Challenges for Rule Systems on the Web

Yuh-Jong Hu[1] and Ching-Long Yeh[2]

[1] Emerging Network Technology (ENT) Lab.
Department of Computer Science
National Chengchi University, Taipei, Taiwan
hu AT cs.nccu.edu.tw
[2] Department of Computer Science Engineering
Tatung University, Taipei, Taiwan
chingyeh AT cse.ttu.edu.tw

Abstract. The RuleML Challenge started in 2007 with the objective of inspiring the issues of implementation for management, integration, interoperation and interchange of rules in an open distributed environment, such as the Web. Rules are usually classified as three types: deductive rules, normative rules, and reactive rules. The reactive rules are further classified as ECA rules and production rules. The study of combination rule and ontology is traced back to an earlier active rule system for relational and object-oriented (OO) databases. Recently, this issue has become one of the most important research problems in the Semantic Web. Once we consider a computer executable policy as a declarative set of rules and ontologies that guides the behavior of entities within a system, we have a flexible way to implement real world policies without rewriting the computer code, as we did before. Fortunately, we have de facto rule markup languages, such as RuleML or RIF to achieve the portability and interchange of rules for different rule systems. Otherwise, executing real-life rule-based applications on the Web is almost impossible. Several commercial or open source rule engines are available for the rule-based applications. However, we still need a standard rule language and benchmark for not only to compare the rule systems but also to measure the progress in the field. Finally, a number of real-life rule-based use cases will be investigated to demonstrate the applicability of current rule systems on the Web.

1 Introduction

The RuleML Challenge competitions started in 2007[3], so the RuleML-2009 Challenge will be the third year for the rule system competition. We offer participants the chance to demonstrate their commercial and open source tools, use cases, and applications for rule related technologies. For the past two RuleML Challenge competitions, only a minimum set of requirements was given for evaluating the submitted demo systems. The criteria were that declarative rules should have to play a central role in the application, and that the demo systems should

[3] RuleML-2007 Challenge, http://2007.ruleml.org/index-Dateien/Page787.htm

preferably be embedded into a Web-based or distributed environment, etc. The Challenge winners were selected and 1st and 2nd places were awarded with prestigious prizes.

The RuleML-2009 Challenge follows similar processes and the evaluation criteria are the same as in the previous two events. But we consider inviting more participants to submit their rule related systems in this year. In the RuleML-2009 Challenge, we organize events as two tracks, one is by invitation, to demonstrate a commercial or open source environment for its rule systems, and the other is open to general public for a real system competition. In addition to the demo systems with reports submitted to the RuleML Challenge website[4], it is also possible to submit demo papers describing research and technical details, and the selected papers will be published in additional special Challenge proceedings, such as CEURS. A final selection of revised papers from the Challenge proceedings will be resubmitted to a special issue of a journal for publishing. In this RuleML Challenge survey paper, we point out the possible research and implementation challenges for rule systems on the Web that are related to the Challenge competition events in the forthcoming years.

1.1 Challenges for Rule Systems

Rules as human understandable policies are everywhere in our daily life to impose human behaviors. For example, before you take a flight, you need to read airline check-in and boarding time rules in the policy statement of your booking itinerary receipt. If you violate any rule you might miss your plane. Related situations in this scenario of using rules are early-bird conference registration, special discount hotel reservation, payment and refund policies, etc. These rules as policies are represented as human understandable natural language. However, we still need to transform these natural language policies into computer programming rules for computer system understanding and automatic execution. Sometimes, not all of the rules imposed on a human are necessarily and possibly represented as software programs to accomplish automatic execution in our computer systems. Usually, these rules restrict only human behavior, without direct connection with any software system. For example, we have law for privacy protection and digital rights management but not all of privacy rights and digital rights for human are required to be represented and evaluated in computer systems.

There are several challenges while implementing rule systems on the Web. Rules should be allowed to cope with the data model, such as RDB/OO-DB, or a knowledge base, such as an ontology, to permit query and modification services on the data models. Policies imposed on human behavior are declared in some policy language by the combination of rules and an ontology (or database), and these policies can be automatically interpreted and executed by a computer. There should be a standard language and framework for rule systems to enable rule interchange services on the Web. A certain number of use cases are easily

[4] http://ruleml-challenge.cs.nccu.edu.tw

represented and executed by rule and ontology reasoning engines with rule interchange and ontology merging standards to ensure rule interoperability and ontology compatibility.

In the early computer development stage, imperative programming languages such as C and Java were used to represent rules and execute them on a computer system. But these rules are inflexible and not easy to maintain when they are distributed on the Web and require interchange and integration between rule systems. Moreover, imperative programming languages are not appropriate to express concepts of human policies as computerized rules. Recently, people use declarative programming to specify the rules and execute them automatically, where XML is used as a standard syntax representation for interchange of declarative rules, such as RuleML [**?**], RIF [**?**], etc.

Even though an XML-based standard rule language and framework provides rule interchange service, pure XML cannot specify a well-defined semantics for rules. So people in the standard rule community constructed a logic foundation behind rule languages and their framework, to preserve the integrity of syntax and semantics of rules interchange for various rule systems. Similarly, OMG SBVR intends to define the vocabulary and rules for documenting semantics of business vocabulary, facts, and rules, as well as an XMI schema for interchange of business vocabularies and rules among organizations and between software tools[5]

In this paper, we first introduce the classification of rules, then, in section 2, we address the issue of rules, and databases and ontologies . In section 3, the current status of a declarative policy as the combination of ontology and rules will be introduced. In addition, Semantic Web Service (SWS) processes also require a declarative policy to express and execute Web Service rules to control information sharing and service execution. In section 4, we examine current different rule management systems and engines. In section 5, we investigate different rule interchange languages. In section 6, we look into the use cases that are possibly represented and executed by the rule systems. Finally, we conclude this study in section 7.

2 Rule and Data Model

2.1 The Classification of Rules

Rules are classified as three types: deductive rules (or derivation rules), normative rules (or integrity rules), and reactive rules (or active rules). One can use deductive rules and facts to trigger a forward or backward reasoning engine to derive implicit facts. Normative rules pose constraints on the data or on the business logic to ensure their consistency in the database or knowledge base. Without reactive rules, we cannot update a database or knowledge base by using deductive rules only.

[5] http://www.omg.org/spec/SBVR/1.0/

Reactive rules are further subdivided into event-condition-action (ECA) rules and production rules. ECA rules are rules of the form *ON Event IF Condition DO Action*, where *Action* should be executed if the *Event* occurs, provided that the *Condition* holds. Production rules are rules of the form *IF Condition DO Action*, where *Condition* queries the working memory containing the data on which the rules operate. *Action* should be executed whenever a change to the underlying database makes *Condition* true [**?**].

In reactive rules, we verify the satisfaction of conditions and also execute the action whenever message arrival or timer event triggers the rule. Declarative rules extend their executive power by the combination of rule semantics and imperative programming in the action part.

2.2 Rules and Databases

As early as 1980, Ullman pointed out the principles of the integration of database and knowledge base systems [**?**] [**?**]. The foundation of database is relation algebra with SQL as a declarative database query language. However, first order logic (FOL) was also proposed as a way to represent "knowledge" and as a language for expressing operations on relations. The roots of relational theory is logic, and so we cannot deny that the foundation of relational DBMS is based on logic [**?**]. The simplest data model of FOL is "Datalog", which was coined to suggest a version of Prolog suitable for database systems where it does not allow function symbols in Datalog's predicate arguments. In the IDEA methodology, deductive rules and reactive rules were built on top of the object-oriented (OO) database as a way to express operations on the OO data model [**?**].

2.3 Rules and Ontologies

Concepts of the Semantic Web have been proposed by Tim Berners-Lee et al. since 2001 [**?**]. Graph-based RDF(S), including RDF and RDF-schema were the first standardized ontology languages to represent an ontology's schema and instances. Then, standardized ontology languages based on Description Logic (DL) [**?**], i.e., OWL-DL (later OWL 2), enhanced RDF(S) that plays the major role of knowledge representation for the Semantic Web. However, the logic program (LP) rule language was also introduced because of the limited expressive power of a DL-based ontology language in some situations, such as property chaining, and the manipulation of events, states, and actions.

Initially, the "rule" layer was laid on top of the "ontology" layer in the Semantic Web layered architecture but it has undergone several revisions reflecting the evolution of layers [6]. The most recent layered architecture of rule and ontology layers is one where they sit side by side to reflect their equal status but with some basic assumption differences between ontology and rule, such as the open world assumption (OWA) vs. the closed world assumption (CWA), or the non-unique name assumption (non-UNA) vs. the unique name assumption (UNA) [**?**].

[6] http://www.w3.org/2007/03/layerCake.svg

It will be a challenge to resolve these basic assumption differences when we combine rule and ontology to execute rule systems on the Web.

Rules and RDF(S) Inspired from F-Logic, TRIPLE[7] was one of the earliest rule languages using Horn rules to access the RDF datasets. Another rule language called Notation3 (N3) uses a CWA forward reasoning engine to access the ontologies generated from RDF(S) [8]. SPARQL is another W3C standardized query language for querying RDF datasets. SPARQL queries are represented as Datalog rules so SPARQL's `CONSTRUCT` queries are viewed as deductive rules, which create new RDF triples from the RDF datasets.

Rules and OWL In addition to the Semantic Web Rule Language (SWRL) [**?**], Rule Interchange Format (RIF) is an emerging rule interchange language from W3C RIF WG [**?**]. It intends to provide core and extend languages with a common exchange syntax for all of the classification rule languages, i.e., deductive, normative, and reactive rules. The requirements of integrating different types of rules with possible data (and meta data) accessing representation, i.e., RDB, XML, RDF, and OWL, drive the development of a RIF core interchange format, the*RIF Core,* and its extensions, *RIF dialects*. Another recent development is to combine RIF and OWL 2 in RIF, RDF, and OWL that specifies the interactions between RIF, RDF and OWL for their compatibilities [9].

2.4 Combination of Rule and Ontology

A one-way knowledge flow exists from an ontology module to a rule module for knowledge acquisition, where an ontology module's instances are imported as basic facts and filtered with conditions in the rules. This passive knowledge query only uses deductive rules. If a rule engine derives implicit new facts not in an ontology module and furthermore updates new facts back to an ontology module, then it provides another reverse knowledge flow from a rule module to an ontology module. In this two-way knowledge flow process, normative and reactive rules are also required to check the knowledge consistency and trigger the message passing for updating the ontology's knowledge base.

The idea of combining rules and ontologies is to fulfill a goal of two-way knowledge flow. The combination is classified as two types: tightly coupled integration and loosely coupled integration [**?**]. In the tightly coupled integration model, all of the terms in the rule's body and head are specified in the ontology schema, but in the loosely coupled integration model we do not have this requirement. So, some rules have their own defined terms in the rules' body or head. This loosely coupled integration model enhances the expressive power of ontology and rule as compared to the tightly coupled one.

[7] http://triple.semanticweb.org/
[8] http://www.w3.org/2000/10/swap/doc/cwm
[9] http://www.w3.org/2005/rules/wiki/OWLRL

Description Logic Program (DLP) [?] and SWRL are two well-known tightly coupled integration models. In general, both DL and LP are subsets of FOL in knowledge representation but each has its own part that cannot be expressed in the other part. DLP only takes intersection of DL and LP so knowledge representation in this model is limited. In SWRL, the major knowledge representation is OWL-DL with additional Datalog rules from LP to enhance the lack of property chaining in OWL-DL. In SWRL, DL-safe is the condition where variables occurring in each rule's head are also required to occur in its body to ensure the decidable reasoning of the rule engine. The availability of SWRL rule and ontology integration development in the popular Protégé environment[10] makes the SWRL model the most attractive one for people to use.

In the loosely coupled integration, DL-log [?], AL-log [?], and DL+Log [?] are three well-known models. In these models, rules are extended to Horn rules. Besides, not all of the terms in rules are required from ontology so rule module in these models provides more powerful knowledge representation and rule reasoning than the ones in SWRL. However, none of loosely coupled integration models provide standardized XML markup languages and a development environment, as SWRL does in Protégé. Obviously, this will be a challenge to represent and execute rule systems for loosely coupled integration on the Web. Moreover, the reactive rules [?] have not been seriously considered in all of the ontology and rule integration models. This will be the biggest impediment to implement rule and ontology systems for distributed applications on the Web.

3 Policy as Ontology and/or Rule

Since computers understand the data semantics in the Semantic Web, people are much more satisfied with the search results when a semantic search engine is fully developed. Policy-aware Web extends Semantic Web that provides computerized policies, such as privacy protection or digital rights management policies for computers to understand and execute automatically [?]. However, pure rule and/or ontology languages are not explicit enough to represent policies that regulate human behavior in the real world. We need a well-defined policy language that describes the concepts of rights, obligation, conditions, resources, etc. between resource owner and user to represent and execute access control policies of resources on the Web.

Following [?], policies are considered as knowledge bases, allowing deontic classes, properties, and access control rules. This has the advantage that many operations are automated, thereby reducing ad hoc program coding to a minimum and enabling automated documentation. Regulations imposed on human behavior and activity are simulated by computerized policies that are specified by using policy languages, such as Rei or KAoS [?]. The semantics of these policy languages is only DL-based, and needs to be further extended by using LP-based semantics of rule languages, such as RuleML, RIF or Protune [?]. Recently, AIR

[10] http://protege.stanford.edu/

(AMORD In RDF) is a policy language that considers using both RDF ontology language and N3 rule language for the privacy protection policy execution[11].

3.1 Policy for Semantic Web Services

The idea of Web Services in the SOA of distributed software systems has become a tremendous success. Semantic Web Services (SWS) employ Semantic Web technology in the Web Services area: service functionality, service inputs and outputs, preconditions and effects, etc.; all are expressed and executed in knowledge representation languages, i.e., ontology and rule languages [**?**]. A policy can be considered in the SWS because of using similar ontology and rule languages' semantics on the Policy-aware Web. Thus policies are represented and executed as Web Service rules for the compliance of human regulations to control information sharing and service execution.

One of the challenges to implement rule systems on the Policy-aware Web is how to design and implement rules as computerized policy by the integration of rule and ontology. This computerized policy imitates human regulation for controlling information sharing and service execution for a composite web service on the Web. And the ultimate goal is the satisfaction of legal regulation compliance from the execution of a computerized policy. This idea is similar to the Legal Knowledge Interchange Format (LKIF) proposed in the past EU FP6 project [**?**].

4 Rule Management Systems and Engines

Before looking into the details of rule management systems, we need to decide about a rule management systems implementation platform. If we choose a rule system that is also embedded in the Semantic Web development environment, then we have several advantages. First, it provides sufficient facilities to implement subsystems for rules and the data model. Second, both the ontology and rule languages used in the Semantic Web are complementary to each other so we can leverage on the declarative knowledge representation. Third, we have a standard query language or a rule language to support the access of underlying knowledge bases for ontology or rule bases. Finally, if applications are embedded in Java or some other popular imperative programming language, we have language typing, control flow, and interaction mechanism available for the implementation of application system on the Web.

4.1 Rule Systems in the Semantic Web Framework

The SemWebCentral[12] is one of the well-known websites providing Open Source development tools for the Semantic Web. The Semantic Web system development framework can be subdivided into three subsystem modules: an application module, a controller module and a view module. The application module

[11] http://dig.csail.mit.edu/TAMI/2007/amord/air-specs.html
[12] http://www.semwebcentral.org/

contains reasoning functions, including task and inference, domain schema and knowledge base. The controller module handles interactions with the user and functions in the application model. The view module provides output for the user. The Semantic Web system development framework usually includes two development parts, one is for ontology and the other is for rule. For example, Protégé has been successfully developed for ontology and rule, such as Jena[13] and Jess[14]. The Jena rule engine was integrated in the Semantic Web system development framework Protégé for having rule-based inference with the access to knowledge base in the ontologies of RDF and OWL[15]. In addition, the system for development of ontology and rule combination, such as SWRL is also available in the Protégé with SWRLTab[16].

4.2 Standalone Rule Systems

A number of standalone rule systems have been investigated by the RIF Working Group[17]. A rule system is defined as a piece of software that implements or supports a rule language in some way (e.g., a rule engine or a rule editor). Among the RIF list, some rule systems are developed for commercial usage but others are for open source purposes. Based on the classification of rule types, some rule systems are developed for a deductive rule engine but others are implemented for a reactive rule engine.

Commercial Rule Systems IBM ILOG Business Rule Management Systems (BRMS)[18] provides a complete BRMS for analysts, architects and developers, featuring tools of rule authoring and rule management besides its rule engine. In fact, ILOG JRules is one of the best-known production rule systems. JBoss Drools[19] Enterprise BRMS is a well-known open source rule system which provides perfect integration with the service-oriented architecture (SOA) Web service solutions. On the other hand, existing rule systems, such as Prova[20] and ruleCore[21] are also available for ECA rules inference. For more details about reactive rules on the Web please refer to [**?**].

Some commercial rule systems are developed from a matured prototype of the Semantic Web middleware, such as OntoBroker[22]. Oracle Business Rules integrates with the Business Process Execution Language (BPEL) and tries to enrich decision making for processes in the SOA[23]. In general, commercial rule

[13] http://jena.sourceforge.net/
[14] http://www.jessrules.com/
[15] http://protege.stanford.edu/plugins/owl/jena-integration.html
[16] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab
[17] http://www.w3.org/2005/rules/wg/wiki/List_of_Rule_Systems
[18] http://ilog.com/products/businessrules/index.cfm
[19] http://www.jboss.com
[20] http://www.prova.ws
[21] http://www.rulecore.com
[22] http://www.ontoprise.de
[23] http://www.oracle.com/technology/products/ias/business_rules/index.html

systems use proprietary rule languages for the development of the rule bases so we need a standard rule interchange language, such as RIF to obtain rule interoperability among these rule systems.

Academic Rule Systems The academic ECA rule system XChange, with its integration of the Web query language Xcerpt, provides the access of data sources to obtain information on the dynamic Web. Other academic rule systems are deductive reasoning rule engines, such as jDREW and its object-oriented extension OO jDREW[24]. An Object-Oriented Knowledge Base Language FLORA-2 provides frame-based logic reasoning engine with the knowledge base development environment[25].

Logic programming systems are also used to develop rule-based applications. For example, Logic Programming Associates Prolog provides a complete rule development environment with a graphical interface for rule editing[26]. Thea is a Prolog library for generating and manipulating OWL content on the Semantic Web. The Thea OWL parser uses SWI-Prolog's Semantic Web library for parsing RDF/XML serialisations of OWL documents into RDF triples, and then it builds a representation of the OWL ontology[27].

One of the challenges for implementing rule systems on the Web is to be aware of the current rule management systems, including commercial and academic ones, and, furthermore, an understanding of their system features and which rule type reasoning they can support. Moreover, we need to investigate the possible application domains they intent to accomplish through the underlying rule interchange standard.

4.3 Performance Benchmark for Rule Systems

It is not easy to propose an acceptable measurement benchmark to evaluate the performance of current rule systems because the diversity of rule syntax and features are available for different rule systems. In [**?**], a set of benchmarks were proposed for analyzing and comparing the performance of numerous rule systems. In this OpenRuleBench, they include five rule technologies to compare with: Prolog-based, deductive databases, production rules, triple engines, and general knowledge bases. Jena and OntoBroker we mentioned before were also two of the selective rule systems in their comparison list. We envision that the benchmark performance evaluation output will be just one of the criteria for people to decide for which rule system they are going to adopt in their application development.

[24] http://www.jdrew.org/oojdrew/
[25] http://flora.sourceforge.net/
[26] http://www.lpa.co.uk/
[27] http://www.semanticweb.gr/TheaOWLLib/

5 Rule Interchange Languages

In early expert systems, a specific language, such as Prolog or LISP was used to encode expert domain knowledge into rules and facts, for execution in a standalone system. However, when rules and facts are created in different rule systems and distributed on the Web, we need a rule standard exchange language for the interchange of heterogeneous rule formats. Otherwise, we cannot implement an application, such as composite (semantic) web services that might require rules created and distributed in the different rule systems [**?**]. Therefore, a common rule format facilitates decision making on the network environment with multiple rule formats. For example, the therapeutic guideline recommendation rules for diabetes type 2 are constructed with the combination of clinical and therapeutic criteria as the condition part and therapeutic options as the actions. When users or organizations switch rules from one rule product to another, they can employ the rule interchange technologies without re-developing their rules.

Proposed rule interchange languages include RuleML [**?**], REWERSE Rule Markup Language (R2ML) [**?**], and W3C RIF[28], where R2ML attempts at integrating aspects of RuleML, SWRL, and Object Constraint Language (OCL). The most recent W3C RIF[29] was proposed to achieve the objective of rules interchange and interoperability for major rule systems. These rule interchange languages provide XML schemata to guarantee the comparability of rule syntax and semantics from source to target rule systems and vice versa. The other important rule language is Semantics of Business Vocabulary and Business Rules (SBVR), submitted by Business Rule Group (BRG) to OMG on the standardization of semantics for business vocabulary and rules[30].

One of the challenges to apply rule systems on the Web is to finalize a rule interchange language to provide a rule interchange framework and format of rules for current major rule systems. When agents proceed towards a two way rule interchange, a rule interchange language with the framework ensures the compatibility of rules' syntax and semantics between rule systems. The related challenge is the requirement to have a software development system and a runtime environment for people to build, design, and implement standardized rule interchange formats to automatically extract and transform rules from different rule systems on the Web.

6 Use Cases with Rules

Rules are used to express computational or business logic in the information systems which do not have explicit control flow, so rules are more suitable for execution in the dynamic situations for business collaborations. Along with the rapid development of the Web, multiparty collaborations for carrying out business services in this environment are more significant than ever before. For ex-

[28] http://www.w3.org/TR/rif-bld/
[29] http://www.w3.org/2005/rules/wiki/RIF_Working_Group
[30] http://www.businessrulesgroup.org/sbvr.shtml

ample, when a credit card transaction is requested from a merchant, a customer needs a payment authorization from the merchant and the card issuer (the bank) to accomplish a successful transaction service. In this case, both merchant and bank have their own policies as rules to conduct their authorization processes.

If both parties are required to combine their policies, we hope they can transform the rules into a formal common rule format, such as RIF. For example, rules from the bank are directly imported by the merchant and processed with his local rule engine to derive an authorization decision. In addition, this situation can be extended to other relevant web services for conducting composite web services. Another use case is a seller, posting his price discount and refund policies for execution as rules on his website, to attract potential customers for making a purchase decision from his selling goods. Moreover, a vendor advertises his lead time policies in formal rules to attract customers and also as a part of contract negotiation in the supply chain management.

Use cases such as the ones we have shown above are categorized by the W3C RIF Working Group as a type of policy-based transaction authorization policy for access control with the interchange of human-oriented business rules. Several other interesting use cases focusing on different application domains are also available on this website[31]. Another interesting use case study was proposed by the Business Rule Group (BRG) to use SBVR for illustrating business rule concepts of EU-Rent, EU-Fly, and EU-Stay. They are available on the BRG website[32]. The challenge here is whether we have enough use cases that can be accomplished by current rule systems on the Web to convince people to adopt and use this technology.

7 Conclusion

In this study, we outlined the objectives of RuleML-Challenge competitions started in 2007. Alos, we have elaborated the possible research and implementation challenges for rule systems on the Web that are closely related to the Challenge competition events in the forthcoming years.

The first challenge is to perfectly implement rule systems with the data model, either from a relational or object-oriented database or from a DL-based knowledge base. The second challenge is to enable computerized policies, created in a policy language that is compliant with human legal regulations. In addition to the legalized policy implementation with policies created from the policy language, computerized policy can be shown as a combination of ontology and/or rule languages for the purpose of information sharing and web service execution. The third challenge is full awareness of current available commercial and open source rule management systems and, moreover, finding out the pros and cons of each rule system by a standard evaluation benchmark to verify its scalability and performance. The fourth challenge is to achieve rule interoperability using available rule interchange languages for rules created and distributed on

[31] http://www.w3.org/2005/rules/wg/wiki/Use_Cases
[32] http://www.businessrulesgroup.org/egsbrg.shtml

the Web. The fifth challenge is to demonstrate sufficient use cases implemented from rule systems, while interchanging their rules through one of rule interchange language.

Acknowledgements

This research was partially supported by the NSC Taiwan under Grant No. NSC 95-2221-E004-001-MY3 and NSC 98-2918-E-004-003.