Exploration of SWRL Rule Bases through Visualization, Paraphrasing, and Categorization of Rules

Saeed Hassanpour, Martin J. O'Connor, and Amar K. Das

Stanford Center for Biomedical Informatics Research,
MSOB X215, 251 Campus Drive, Stanford, California, USA 94305
{saeedhp,martin.oconnor,amar.das}@stanford.edu

Abstract. Rule bases are increasingly being used as repositories of knowledge content on the Semantic Web. As the size and complexity of these rule bases increases, developers and end users need methods of rule abstraction to facilitate rule management. In this paper, we describe a rule abstraction method for Semantic Web Rule Language (SWRL) rules that is based on lexical analysis and a set of heuristics. Our method results in a tree data structure that we exploit in creating techniques to visualize, paraphrase, and categorize SWRL rules. We evaluate our approach by applying it to several biomedical ontologies that contain SWRL rules, and show how the results reveal rule patterns within the rule base. We have implemented our method as a plug-in tool for Protégé-OWL, the most widely used ontology modeling software for the Semantic Web. Our tool can allow users to rapidly explore content and patterns in SWRL rule bases, enabling their acquisition and management.

Keywords: Rule Management, Rule Abstraction, Rule Patterns, Rule Visualization, Rule Paraphrasing, Rule Categorization, Knowledge Representation, OWL, SWRL.

1 Introduction

Rules are increasingly being used to represent knowledge in ontology-based systems on the Semantic Web. As the size of such rule bases increases, users face a perennial problem in understanding and managing the scope and complexity of the specified knowledge. To support rapid exploration of rule bases and meet the scalability goals of the Semantic Web, automated techniques are needed to provide simplified interpretations of rules as well as high-level abstractions of their computational structures. In particular, rule paraphrasing and rule visualization can help non-specialists understand the meaning of logically complex rules. Abstraction of common patterns in rule bases can also enable automatic or semi-automatic categorization of rules into related groups for knowledge management. Such categorized patterns could ultimately form the basis of rule elicitation tools that guide non-specialists entering new rules.

We are addressing the need for such rule management solutions in our development of tools for the Semantic Web Rule Language (SWRL) [35]. In prior work, we developed SWRLTab [37], a plug-in for editing SWRL rule bases within Protégé-OWL [38].

G. Governatori, J. Hall, and A. Paschke (Eds.): RuleML 2009, LNCS 5858, pp. 246–261, 2009.

Protégé-OWL is freely available, open-source knowledge management software that is widely used to specify OWL ontologies for Semantic Web applications. In this paper, we describe a novel approach for exploration of SWRL rule bases through three related techniques: (1) rule visualization, (2) rule paraphrasing, and (3) rule categorization. These three techniques are based on a method of syntactic analysis of SWRL rules. We use the data structure output of this analysis to graphically present the structure of a rule for rule visualization. We use the structural information, along with general heuristics, to paraphrase SWRL rules into simplified, readable English statements. We also apply a pattern recognition algorithm to the structural information to automatically categorize rules into groups that share a common syntactic representation. We show how these techniques can be used to support exploration and analysis of SWRL rule bases, allowing users to more easily comprehend the knowledge they contain. We evaluate the use of our approach by applying these techniques to several biomedical ontologies that contain SWRL rule bases. Finally, we discuss the development of a Protégé-OWL plug-in, called Axiomé that provides these three management techniques for users and developers of SWRL rule bases.

2 Background

OWL [34] is the standard ontology language of the Semantic Web and is rapidly becoming one of the dominant ontology languages in the development of knowledge bases. OWL provides a powerful language for building ontologies that specify high-level descriptions of Web content. These ontologies are created by constructing hierarchies of classes describing concepts in a domain and relating the classes to each other using properties. OWL also provides a powerful set of axioms for precisely defining how to interpret concepts in an ontology and to infer information from these concepts.

The Semantic Web Rule Language (SWRL) [35] is an extension to the OWL language to provide even more expressivity. The SWRL language allows users to write Horn-like rules that can be expressed in terms of OWL concepts and that can reason about OWL individuals. SWRL thus provides deductive reasoning capabilities that can infer new knowledge from an existing OWL ontology. For example, a SWRL rule expressing that a person with a male sibling has a brother can be defined using the concepts of 'person', 'male', 'sibling' and 'brother' in OWL. Intuitively, the concept of person and male can be captured using an OWL class called `Person` with a subclass `Male`; the sibling and brother relationships can be expressed using OWL properties `hasSibling` and `hasBrother`, which are attached to `Person`. The rule in SWRL would be[1]:

```
Person(?x) ^ hasSibling(?x,?y) ^ Male(?y) → hasBrother(?x,?y)
```

Eecuting this rule would have the effect of setting the `hasBrother` property of `x` to `y`. Similarly, a rule that asserts that all persons who own a car should be classified as drivers can be written as follows:

```
Person(?p) ^ hasCar(?p, true) → Driver(?p)
```

[1] The SWRL Submission [35] does not detail a standard syntax for language presentation; the examples shown in this paper reflect the presentation syntax adopted by the Protégé-OWL SWRL Editor.

This rule would be based on an OWL ontology that has the property `hasCar` and the class `Driver`. Executing this rule would have the effect of classifying all car-owner individuals of type `Person` to also be members of the class `Driver`.

One of SWRL's most powerful features is its ability to support user-defined methods or *built-ins* [37]. A number of core built-ins for common mathematical and string operations are defined in the SWRL W3C Submission. For example, the built-in `greaterThan` can be used to determine if one number is greater than another. A sample SWRL rule using this built-in to help classify as adults any person who has an age greater than 17 can then be written as:

```
Person(?p)^ hasAge(?p,?age) ^ swrlb:greaterThan(?age,17) → Adult(?p)
```

When executed, this rule would classify individuals of class `Person` with a `hasAge` property value greater than 17 as members of the class `Adult`.

SWRL rules can also establish relationships between entities in an ontology. For example, the following rule from the California Driver Handbook [36] provides California's driving regulations about minor visitors:

An individual under the age of 18 as a potential driver of a vehicle with a weight of less than 26,000 lbs if they possess an out-of-state driver's license and are visiting the state for less than 10 days.

can be written in SWRL as:

```
Person(?p) ^ has_Driver_License(?p,?d) ^ issued_in_State_of(?d,?s) ^
swrlb:notEqual(?s,"CA") ^ has_Age(?p,?g) ^ swrlb:lessThan(?g,18) ^
number_of_Visiting_Days_in_CA(?p,?x) ^ swrlb:lessThan(?x,10) ^ Car(?c)
^ has_Weight_in_lbs(?c,?w) ^ swrlb:lessThan(?w,26000) →
can_Drive(?p,?c)
```

As mentioned, all classes and properties referred to in this rule must preexist in an OWL ontology.

Table 1. SWRL atom types and example atoms from the Californian Driver Handbook rule

SWRL Atom Type	Example Atom
Class atom	`Person(?p), Car(?c)`

Individual property atom	`has_Driver_License(?p,?d)`
	`issued_in_State_of(?d,?s)`
	`can_Drive(?p,?c)`

Same/Different atom	`sameAs(?x, ?y)` `differentFrom(?x, ?y)`

Datavalued property atom	`has_Age(?p,?g)`
	`number_of_Visiting_Days_in_CA(?p,?x)`
	`has_Weight_in_lbs(?c,?w)`

Built-in atom	`swrlb:notEqual(?s,"CA")` `swrlb:lessThan(?g,18)`
Data range atom	`xsd:double(?x)`

As can be seen from these examples, SWRL rules have a simple Horn-like rule structure. A rule is composed of a body and a head, each of which contain conjunctions of atoms. SWRL does not support disjunction. There are six main types of SWRL atoms defined in the W3C Submission for SWRL. Table 1 lists these atom types and provides example atoms based on the previous rule from the California Driver Handbook.

3 Related Work

Rule management is a very active application area in the business rules domain. These systems are used to define, execute, monitor and maintain the rules used by operational systems [30]. There are a wide variety of commercial rule management tools, which are used to help business organizations standardize and enhance the visibility and consistency of their rule bases. These tools typically provide business user-friendly rule formats, multiple data models for rules implementation, rule testing and refinement, high level rules management interfaces and editors, in addition to other capabilities such as rules versioning, access control, and justification capabilities [31, 32].

Rules are increasingly being used for knowledge management in combination with ontologies [1-3]. As these rule bases grow larger, standard business rule management solutions are being investigated to deal with the resulting complexity. The intimate interactions between rules and the underlying ontology formalisms often require novel solutions [4, 5], however. In particular, the formal underpinnings of the technologies can sometimes be exploited to automatically infer information that may not be possible with the more loosely coupled interactions that are typical between business rules and underlying data.

Some of the traditional approaches used with expert systems can be utilized for certain management tasks. For example, a substantial amount of work has been done in automatic extraction of rules from data [6-11, 33]. Comparatively little work has been done in mining rule bases themselves to assist user comprehension. Rule argumentation techniques [12-15] do typically examine the relationships between rules in a rule base. However, these techniques do not focus on making the rule bases themselves easier to understand. Instead, the goal is to explain the reasoning steps that have been operationalized by the rules. Similarly, descriptive user-friendly text has been used in expert systems to explain the behavior of systems [16, 18], but, again, these textual descriptions have not primarily aimed to explain rule bases and their structure. Other work on rule visualization has mostly focused on showing the connections between rules themselves or and ontology entities or connections between rules and their supporting data, not the structure of the rules bases [19, 20]. UML-based visualization techniques have been used in the business rules domain [43, 44] but these methods are typically designed to provide very detailed views of rule interactions and are not designed for high level rule base exploration.

Principled methods to examine structural patterns in rule bases may significantly aid user comprehension. These approaches can help users to rapidly explore and understand large unfamiliar rule bases. They can also be used to help users understand their own rule bases and spot non-obvious knowledge patterns, which can ultimately help them better structure both the rule bases and the associated ontologies.

4 Methods

In Section 4.1, we discuss a rule abstraction method that parses a SWRL rule and provides as output a tree data structure to represent the rule. We then describe how we use this data structure to visualize, paraphrase and categorize SWRL rules, respectively, in Sections 4.2, 4.3 and 4.4. In Section 4.5, we present a plug-in for Protégé-OWL that supports these three techniques.

4.1 Rule Abstraction

As a first step in our rule management approach, we apply a rule abstraction method to analyze the syntactic structure of a SWRL rule. This method scans the atoms in the body and head of each rule using lexical analysis, reorders the atoms using a set of heuristics, and maintains them in a tree data structure.

Table 2. SWRL atom types and their corresponding rule abstraction priority

SWRL Atom Type	Priority
Class atom	1
Individual property atom	2
Same/Different atom	3
Data-valued property atom	4
Built-in atom	5
Data range atom	6

We give each of the six main types of SWRL atom (shown in Table 2) an ordinal ranking from 1 to 6 that indicates an intuitive sense of the semantic importance of each atom type. Class atoms (e.g., `Person(?p)`) are given the highest priority since they typically refer to the entities of primary interest in a rule. This ranking is followed by object property atoms (e.g., `Can_Drive(?x,?y)`), which capture relationships between these entities. Same as and different from atoms indicate relationships of similarity or difference between entities and are given a lower priority because their use is typically complementary to the use of object property atoms. Data valued atoms (e.g., `has_Age(?x,?y)`) specify the values of properties of particular entities, so are given less priority than inter-entity relationships. Built-in and data range properties operate on these data values so are hence given a lower ordering.

We then reorder the atoms in the head and body of each rule using these priorities. Figure 1 shows the resulting representation for the body of our sample California Driver Handbook rule.

After performing this atom reordering, we build a tree data structure that reflects the information captured by the variable chains in the rule together with the priority information associated with each atom. These trees are generated by a depth-first

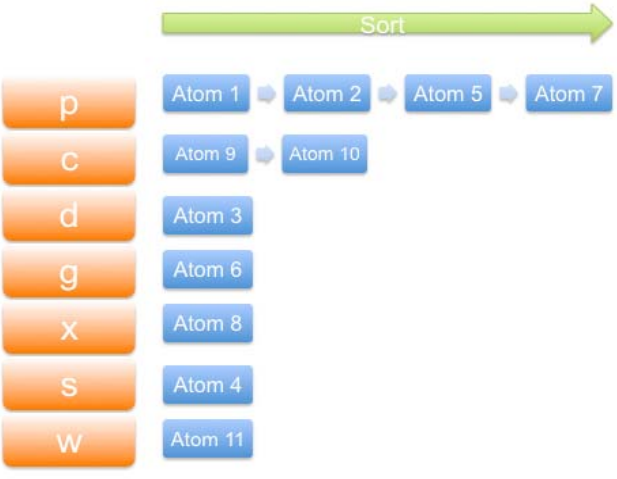

Fig. 1. Example tree data structure that uses a set of priority heuristics to reorder atoms for the sample rule from the California Driver Handbook. The left-hand column (orange boxes) contains variables used in the first position of a SWRL predicate. The atom number (blue boxes) represents the original ordering provided in the SWRL rule.

search of each variable chain in a rule. Once a variable is chosen as a root of a particular tree, atoms that contain that variable as their first argument are created as nodes of the tree at the same level. Any variables that appear as the second arguments of atoms are used to recursively expand the tree to the next level. Loops are avoided by keeping track of atom use.

If several variables share atoms with the same priority we break the tie by giving a higher priority to the variable with a longer list of atoms that start with that variable. If there still is a tie, we use the original ordering of atoms that the rule writer used in creating the rule to determine the first variable to expand. For trees with multiple disconnected roots we chose a new root from atoms not contained in earlier trees and begin the process again. We continue this process until we have scanned all the atoms in the atom list.

4.2 Rule Visualization

Our canonical representation of a SWRL rule can be used to provide a visual representation of the rule. Figure 2 show the visual representation of this data structure for the California Driver Handbook sample rule. This representation allows complex rules that have many classes and properties to be shown as an easily understood nested diagram. In Section 4.5, we show how this graphical representation is used to visualize and browse individual rules in the Axiomé rule management Protégé-OWL plug-in.

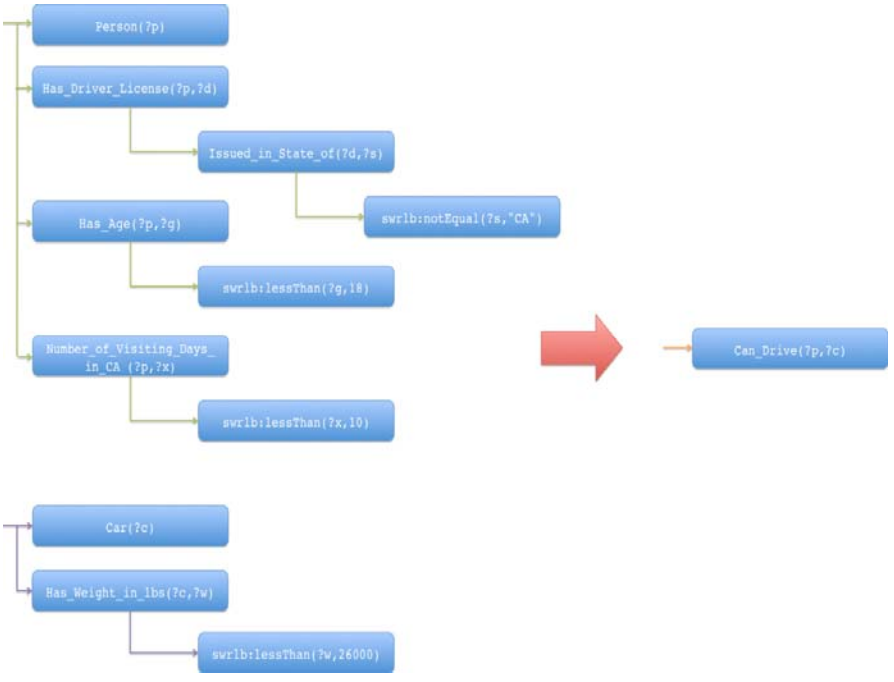

Fig. 2. Data structure showing a reorganization of atoms for the California Driver Handbook sample rule based on the described priority heuristics and variable chains

4.3 Rule Paraphrasing

We also use the tree data structure created by the rule abstraction method to generate paraphrases of SWRL rules, which are more understandable than the syntactic form. We have developed a textual template for each type of atom to generate these paraphrases. The templates use the first atom argument as the subject and the second argument (if any) as the object of sub phrases. An atom's predicate is used in an appropriate form in the template to convey the semantics. In general, we use the name of the underlying OWL classes and properties when generating paraphrases, but we can also support the use of OWL annotations to provide these names. We use heuristics for special cases such as property predicate names or annotation text starting with "has", articles before letters, and predicates beginning with silent 'h'. The ordering of paraphrased atoms is based according to their position in the rule abstraction tree and the paraphrased atoms are connected with appropriate conjunctions. Indentation is used to indicate atom depth. The same margins are used for phrases that are in the same tree level. Each successively deeper level has a larger margin.

Built-in atoms require more elaborate processing because SWRL built-ins can have a variable number of arguments and it is generally not possible to automatically paraphrase the built-in operation by simply using its name. So in the case of built-ins we have defined an annotation ontology that can be used to associated text with a built-in that can be used directly in paraphrases. We have defined annotations for a set of the

standard built-ins defined in the core SWRL built-in ontology [37]. We specially process some standard mathematical operators such as less than and equal and generate condensed paraphrases that omit the mention of some variables to produce more concise text.

For sameAs atoms we make the equality between the variables in the rule explicit when paraphrasing. While we are scanning the atom list to build the tree data structure we note variable pairs that are described to be the same as each other with the sameAs atom. After building the tree we then merge the entries that have been noted to be the same. We also scan the pair list to discover the pairs that are not mentioned explicitly to be the same in the rule but can be inferred to be the same based on the transitivity of the sameAs property. These new discovered pairs are also added to the sameAs pairs list. In paraphrasing the rule we then use only one variable name for the equivalent variables.

Our paraphrasing approach can produce concise and easy-to-read English forms of SWRL rules. The following, for example, is the text that is generated for our sample rule earlier California Driver Handbook (see Section 2):

```
IF
        "p" IS A Person
        AND "p" HAS Driver License "d"
                WHERE "d" Issued in State of "s" WHERE "s" IS NOT EQUAL-
                TO "CA"
        AND "p" HAS Age LESS THAN 18
        AND "p" HAS VALUE "x" FOR Number of Visiting Days in CA WHERE-
        "x" IS LESS THAN 10

AND IF
        "c" IS A Car
        AND "c" HAS Weight in lbs "w" WHERE "w" IS LESS THAN 26000

THEN
        "p" Can Drive "c"
```

Our rule management tool, Axiomé, can generate English paraphrases of rules as a part of the Protégé-OWL plug-in.

4.4 Rule Categorization

We can use the tree data structure to categorize SWRL rules based on the patterns of atoms used. To undertake this rule management technique, we first establish a rule signature for each SWRL rule to capture the structure of the atoms in our abstracted representation. The rule signature is based on a regular expressions language that is composed of an alphabet Σ, and a set of quantifiers Q, such that:

$\Sigma = \{1, 2, 3, 4, 5, 6\}$
$Q = \{-, \hat{}, (\ ), \#, +\}$

Literals in the alphabet Σ represent each of the six main atom types in Table 1. The quantifiers Q are used in the following ways: (1) '-' separates the atoms in the body from the atoms in the head; (2) '^' separates different trees; (3) parenthesis pairs are placed around direct descendants of a node; (4) a '#' is used to expansion of an atom in the data structure and is placed before the next level's atoms; and (5) A '+' is used to show repeated use of the same atoms. Table 3 summarizes the role of each quantifier.

Table 3. Signature quantifiers in rule signature regular expression language

Rule Quantifier	Role
-	Body-Head separator
^	Tree separator
( )	Direct descendents of a node
#	Node expansion
+	Repetition

Consider, for example, rule from family history ontology, which defines a paternal aunt relationship:

```
has_natural_father(?a,?b) ^ has_natural_sister(?b,?c) →
has_paternal_aunt(?a,?c)
```

Each atom in this rule is an individual property atom. Using the rule abstraction method from section 4.1 we can generate a tree structure for the rule, which can be paraphrased as:

```
IF
       "a" HAS Natural Father "b"
               WHERE "b" HAS Natural Sister "c"

THEN
       "a" HAS Paternal Aunt "c"
```

The rule signature is represented:

```
(2#(2))-(2)
```

Using the notations of our regular expressions language, we can define the signature of the example rule California Driver Handbook as:

```
(12#(2#(5))4+#(5)#(5))^(14#(5))-(2)
```

We then use these signatures to group rules into categories. In the Axiomé rule management tool, we support invocation of this categorization technique and graphically show the resulting categories in a tree table.

4.5 Rule Management Tool

We have implemented the three rule-management techniques in a tool called Axiomé, Axiomé is developed as a Protégé-OWL plug-in with functional areas for each of these techniques. These are available as sub-tabs within the plug-in: (1) a Rule Visualization tab to visualize individual rules; (2) a Rule Paraphrasing tab that displays an English-like text explanation for each rule; and (3) a Rule Categorization tab to automatically categorize rules in a rule base. A Rule Browser component is permanently displayed to show a tree-table representation of the SWRL rules in an ontology. This tree-table enables users to explore the rule base and lunch any of three sub-tabs for the rule being explored.

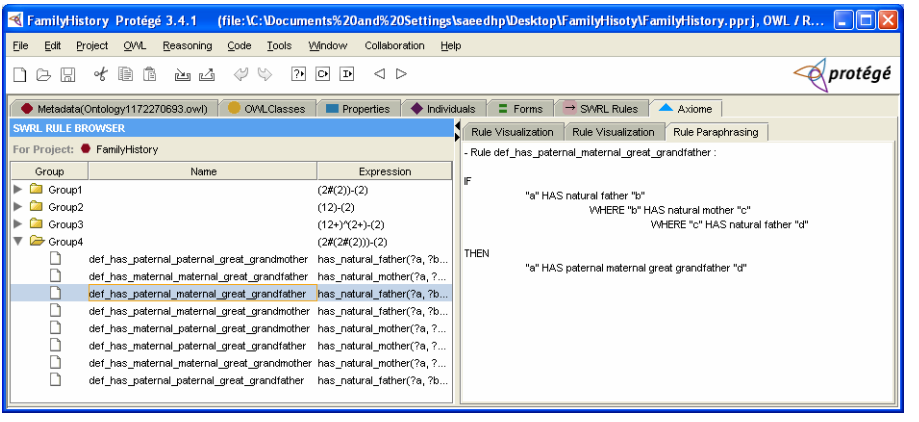

Fig. 3. An screenshot from Axiomé plug-in with three tabs and the SWRL rule browser. Figure shows the paraphrasing tab for one of the rules in the family history ontology.

5 Results

To evaluate the usefulness and efficacy of our visualization, paraphrasing and categorization techniques, we applied our method to four OWL ontologies containing SWRL rules bases. Each of these ontologies was developed as part of a biomedical application and designed by a knowledge engineer or domain expert who was not one of the authors.

The first set of rules that we analyzed is part of an ontology for family medical history [39]. This rule base is composed of 146 rules, which define possible relations between people in a family. We applied our method on the rule base to generate and visualize the data structures and paraphrase them. Our categorization method found four types of rule signatures and thus divides the rule base into four groups. The number of members in each group and their signatures are shown in Table 4. Because the rule base contains general knowledge about family relatedness, we were able to verify the integrity and clarity of these results directly.

Table 4. Rule categorization for family history ontology

Rule signature	**Number of instances in the rule base**	**Examples**
`(2#(2))-(2)`	110	Two link relations, e.g., Uncle and Aunt
`(12)-(2)`	22	One link relations, e.g., Son and Daughter
`(2#(2#(2)))-(2)`	8	Three link relations, e.g., Great Grandfather/Grandmother
`(12+)^(2+)-(2)`	6	Natural/half relations, e.g., Half Brother/Sister
Total Number of rules in the rule base:	*146*	-

Given space limitations, we provide one example in this paper for paraphrasing and signature generation using a representative rule from the family history ontology. Our sample rule defines a 'paternal maternal' great grandfather through the following ancestry:

```
has_natural_father(?a,?b) ^ has_natural_mother(?b,?c) ^
has_natural_father(?c,?d) →
has_paternal_maternal_great_grandfather(?a,?d)
```

The rule is graphically structured by the Axiomé tool as shown in Figure 3.

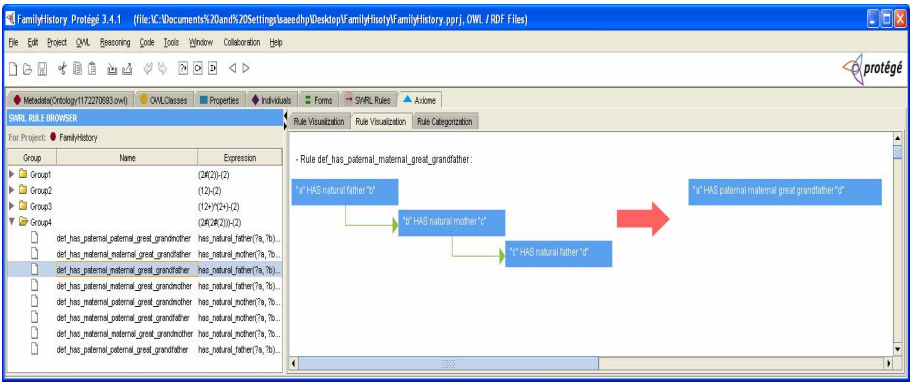

Fig. 4. Axiomé tool as a Protege-OWL plug-in tab, showing a visualization of a simple rule from the family history ontology

The text generated by our paraphrasing method for this rule is:

```
IF
        "a" HAS natural father "b"
                WHERE "b" HAS natural mother "c"
                        WHERE "c" HAS natural father "d"

THEN
        "a" HAS paternal maternal great grandfather "d"
```

And the rule signature is:

```
(2#(2#(2)))-(2)
```

The second rule base we evaluated was developed as part of an ontology of disease phenotypes, or genetically relevant clinical characteristics, for the neurodevelopmental disorder of autism. The ontology and rules will support concept-based querying of the National Database of Autism Research (NDAR), a public resource funded by the National Institutes of Health for archiving, sharing, and analyzing data collected in autism research [40]. NDAR uses the ontology as an information model representing research and clinical data about study subjects and as a domain ontology that defines terms and relationships in autism. The SWRL rules define how each phenotype is to be derived

from a set of clinical findings. The terms, relationships, and abstractions for building the autism ontology are gathered by a literature search of the PubMed database [41].

We applied our categorization technique to the SWRL rules in the autism ontology to find rule signatures. The 14 rules in the current rule base are divided into five groups; one of the groups contains 6 rules with a common structure. The signature and the numbers in each group are shown in Table 5. To check the validity of our results, we asked the developer of the autism ontology to review them. The developer confirmed that our graphical representation and English paraphrases of the rules are semantically equivalent to those in the rule base, and that our categories include the two major types of patterns he used to develop the rule base.

Table 5. Rule categorization for autism ontology

Rule Signature	Number of instances in the rule base
(14+)^(5+)-(12+4+)	6
(14#(5))^(4+)-(12+4+)	3
(14+#(5))^(4)-(12+4+)	2
(14+)^(5)^(5+)-(12+4+)	2
(14+#(5))-(12+4+)	1
Total Number of rules in the rule base:	*14*

The third rule base to which we applied our method was part of a heart disease ontology developed at Stanford Medical School in collaboration with the European Union HEARTFAID project. The resulting THINHeart ontology contains 70 SWRLrules, each of which classifies heart conditions based on presumed cause. A domain expert encoded each of these 70 definitions using a single template. When we applied our categorization technique to the rule base, we found that all 70 rules matched a single rule signature, shown in Table 6.

Table 6. Rule categorization for THINHeart ontology

Rule Signature	Number of instances in the rule base
(14)-(2)	70
Total Number of rules in the rule base:	*70*

We applied our method to a fourth biomedical ontology that contained 63 rules to assess a patient's response to cancer treatment over time [42]. Our categorization method divided the rules into 41 groups; 37 of these groups contain less that 3 rules. Table 7 shows the rule signatures for the groups that had 3 rules or more. We confirmed with the ontology developers that the SWRL rules were intentionally written to fit into a set of distinct rule templates by analyzing and merging rules with the same structures during the authoring process.

Table 7. Rule categorization for cancer response assessment ontology

Rule Signature	Number of instances in the rule base
(12#(2#(5))4#(5))^(12)^(12#(2+4))-(12)	7
(1+2#(2#(5)))^(12)^(12#(2+))^(5)^(5+)-(12+)^(2)	4
(1+2#(2#(5)))^(12)^(12#(2))-(12)	4
(12#(12#(2#(5))))^(12#(2))-(12)	3
8 categories with 2 members	2
29 categories with 1 member	1
Total Number of rules in the rule base:	*63*

6 Discussion

Research on rule representation and rule management has been an active area of work in expert systems, active database systems, association rule mining, and business systems. Rule bases also play an increasingly important role in encoding declarative knowledge within ontology-based systems on the Semantic Web. In this paper, we present work we are undertaking to enable analysis and management of rule bases as part of providing SWRL tool support. We propose a rule abstraction approach that uses a tree data structure to represent a SWRL rule. We have shown that this simple data structure can enable three techniques for visualization, paraphrasing, and categorization of rules. This analytic approach is similar to prior work on static code analysis and formal methods verification [21-25]. Related methods from this field, such as model checking, data-flow analysis and abstract interpretation, could also be applied to perform rule base analysis and integrity checking of the rule base and ontology.

In addition to creating a plug-in tool within Protégé-OWL to make our method available to developers of SWRL rule bases, we applied the method to four existing biomedical ontologies with SWRL rule bases. We have checked the visualization and paraphrasing output of our method for each ontology, and ensured that the outputs accurately represent the rules and have face validity. We are planning a more extensive user evaluation of the Axiomé tool. Our initial application of the method to the four available ontologies was revealing about the nature of the categorizations that were created. In the family history and autism ontologies, the method revealed multiple patterns of rule signatures that we could verify as being valid ourselves or with the developers of the ontology.

The discovery of such 'hidden' rule signatures may enable rule elicitation. The most common rule elicitation methods are performed by knowledge engineers [26, 27]. Another common approach is using domain experts to provide predefined templates and categories for rules or to ask fixed questions to make the rule elicitation easier and semi-automated [28, 29]. This approach may be limited by the skill of the domain expert who provides the templates and the questions. Our rule categorization approach can be used during ontology and rule base development as an alternative method to create the design of templates. As the development of a rule base occurs, common groups of rules may be found based on their signature. The signature can

then be used to create a template, which domain experts can employ to specify new rules. A rule elicitation interface using the rule signatures could also interactively suggest to users how many and what types of atoms a rule might have based on what the user has partially specified for the rule. Such suggestions may speed rule base development and increase the quality of the content.

The other two ontologies that we analyzed in this paper contained rule bases for which the developers had used one or more rule templates, although they were not explicitly represented or constrained by a user interface. We observed that our categorization method accurately identified rule templates. As a result, our categorization method could be used for post hoc analysis of a rule base to ensure that the rules do match known templates. Our rule management approach could be extended to support the design of templates that are based on ontology class restrictions rather than syntactic structure alone. For example, in the case of the cancer response assessment ontology, we can divide the classes and properties of the ontology into several independent sub-ontologies, and categorize the rules based on the sub-ontologies that they cover. Such an approach was of particular interest to the developers of the cancer response assessment ontology who are seeking further approaches to reducing the number of templates needed for rule elicitation by domain experts.

Finally, we believe that creating a simple data structure to represent rules provides an opportunity to perform machine learning on rule bases and to discover frequently occurring or higher level patterns among the rule signatures. We are thus planning to use powerful and sophisticated classifiers such as support vector machines, genetic algorithms and artificial neural networks in our future work.

Acknowledgments. The authors would like to thank Richard Waldinger for his comments on this work and manuscript, and Jane Peace, David Kao, and Mia Levy for sharing their ontologies for our analysis. This research was supported in part by NIH grant 1R01LM009607.

References

1. Maedche, A., Motik, B., Stojanovic, L., Studer, R., Volz, R.: Ontologies for Enterprise Knowledge Management. IEEE Intelligent Systems 18(2), 26–33 (2003)
2. Maedche, A., Staab, S.: Ontology learning for the Semantic Web. IEEE Intelligent systems 16(2) (2001)
3. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology Based Context Modeling and Reasoning using OWL. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, PERCOMW, vol. 18. IEEE Computer Society, Washington (2004)
4. Ostrowski, D.A.: Rule Definition for Managing Ontology Development. Advances in Rule Interchange and Applications, 174–181 (2007)
5. Dou, D., McDermott, D., Qi, P.: Ontology translation on the Semantic Web. Journal on Data Semantics (JoDS) II, 35–57 (2005)
6. Maedche, A., Staab, S.: Discovering conceptual relations from text. In: ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence. IOS Press, Amsterdam (2000)

7. Berendt, B., Hotho, A., Stumme, G.: Towards Semantic Web Mining. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 264–278. Springer, Heidelberg (2002)
8. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Appeared in KDD 1998, New York (1998)
9. Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: Proceedings of the Eighth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada (2002)
10. Hipp, J., Güntzer, U., Nakhaeizadeh, G.: Algorithms for association rule mining — a general survey and comparison. SIGKDD Explor. Newsl. 2, 1 (2000)
11. Bayardo, R.J., Agrawal, R., Gunopulos, D.: Constraint-Based Rule Mining in Large, Dense Databases. Data Min. Knowl. Discov. 4, 2–3 (2000)
12. Rahwan, I., Amgoud, L.: An argumentation based approach for practical reasoning. In: Proceedings of the Fifth international Joint Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan (2006)
13. García, A.J., Simari, G.R.: Defeasible logic programming: an argumentative approach. Theory Pract. Log. Program. 4(2), 95–138 (2004)
14. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in artificial intelligence. Artificial Intelligence 171(10-15), 619 (2007)
15. Chesñevar, C., McGinnis, J., Modgil, S., Rahwan, I., Reed, C., Simari, G., South, M., Vreeswijk, G., Willmott, S.: Towards an argument interchange format. Knowl. Eng. Rev. 21(4), 293–316 (2006)
16. Core, M.G., Lane, H.C., van Lent, M., Gomboc, D., Solomon, S., Rosenberg, M.: Building explainable artificial intelligence systems. In: Proceedings of the 18th Conference on Innovative Applications of Artificial Intelligence (IAAI 2006), Boston, MA (2006)
17. Johnson, W.L.: Agents that explain their own actions. In: Proc. of the Fourth Conference on Computer Generated Forces and Behavioral Representation, Orlando, FL (1994)
18. Van Lent, M., Fisher, W., Mancuso, M.: An explainable artificial intelligence system for small-unit tactical behavior. In: Proceedings of the 16th Conference on Innovative Applications of Artificial Intelligence (IAAI 2004), San Jose, CA, pp. 900–907 (2004)
19. Wong, P.C., Whitney, P., Thomas, J.: Visualizing Association Rules for Text Mining. In: Proceedings of the 1999 IEEE Symposium on information Visualization. INFOVIS, p. 120. IEEE Computer Society, Washington (1999)
20. Blanchard, J., Guillet, F., Briand, H.: Exploratory Visualization for Association Rule Rummaging. In: KDD 2003 Workshop on Multimedia Data Mining (MDM 2003) (2003)
21. Pfleeger, S.L., Hatton, L.: Investigating the Influence of Formal Methods. Computer 30(2), 33–43 (1997)
22. Tilley, T., Cole, R., Becker, P., Eklund, P.: A survey of formal concept analysis support for software engineering activities. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3626, pp. 250–271. Springer, Heidelberg (2005)
23. Blanchet, B., Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., Monniaux, D., Rival, X.: A static analyzer for large safety-critical software. In: Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation. PLDI 2003, San Diego, California, USA, pp. 196–207. ACM, New York (2003)
24. Bush, W.R., Pincus, J.D., Sielaff, D.J.: A static analyzer for finding dynamic programming errors. Softw. Pract. Exper. 30(7), 775–802 (2000)
25. Dean, J., Grove, D., Chambers, C.: Optimization of Object-Oriented Programs Using Static Class Hierarchy Analysis. In: Olthoff, W. (ed.) ECOOP 1995. LNCS, vol. 952, pp. 77–101. Springer, Heidelberg (1995)

26. Leite, J.C., Leonardi, M.C.: Business Rules as Organizational Policies. In: Proceedings of the 9th international Workshop on Software Specification and Design. International Workshop on Software Specifications & Design, p. 68. IEEE Computer Society, Washington (1998)
27. Wright, G., Ayton, P.: Eliciting and modelling expert knowledge. Decis. Support Syst. 3(1), 13–26 (1987)
28. Mechitov, A.I., Moshkovich, H.M., Olson, D.L.: Problems of decision rule elicitation in a classification task. Decis. Support Syst. 12(2), 115–126 (1994)
29. Larichev, A.I., Moshkovich, H.M.: Decision support system "CLASS" for R&D planning. In: Proceedings of the First International Conference on Expert Planning Systems, Brighton, England, pp. 227–232 (1990)
30. Business Rule Management Systems, http://en.wikipedia.org/wiki/BRMS
31. SAP NetWeaver, https://www.sdn.sap.com/irj/sdn/nw-rules-management
32. ILOG, http://www.ilog.com/products/businessrules/
33. Park, S., Lee, J.K.: Rule identification using ontology while acquiring rules from Web pages. Int. J. Hum. Comput. Stud. 65(7), 659–673 (2007)
34. McGuinness, D.L., van Harmelen, F. (eds.): OWL Web Ontology Language Overview. W3C Recommendation (February 10, 2004), http://www.w3.org/TR/2004/REC-owl-features-20040210/
35. SWRL Submission, http://www.w3.org/Submission/SWRL/
36. California Driver Handbook, http://www.dmv.ca.gov/pubs/dl600.pdf
37. O'Connor, M.J., Musen, M.A., Das, A.: Using the Semantic Web Rule Language in the Development of Ontology-Driven Applications. In: Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches, ch. XXII. IGI Publishing (2009)
38. Knublauch, H., Fergerson, R.W., Noy, N.F., Musen, M.A.: The Protégé OWL Plugin: An open development environment for semantic web applications. In: Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, pp. 229–243 (2004)
39. Peace, J., Brennan, P.F.: Instance testing of the family history ontology. In: Proceedings of the American Medical Informatics Association (AMIA) Annual Symposium, Washington, DC, p. 1088 (2008)
40. Young, L., Tu, S.W., Tennakoon, L., Vismer, D., Astakhov, V., Gupta, A., Grethe, J.S., Martone, M.E., Das, A.K., McAuliffe, M.J.: Ontology-Driven Data Integration for Autism Research. In: Proceedings of the 22nd IEEE International Symposium on Computer-Based Medical Systems, IEEE CBMS (2009)
41. Tu, S., Tennakoon, L., O'Connor, M., Shankar, R., Das, A.: Using an integrated ontology and information model for querying and reasoning about phenotypes: the case of autism. In: Proceedings of the American Medical Informatics Association (AMIA) Annual Symposium, Washington, DC, pp. 727–731 (2008)
42. Levy, M.A., Rubin, D.L.: Tool support to enable evaluation of the clinical response to treatment. In: Proceedings of the American Medical Informatics Association (AMIA) Annual Symposium, Washington, DC, pp. 399–403 (2008)
43. Kulakowski, K., Nalepa, G.J.: Using UML state diagrams for visual modeling of business rules. In: International Multiconference on Computer Science and Information Technology, 2008. MCSIT 2008, October 20–22, pp. 189–194 (2008)
44. Lukichev, S.: Visual Modeling and Verbalization of Rules, KnowledgeWeb PhD Symposium (2006)