests should be logged and dated. For example, if a user turns 21 three months after applying, he/she is now eligible for the Workfirst program and he/she should be informed of that fact.

References

- [1] Njhelps.org, <http://www.njhelps.org/go>, 2002
- [2] New Jersey Department of Human Services, <http://www.state.nj.us/humanservices/index.shtml>
- [3] New Jersey Department of Human Services <http://www.state.nj.us/humanservices/workfirstnj.html>
- [4] NJ Family Care <http://www.njfamilycare.org/>
- [5] NJ Supplemental Nutrition Program for Women, Infants and Children, <http://www.state.nj.us/health/fhs/wic/index.shtml>
- [6] JESS 6.1 Manual. Sandia National Laboratories. 2004 <http://herzberg.ca.sandia.gov/jess/docs/index.shtml>
- [7] Holger Knublauch, An AI tool for the real world: Knowledge modeling with Protégé, <http://www.javaworld.com/javaworld/jw-06-2003/jw-0620-protege.html>, 2003.
- [8] Protégé. Stanford University. <http://protege.stanford.edu/whatis.html>, 2004.
- [9] Soon Ae Chun, Y. Lee and James Geller, Ontological and Pragmatic Knowledge Management for Web Service Composition, Proceedings of DASFAA 2004.
- [10] Holger Knublauch, An AI tool for the real world: Knowledge modeling with Protégé, <http://www.javaworld.com/javaworld/jw-06-2003/jw-0620-protege.html>.

- [11] Yugyung Lee & Chintan, Soon Ae Chun and James Geller, Compositional Knowledge Management for Medical Services on Semantic Web, Proceedings of WWW2004, May 2004, pp. 17–22.
- [12] JAVA API 1.5. Sun Microsystems. 12/1/04
<http://java.sun.com/j2se/1.5.0/docs/api/index.html>
- [13] JAVA Beans. Sun Microsystems. 12/1/04 <http://java.sun.com/products/javabeans/>
- [14] OPS83, 12/1/04 <http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/0.html>
- [15] CLIPS, A Tool for Building Expert Systems, 12/1/04 <http://www.ghg.net/clips/CLIPS.html>
- [16] Vision Corporation, VTECH* BRDS (Business Rule Development System),
<http://www.visionca.com/English/Products/Vtech.htm>
- [17] Grosf, Benjamin, Rouvellou, Isabelle, Degenaro, Lou, Chan, Hoi, Rasmus, Kevin, Ehnebuske, Dave, McKee, Barbara, Combining Different Business Rules Technologies, A Rationalization, Proc. Of the OOPSLA 2000 Workshop on Best-practices in Business Rule Design and Implementation, October 15, 2000.
- [18] Junghoo Cho, Alexandros Ntoulas Effective Change Detection using Sampling,” Proceedings of the 28th VLDB Conference, Hong Kong, China 2002.
- [19] J. Cho and H. Garcia-Molina, Synchronizing a database to Improve Freshness. In Proceedings of 2000 ACM International Conference on Management of Data (SIGMOD) Conference, May 2000
- [20] E. Coffman, Jr., Z. Liu, and R. R. Weber. Optimal robot scheduling for web search engines. *Journal of Scheduling*, 1(1):15–29, June 1998.
- [21] Li Qin, Vijayalakshmi Atluri, An Access Scheduling Tree to Achieve Optimal Freshness in Local Repositories, E-Commerce and Web Technologies: 4th International Conference, EC-Web Prague, Czech Republic, September 2-5, 2003
- [22] Yuan Wang, David J. DeWitt, Jin-Yi Cai, X-Diff: An Effective Change Detection Algorithm for XML Documents, 19th ICDE, 2003.
- [23] <http://www.changedetection.com/>
- [24] Thomas M. Breuel, Information Extraction from HTML Documents by Structural Matching, PARC, Inc., Palo Alto, CA, USA
- [25] Latifur Khan, Lei Wang and Yan Rao, Change Detection of XML Documents Using Signatures, University of Texas at Dallas
- [26] Michael Lanham, Ajay Kang, Joachim Hammer, Abdelsalam Helal, and Joseph Wilson, Format-Independent Change Detection and Propagation in Support of Mobile Computing, University of Florida
- [27] S. Flesca, E. Masciari, Efficient and Effective Web Change Detection, *Data & Knowledge Engineering*, Volume 46, Issue 2, 2003
- [28] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118, Number 1-2, 2000, pp 15-68.
- [29] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner, Towards automatic data extraction from large web sites. In 27th International Conference on Very Large Databases (VLDB2001), 2001.
- [30] S. Soderland, D. Fisher, J. Aseltine, W. Lehnert, Crystal: Inducing a conceptual dictionary, Proceedings of the 14th IJCAI, 1995.
- [31] Biagioli C., Francesconi E., Spinosa P.L., Taddei M., The NIR Project. Standards and tools for legislative drafting and legal document Web publication, Proceedings of the International Conference of Artificial Intelligence and Law, Edinburgh, June 24, 2003.
- [32] Ben Hachey and Claire Grover (2005). Automatic Legal Text Summarisation: Experiments with Summary Structuring. Proceedings of the 10th International Conference on Artificial Intelligence and Law (ICAIL 2005), Bologna, Italy.

[33] Emile de Maat and Tom M. van Engers, *Mission Impossible?: Automated Norm Analysis of Legal Texts*, *Legal Knowledge and Information Systems, Jurix 2003: The Sixteenth Annual Conference*, edited by Daniele Bourcier, 2003.