A Review and Comparison of Rule Languages and Rule-based Inference Engines for the Semantic Web

Thanyalak Rattanasawad, Kanda Runapongsa Saikaew
Department of Computer Engineering,
Faculty of Engineering, Khon Kaen University, Thailand
thanyalak.rattanasawad@gmail.com, krunapon@kku.ac.th

Marut Buranarach, Thepchai Supnithi
Language and Semantic Technology Laboratory,
National Electronics and Computer Technology Center
National Science and Technology Development Agency, Thailand
{marut.buranarach, thepchai.supnithi}@nectec.or.th

Abstract—**With the Semantic Web data standards defined, more applications demand inference engines in providing support for intelligent processing of the Semantic Web data. Rule-based inference engines or rule-based reasoners are used in many domains such as supporting clinical decision support system, e-commerce recommender system, access control mechanism. In order to allow rule reuse and interoperation, several rule languages have been designed for the Web. In this paper, we review and compare some rule languages designed for the Web including FOL-RuleML, SWRL, Notation3, Jena rules and RIF. Some of the comparison criteria include rule expressiveness, syntax, production operations, and built-in functions. In addition, we review and compare some free and public domain rule-based reasoners including Jena inference engine, EYE, OWLIM-lite, BaseVISor and FuXi. Some of the comparison criteria include inference algorithms, supported programming languages, RDFS/OWL reasoning, rule languages and functions. The review and comparison results will provide some guideline for researchers and developers in choosing the rule languages and systems that match the application requirements.**

Keywords: Semantic Web, Rule Languages, Rule-based Inference Engines, Logical Reasoning, Knowledge Representation.

I. INTRODUCTION

With the Semantic Web data standards defined, metadata and ontology information, such as Resource Description Framework (RDF), and Web Ontology Language (OWL) data, are increasingly published on the Web. More applications demand inference engines in providing support for intelligent processing of the Semantic Web data. Rule-based inference engines or rule-based reasoners are used in many domains such as supporting clinical decision support system, e-commerce recommender system, access control mechanism. Functions of rule-based inference engines for the Semantic Web typically include high-performance reasoning algorithms, compatibility with the Semantic Web standards, providing interchangeable syntax and supporting expressive rule languages with built-in functions.

The Semantic Web stack [1] emphasizes the need for rule language for the Web. The rule language can enhance the ontology language by allowing one to describe relations that cannot be described using Description Logic (DL) used in OWL. The rule language also allows sharing and reuse of existing rules on the Web. Rule-based reasoners can benefit from the rule language, which permits data interoperation between different reasoners. Requirements of rule language for the Semantic Web include expressiveness, rule interchange, rule integration, rule language interoperability, and compatibility with other Semantic Web standards.

This paper reviews and compares some rule languages and rule-based reasoners designed for the Semantic Web. One of the main goals is to identify some comparison criteria and key features of different rule languages and rule-based reasoners. We hope that this will provide some guideline for researchers in choosing the languages and reasoners that match the application requirements. Although there are some comparative studies of semantic reasoners [2][3], our study focuses more on features of rule-based reasoners and rule languages designed for the Semantic Web data. This work is the basis for helping users to choose the appropriate rule languages and rule-based inference engines for their applications. This paper is organized as follows. Section II presents comparison criteria and review of some rule languages designed for the Web. Section III presents comparison criteria and review of some free and public domain rule-based inference engines. Section IV presents comparison results of the reviewed rule languages and rule-based reasoners. Finally, section V provides a summary and discusses some future works.

II. RULE LANGUAGES FOR THE SEMANTIC WEB

A. Overview

Rules are representations of knowledge with conditions in some domains of logic, such as first-order logic. A rule is basically defined in form of If-then clauses containing logical functions and operations, and can be expressed in rule languages or formats. Generally, a rule consists of antecedence and consequence containing clauses (or statements), and logical quantifiers used to quantify possible domains of variables. The antecedence contains conditions (premises) combined using logical operators, while the consequence part contains conclusions (or actions). If all the conditions are matched (or fired), the conclusions are operated. A clause is in the form of subject-relation-object, where subject and object can be variables, individuals, literal values or other data structures. The relation can be a function or a predicate. A clause can also be in a form of n-ary function which accepts many arguments. Axioms or facts can also be specified in the document.

For the Semantic Web, the rule language can enhance the ontology language by allowing one to describe relations that cannot be described using DL used in OWL. The rule language also allows sharing and reuse of existing rules between different systems. Some comparison criteria for the rule languages and reviews of some major rule languages are discussed in the following sections.

B. Comparison Criteria

1) Logical Quantifiers

Logical quantifiers specify scopes of possible quantity of variables used in a rule formula. There are two types of quantifiers: universal and existential quantifiers. Universal quantifier specifies that the predicates in the scope can be satisfied by every member in the domain of discourse. Existential quantifier specifies that the predicates can be satisfied by at least one member. Usually, if no quantifier is specified, inference engines treat variables as universally quantified, while blank nodes as existentially quantified [4].

2) Logical Operators

Complex rule typically consists of combination of rule terms and connectives (or logical operators) to connect the rule terms together. Logical operators are conjunction (and), disjunction (or), negation (not), and implication (if-then).

3) Data Types

The RDF data can contain statements with literal values. Literal values can have data types which specify constraints of possible values. Examples of common data types include string, numerical value (integer, float, etc.), date and time, list, and XML Schema (XSD) data types.

4) Built-in Functions

Built-in functions are functions provided for computing or calculating values of variables in rule clauses. The common built-in functions include mathematical functions, boolean functions, and string functions,.

5) Rule-set and rule name

A rule-set is used to group related rules together and can extend scopes of variables in a rule to be reused by other rules in the same rule-set. Rule name is used to specify the name of a rule (or a rule-set) which has advantages in some applications, such as retrieving rules from the database by name, or proof explanation with rule names.

6) Production operations

Production operations specify actions in the conclusions of a rule to be executed by inference engines when the conditions are fired. The actions can be knowledge base modification, e.g., fact assertion, retraction and modification, and other operations such as instance displaying. However, some rule-based systems do not allow fact removing in the knowledge base, which is known as monotonicity.

7) Interchange formats

Interchange formats allow different rule-based systems to interchange rules with other systems. A rule language may support exporting rule to some standard data interchange formats such as XML and JSON.

C. Review of Rule Languages

Rule languages designed for the Semantic Web typically include those introduced by W3C as well as those introduced by different inference engines. Each rule language usually differently supports various logic concepts, and functions. This section reviews five of the most widely-used rule languages designed for the Semantic Web: FOL-RuleML, SWRL, RIF, Notation3 and Jena rule format. Table 1 exemplifies and compares the syntax of the reviewed rule languages in representing an example rule.

1) FOL-RuleML

FOL-RuleML (First-order Logic Rule Markup Language) [5], introduced by W3C in 2004, is a rule language for expressing First-order logic for the web. It is a sublanguage of RuleML. Each of rules in FOL-RuleML consists of a set of statements called ‘atom’. An atom is a form which consists of objects which are individuals or variables, and a relation between the objects. For each rule, scope of quantification of its variables can be optionally specified using 'closure' attribute. The rule language supports conjunction, disjunction and negation operators, and supports universal and existential operator. Besides rule implication (If-Then), equivalence (If and only If) is also supported to represent two-way implication.

FOL-RuleML’s top-level elements are ‘Imp’, ‘Fact’ and ‘Query’. An ‘Imp’ element represents a rule implication which is of the If-Then form. A ‘Fact’ element denotes that its inner contents are axioms or facts. A ‘Query’ element wraps its inner contents as a query. These elements may contain extra information, such as a rule label. The rule language also has performative wrapping elements, e.g., a ‘Consider’ element for wrapping the content without specifying what should be done with the content and an ‘Assert’ element for fact assertion. Serialization of FOL-RuleML is in XML format.

2) SWRL

SWRL (Semantic Web Rule Language) [6] is based on combination of OWL-DL and OWL-Lite, sublanguages of OWL, with Unary/Binary Datalog RuleML, a sublanguage of RuleML) and was introduced by W3C in 2004. SWRL extends the set of OWL axioms to enable rule to be combined with an OWL knowledge base. Syntax of the rule language is relatively like RuleML. They can also interoperate with each other. Logical operators and quantifications supports of SWRL are the same as RuleML's. In addition, RuleML contents can be parts of SWRL content. Axioms may consist of RDF, OWL and rule axioms. A relation can be an IRI, a data range, an OWL property or a built-in relation. An object can be a variable, an individual, a literal value or a blank node. Additionally, the rule language provides many sets of built-in functions, e.g. string functions and mathematical functions.

3) Notation3

Notation3 [7] or N3, submitted by W3C in 2008, is an assertion and logic language which is a superset of RDF. Notation3 extends the RDF model by adding rule formulae, variables, logical implication and functional predicates, as well as providing a more human-readable syntax alternative to RDF/XML syntax. The rule language provides a number of short syntaxes to simplify notation in making statement such as short-hands for path traversal, lists, blank node and common predicates (e.g. is-a predicate). All statements in Notation3 are in the form of subject-predicate-object triple of nodes like RDF, except that a node can also be a formula. A formula, which contains a set of statements, allows itself to be referred to within the language. A common use of formulas is rule implication notation which is basically a combination of a premise formula, a conclusion formula and an implication predicate. Built-in functions are also denoted as predicates. Notation3 is based on 'N3Logic' [4] specifying the logic and operational semantic for the rule language, which is fundamentally monotonic. The language supports both universal and existential quantifiers.

4) Jena Rule

Jena rule [8] is a rule format used only by inference engine in the Jena framework [9]. The rule language syntax is based on RDF(S) and uses the triple representation of RDF descriptions, which is almost like Notation3 except that a rule name can be specified in a rule, no formula notation, and built-in functions are written in function terms. The built-in functions consists of many set of functions including production functions such as instance creation and instance removing, and can also be extended by the user.

5) RIF

RIF (Rule Interchange Format) [10], which became a W3C recommendation in 2013, is a standard developed to facilitate rule-set integration and synthesis. The rule language can be separated into a number of parts, RIF-core [11] which is the common core of all RIF dialects, RIF-BLD [12] (Basic Logic Dialect) comprising basic dialects for writing rules, RIF-PRD [13] (Production Rule Dialect) for expressing production rules and RIF-DTB [14] (Datatypes and Built-in Functions)comprising a set of data types and built-in functions. RIF-BLD and RIF-PRD are extended from RIF-core. The logic which RIF supports is first-order logic, which consists of universal and existential quantifiers, logical conjunction, disjunction, negation and equality operators.

Rules in RIF, similar to other rule languages, can contain both facts and rules, which are a combination of statements. RIF provides additional rule elements and features. For example, 'Group' element is used to group a set of rules together, which also specifies context scope of variables or predicates. A document, represented by a 'Document' element, may contain one or more groups. RIF allows importing other documents to form a multi-document object by using 'import' element. An 'External' element can be used to call externally defined document contents. Shorthand symbols are also provided for writing common predicates.

One important feature of RIF is production rule notation supported in RIF-PRD. In a production rule, conclusions may specify explicit production actions to be operated in the knowledge base. The actions can be knowledge base modification, i.e., fact assertion, retraction or modification , or function execution, e.g. instance displaying. An additional supported feature is frame notation which allows rule notation to be written based on the frame concept.

TABLE I. EXAMPLE SYNTAX OF THE RULE LANGUAGES IN REPRESENTING AN EXAMPLE RULE

Rule Languages	Examples
Example rule	$\forall a \forall b \forall c(hasFather(a,b) \land hasBrother(b,c) \rightarrow hasUncle(a,c))$
FOL-RuleML	<Implies closure='universal' xml:base="#"> <head> <And> <Atom> <Rel>hasFather</ Rel> <Var>a</Var> <Var>b</Var> </Atom> <Atom> <Rel>hasBrother</ Rel> <Var>b</Var> <Var>c</Var> </Atom> </And> </head> <body> <Atom> <Rel>hasUncle</Rel> <Var>a</Var> <Var>c</Var> </Atom> </body> </Implies>
SWRL	<ruleml:imp xml:base="#"> <ruleml:_body> <swrlx:individualPropertyAtom swrlx:property="hasParent"> <ruleml:var>a</ruleml:var> <ruleml:var>b</ruleml:var> </swrlx:individualPropertyAtom> <swrlx:individualPropertyAtom swrlx:property="hasBrother"> <ruleml:var>b</ruleml:var> <ruleml:var>c</ruleml:var> </swrlx:individualPropertyAtom> </ruleml:_body> <ruleml:_head> <swrlx:individualPropertyAtom swrlx:property="hasUncle"> <ruleml:var>a</ruleml:var> <ruleml:var>c</ruleml:var> </swrlx:individualPropertyAtom> </ruleml:_head> </ruleml:imp>
Notation3	@prefix : <#>. @forAll ?a ?b ?c { ?a :hasFather ?b . ?b :hasBrother ?c . } => { ?a :hasUncle ?c . }.
Jena Rule	@prefix : <#>. [RulehasUncle: (?a :hasFather ?b) (?b :hasBrother ?c) -> (?a :hasUncle ?c)]
RIF	Document (Prefix(<#>) Group (ForAll ?a ?b ?c And(:hasFather(?a ?b) :hasBrother(?b ?c)) :- :hasUncle(?a ?c)))

III. Rule-based Inference Engines

A. Overview

Inference engines (or reasoners) are application software for computing or deriving new facts from existing knowledge bases. Although there are various types of inference engines, our study only focus on rule-based inference engines. A rule-based inference engine applies rules with data to reason and derive new facts. When the data match the rules conditions, the inference engine can modify the knowledge base, e.g., fact assertion or retraction, or to execute functions ,e.g., display the derived facts. Our review focuses on rule-based reasoners aimed for the Semantic Web data processing. Functions of these systems typically include high-performance reasoning algorithms, compatibility with the Semantic Web standards, providing interchangeable syntax, supporting expressive rule languages and built-in and user-defined functions. Some comparison criteria for the rule-based reasoners and reviews of some existing reasoner systems are discussed in the following sections.

B. Comparison Criteria

1) Reasoning Strategies and Algorithms

Reasoning strategies are methods used by an inference engine to perform reasoning tasks. There are two main strategies of reasoning, forward chaining and backward chaining [15]. Forward chaining starts from existing facts and applying rules to derive all possible facts, while backward chaining starts with the desired conclusion and perform backward to find supporting facts. Optimized algorithms and techniques may be used to improve the performance of the reasoning process.

2) Expressivity for Reasoning

Inference engines may support different levels or subsets of logics in reasoning. For the Semantic Web data, the common levels of reasoning are typically RDFS, various subsets of OWL, and user-defined rules.

3) Built-in functions and User-defined Functions

Built-in functions are functions provided for computing or calculating values of variables in rule clauses. Some common built-in functions include mathematical functions, boolean functions, and string functions. Some inference engines may provide supports for the user to additionally create custom functions.

4) Reasoning Features

In addition to performing inference and deriving new facts, an inference engine may provide some additional useful function, such as proof explanation, proof tracing etc. An inference engine may also provide a number of configuration options of reasoning, for example, caching, and output filtering.

5) Supported Rule Languages

Rules can be expressed in various languages and formats. Inference engines either support standard rule languages, e.g., RIF, Notation3, or their own rule languages, e.g., Jena rules.

6) Supported Programming Languages and API

User applications typically interact with inference engines via APIs (application programming interfaces). These APIs normally support selecting reasoning strategies, features and operations, storing facts and querying the result data from the knowledge base etc. The forms of APIs include programming languages, and web services, etc.

C. Review of Rule-based Inference Engines

This section reviews five rule-based reasoners aimed for the Semantic Web data processing: Jena Inference Engine, EYE, OWLIM-Lite, BaseVISor and FuXi. Although some systems may contain other functions beyond inference engines, e.g. semantic repository, programming environment, etc., our review of such systems only focuses on its inference engine sub-system. The systems selected for the review only include those with free or public-domain license.

1) Jena Inference Engine

Jena [9] is a Java-based open-source application framework for developing Semantic Web applications. It was adopted by the Apache Software foundation in 2010. Jena provides collections of development tools, for instance, RDF data processing libraries, RDF database systems, 'TDB', and 'SDB' -- a triple store that uses a relational database backend, and its own rule-based inference engine [8]. The framework has a number of predefined reasoners including a transitive reasoner supporting simple taxonomy traversal, an RDFS reasoner, an OWL-lite reasoner and a generic rule reasoner supporting user-defined rules written in the Jena own format. The generic rule reasoner has three execution strategies, forward-chaining, tabled backward-chaining and hybrid. Internally, there are two rule engines, a forward chaining RETE [16] engine and a tabled datalog engine. Built-in functions, e.g., string and mathematical functions are also provided and can be extended by the user. Additional reasoning functions include proof tracing, proof explanation, etc.

2) EYE

EYE (Euler YAP Engine) [17] is a high-performance inference engine which supports clients in various languages e.g. Java, C#, Python, JavaScript and Prolog. The inference engine was developed by Jos De Roo, a researcher at AGFA healthcare. EYE uses backward-forward-backward chaining via an underlying Prolog backward chaining and Euler path detection to avoid loops in an inference graph. Internally, EYE translates the supported rule language, Notation3, to Prolog Coherent Logic intermediate code and runs it on YAP (Yet Another Prolog) engine, a high-performance Prolog compiler for demand-driven indexing. The inference engine supports monotonic abduction-deduction-induction reasoning cycle. EYE can be configured with many options of reasoning, such as not proving false model, output filtering, and can also provide useful information of reasoning, for example, proof explanation, debugging logs, and warning logs. The inference engine can be added new features by using user-defined plugins.

3) OWLIM-lite

OWLIM,-lite [18], is a free version of OWLIM, a semantic database management system including a native RDF rule entailment engine. It was implemented as a part of Sesame

[19], a Semantic Web framework developed by Ontotext AD via the SAIL (Storage and Inference Layer) interface. OWLIM-lite is based on Triple Reasoning and Rule Entailment Engine (TRREE) – a native RDF rule-entailment engine [20]. The supported reasoner can be configured through the definition of rule-sets e.g. OWL-Horst, RDFS, user-defined rule-set, etc. OWLIM-lite performs reasoning based on forward-chaining in main memory, and uses the configured rule-set to compute all inferred statements at load-time.

4) BaseVISor

BaseVISor [21], developed by VIStology, is a forward chaining inference engine. The inference engine is based on RETE network optimized for RDF processing and supports OWL RL and R-entailment [22]. It uses its own rule language for reasoning and provides support for RuleML by automatically translating it into the own rule format. Rules can be specified priority for ordering in execution. The inference engine supports production functions (e.g. fact assertion and retraction) and built-in functions (e.g. common mathematical functions, date and time functions) which can be extended by the user.

5) FuXi

FuXi [23] is a bi-directional logical reasoning system for the Semantic Web developed in Python. It is a forward-chaining production system for Notation3 implemented as a companion to RDFLib, a Python library for RDF processing. The inference engine relies on the RETE algorithm for pattern or object match problem. FuXi currently implements production capabilities for a limited subset of Notation3 and support negation, addition and removal of rule in runtime and statement removal. Backward chaining reasoning is also supported using SPARQL endpoints. In addition to implementing the compilation of a RETE network from Notation3 document, FuXi can also export the complied network into a diagram which can be applied to graph algorithms, e.g., breadth-first search, heuristic algorithms, etc., to optimize reasoning.

IV. Comparison of Features

A. Comparison of the Rule Languages

In this section, we compare features of the reviewed rule languages, i.e. FOL-RuleML, SWRL, Notation3, Jena Rule and RIF, based on various criteria. Table II presents comparison of features related to expressiveness of the language, i.e. whether it requires explicit logical operators, explicit logical quantifiers and supports data type. Table III presents comparison of features related to supported operations and syntax of the language, i.e. whether it supports different types of production operation, rule naming, rule set and data interchange format. Table IV presents comparison of features related to different types of built-in functions provided.

B. Comparison of the Rule-based Inference Engines

In this section, we compare features of the reviewed rule-based reasoners based on various criteria. The comparison were based on the following software versions: Jena 2.10.1, EYE 2013-05, OWLIM-lite 5.3, BaseVisor 2.0 and Fuxi 1.4. Table V presents comparison of features related to reasoning strategies and algorithms of the system, i.e. whether it supports forward or backward reasoning strategies, and what reasoning algorithms are mainly used. Table VI presents comparison of features related to supported RDFS/OWL reasoning, rule languages and programming languages. Table VII presents comparison of features related to reasoning operations and features provided, i.e. supported production operations, built-in and user-defined function and proof explanation facility.

TABLE II. Comparison of Expressiveness of the Rule Languages

Rule Languages	Explicit Logical Operators			Explicit Logical Quantifiers		Data-types Support
	Conj.	*Disj.*	*Neg.*	*Univ.*	*Exist.*	
FOL-RuleML	✓	✓	✓	✓	✓	✓
SWRL	✓	✓	✓	✓	✓	✓
Notation3	-	-	-	✓	✓	✓
Jena Rule	-	-	-	-	-	✓
RIF	✓	✓	✓	✓	✓	✓

'Conj.'=Conjunction, 'Disj.'=Disjunction, 'Neg.'=Negation, 'Univ.'=Universal, 'Exist.'=Existential

TABLE III. Comparison of Supported Operations and Syntax of the Rule Languages

Rule Languages	Rule -set	Rule Name	Production Operations			Inter change-format
			Assert.	*Retrc.*	*Mod.*	
FOL-RuleML	-	✓	✓	✓	-	XML
SWRL	-	✓	✓	✓	-	XML
Notation3	-	-	✓	-	-	-
Jena Rule	✓	✓	✓	✓	-	-
RIF	✓	✓	✓	✓	✓	XML

'Assert.'=Assertion, 'Retrc.'=Retraction, 'Mod.'=Modification

TABLE IV. Comparison of Built-in Functions of the Rule Languages

Rule Languages	Built-in Functions					
	Math	*String*	*Logic*	*Time*	*List*	*URI*
FOL-RuleML	✓	✓	✓	✓	✓	✓
SWRL	✓	✓	✓	✓	✓	✓
Notation3	✓	✓	✓	✓	✓	✓
Jena Rule	✓	✓	-	✓	✓	-
RIF	✓	✓	✓	✓	✓	✓

TABLE V. Comparison of Reasoning Strategies and Algorithms of the Inference Engines

Inference Engines	Reasoning Strategies		Reasoning Algorithms
	Forward Chaining	*Backward Chaining*	
Jena	✓	✓	RETE engine
EYE	✓	✓	Euler Path Detection, Prolog Demand-driven Indexing
OWLIM	✓	-	SAIL, TRREE
BaseVISor	✓	-	RETE engine
FuXi	✓	✓	RETE-UL engine

TABLE VI. COMPARISON OF SUPPORTED RDFS/OWL REASONING, RULE LANGUAGES AND PROGRAMMING LANGUAGES OF THE INFERENCE ENGINES

Inference Engines	**RDFS/OWL Reasoning**	**Rule Languages**	**Programming Languages / API**
Jena	RDFS, OWL-lite	Own format	Java
EYE	selected predicates of RDFS/ OWL	Notation3	Java, C#, Python, JavaScript, Prolog, Web Service
OWLIM-lite	RDFS, OWL-Horst, OWL-Max, OWL2-QL	Own format	Java, Web Service
BaseVISor	OWL2-RL, R-entailment	Own format, RuleML	Java, Web Service
FuXi	OWL2-RL, RIF-core	Notation3, RIF-BLD	Python, Web Service

TABLE VII. COMPARISON OF REASONING OPERATIONS AND FEATURES

Inference Engines	**Production Operations**		**Built-in Function**	**User-defined Function**	**Proof Explan -ation**
	Assert.	*Retract.*			
Jena	✓	✓	✓	✓	✓
EYE	✓	-	✓	✓	✓
OWLIM-lite	✓	-	✓	-	-
BaseVISor	✓	✓	✓	✓	-
FuXi	✓	✓	✓	✓	✓

’Assert.’=assertion, ‘Retrac.’=retraction

V. CONCLUSIONS

In this paper, we reviewed and compared some major rule languages and rule-based reasoners designed for the Semantic Web data. One of the main purposes is to identify some criteria that can be used to compare the rule languages and rule-based reasoners. In addition, we compared some key features of some existing rule languages and rule-based reasoners based on the given criteria. We believe that this work can help to provide some guideline for researchers and developers in choosing the rule languages and reasoners to match the application requirements.

Our planned future works include development of an integration framework to promote interoperability between different rule-based systems. In addition, we plan to apply the framework to evaluate performance of different rule-based reasoners.

ACKNOWLEDGMENT

The financial support from Young Scientist and Technologist Programme, NSTDA (YSTP: SP-56-NT03) is gratefully acknowledged.

REFERENCES

[1] W3C, “Semantic Web,” 2013. [Online]. Available: http://www.w3.org/standards/semanticweb.

[2] S. Singh and R. Karwayun, “A Comparative Study of Inference Engines,” in *International Conference on Information Technology*, 2010.

[3] “Semantic Reasoner.” [Online]. Available: http://en.wikipedia.org/wiki/Semantic_reasoner.

[4] T. Berners-Lee, “Notation3 Logic,” 2005. [Online]. Available: http://www.w3.org/DesignIssues/N3Logic.

[5] H. Boley, M. Dean, B. Grosof, M. Sintek, B. Spencer, S. Tabet, and G. Wagner, “FOL RuleML: The First-Order Logic Web Language,” 2005. [Online]. Available: http://www.w3.org/Submission/FOL-RuleML/.

[6] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, “SWRL: A Semantic Web Rule Language Combining OWL and RuleML,” 2005. [Online]. Available: http://www.w3.org/Submission/SWRL/.

[7] T. Berners-Lee and D. Connolly, “Notation3 (N3): A readable RDF syntax,” 2011. [Online]. Available: http://www.w3.org/TeamSubmission/n3/.

[8] The Apache Software Foundation, “Apache Jena - Reasoners and rule engines Jena inference support.” [Online]. Available: http://jena.apache.org/documentation/inference.

[9] The Apache Software Foundation, “Apache Jena,” 2013. [Online]. Available: http://jena.apache.org/.

[10] M. Kifer and H. Boley, “RIF Overview (Second Edition),” 2013. [Online]. Available: http://www.w3.org/TR/rif-overview/.

[11] H. Boley, G. Hallmark, M. Kifer, A. Paschke, A. Polleres, and D. Reynolds, “RIF Core Dialect (Second Edition),” 2013. [Online]. Available: http://www.w3.org/TR/rif-core/.

[12] H. Boley and M. Kifer, “RIF Basic Logic Dialect (Second Edition),” 2013. [Online]. Available: http://www.w3.org/TR/rif-bld/.

[13] C. de Sainte Marie, G. Hallmark, and A. Paschke, “RIF Production Rule Dialect (Second Edition),” 2013. [Online]. Available: http://www.w3.org/TR/rif-prd/.

[14] A. Polleres, H. Boley, and M. Kifer, “RIF Datatypes and Built-Ins 1.0 (Second Edition),” 2013. [Online]. Available: http://www.w3.org/TR/rif-dtb/.

[15] J. C. Giarratano and G. D. Riley, *Expert Systems: Principles and Programming*, 4th ed. Course Technology, 2004.

[16] C. L. Forgy, “Rete: A Fast Algorithm for the Many PatternIMany Object Pattern Match Problem,” *Artificial Intelligence*, vol. 19, pp. 17 – 37, 1982.

[17] J. De Roo, “Euler Yet Another Proof Engine.” [Online]. Available: http://eulersharp.sourceforge.net/.

[18] Ontotext AD, “OWLIM-lite Reasoner.” [Online]. Available: http://owlim.ontotext.com/display/OWLIMv53/OWLIM-Lite+Reasoner.

[19] Aduna, “openRDF.org: Home.” [Online]. Available: http://www.openrdf.org/.

[20] Ontotext AD, “Semantic Repository.” [Online]. Available: http://www.ontotext.com/semantic-repository.

[21] VIStology, “BaseVISor by VIStology, Inc.,” 2013. [Online]. Available: http://www.vistology.com/basevisor/basevisor.html.

[22] C. J. Matheus, K. Baclawski, and M. M. Kokar, “BaseVISor: A Triples-Based Inference Engine Outfitted to Process RuleML and R-Entailment Rules.” .

[23] “FuXi 1.4: A Python-based, bi-directional logical reasoning system for the semantic web.” [Online]. Available: https://code.google.com/p/fuxi/wiki/FuXi.