A Framework for Web-based Interoperation among Business Rules

Yevgen Biletskiy
Department of Electrical and Computer Engineering
University of New Brunswick
Fredericton, Canada
biletski AT unb.ca

Abstract—**The present paper describes the approach and two technical solutions for interoperation between business rules represented in various formats. The Semantic Web techniques are used to enable this interoperation. One of the interoperation methods uses the Java Interoperation Object (JIO) described in the context of Positional-Slotted Language (POSL), which a human-friendly variant of the Rule Markup Language (RuleML), and Notation 3 (N3) representations. Details of the connections between these document representations are demonstrated with the use of query-based interoperation between POSL and N3. Another solution described in the present paper is conversion of business rules stored in Microsoft Excel as decision tables into POSL using OpenL tablets. Although the current business rules interoperation framework involves three formats (Excel, POSL, and N3), it can be extended to other document representations through appropriate conversions of data in rule bases and queries.**

Keywords— Knowledge Representation, XML, RDF, Notation 3, Positional-Slotted Language, Rule Markup Language, Semantic Web, Query

I. INTRODUCTION

Business rules are becoming ubiquitous in modern industry and are usually created, stored, and maintained by business analysts, knowledge engineers and software engineers in various formats. Some formats are technical, and some formats are non-technical and more user-friendly. Classically, business rules are logic constructs (e.g. "IF-THEN" type), and they are often represented using decision tables or decision trees. Technically, business rules can also be implemented using a programming language like C or Cobol, or by the use of a controlled English. There are many specific solutions for creation and maintenance of business rules. For instance, Microsoft Excel tables can be deployed as a user-friendly way to build documents representing decision tables. Systems like Drools provide excellent platforms to build and maintain more complex business rules. There are some standards for business rules representations. The most known is the Semantics of Business Vocabulary and Business Rules (SBVR) [1] adopted by the Object Management Group (OMG). With the wide proliferation of the Semantic Web techniques, some new languages for business rules representation appeared, for instance, Rule Markup Language (RuleML) [2], Positional-Slotted Language (POSL) [3] and Notation 3 (N3) [4]. These and other Web-based techniques can be integrated with the purpose to find better business solutions based on information stored in different rule bases accessible through Internet. The purpose of the present work is to enable semantic interoperability between business rules created in various formats.

II. THE FRAMEWORK

The resented approach to interoperation is based on semantic interoperability using a mediator, which can convert business rules among various knowledge representations. The software mediator can process and interpret business rules stored in various formats, as well as convert a query formulated in any of these formats to search an answer in all rule bases connected. This will assist clerks, brokers, managers, and other specialists in finding better business solutions and decision-making.

Since business rules become Web-based, the modern solutions for interoperation can be deployed. The solutions presented in this work use the Sematic Web infrastructure and related tools. The Semantic Web offers solutions allowing to semantically enriching business rules using a background ontology, which serves as a knowledge base (or vocabulary). On the other hand, the disadvantage of creating and maintaining business rules in a Semantic Web language is that rules are difficult for human understanding. Even POSL, which is more human-oriented than RuleML, is difficult for a non-specialist to understand. The focus of the present work is query-based interoperation between two Semantic Web based languages: POSL and N3, and conversion of business rules in MS Excel format into POSL. The focused interoperation framework is presented in Fig. 1.

The framework presented in Fig. 1 consists of the following main components:

1. POSL rule base, which consists of business rules and facts in the POSL format. rule base, which consists of rules and facts in the N3 format.
2. MS Excel database, which contains rules and facts in the user-friendly format.

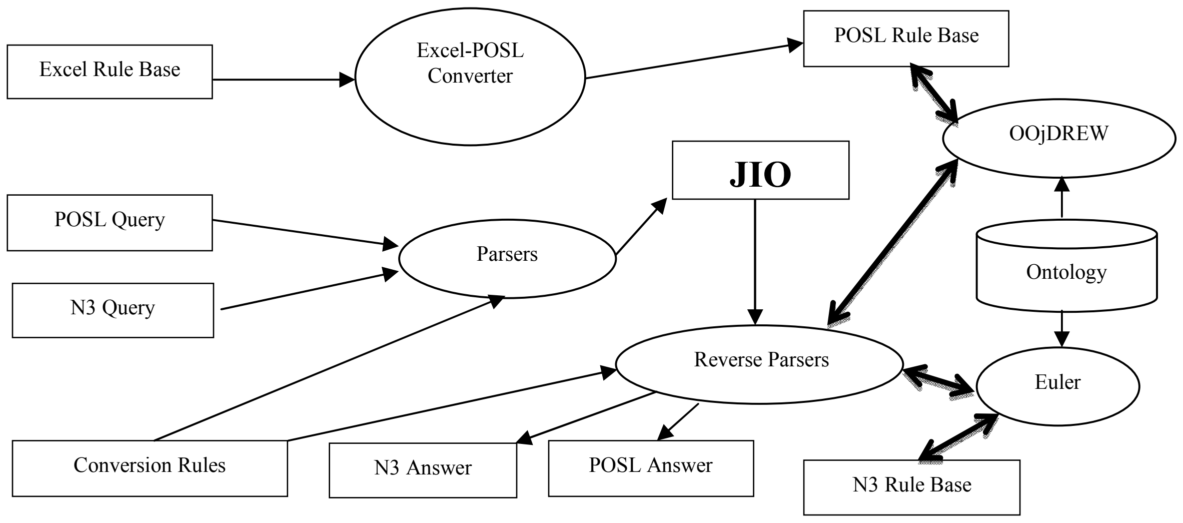

Figure 1. Focused Business Rules Interoperation Framework

3. JIO (Java Interoperation Object), which The Java Interoperation Object is the basis of interoperation methods developed for the Knowledgebase Representation Interoperation Tool (KRLIT) described technically in [5]. The objective of KLIRT is to facilitate interoperation among existing knowledge representation paradigms through a universal Java-based architecture, which used RuleML as a building block. The current KRLIT has been successfully developed and used for query-based translation between POSL and N3 knowledge bases.
4. Excel – POSL converter is presented in [6]. POSL can be generated from Excel decision tables. Before this, it is necessary to identify and extract the facts and rules contained within the tables. OpenL Tablets [7] provides an API that facilitates simple creation and processing of Java-based tables in Microsoft Excel. While OpenL itself has a rule engine capable to process these decision tables, this engine does not offer semantic enrichment with a background ontology and application-independence. As a result, OpenL is used solely for its ability to process Excel tables and externalize Java code from the application logic. The OpenL table parser uses templates to extract the relevant information (data and rules) from the decision tables into memory, where it can generate semantically rich POSL.
5. Reasoning engines: OOjDREW [8] for RuleML/POSL and Euler for N3.
6. Parsers – to syntactically analyze business rules in POSL and N3, and convert them into the JIO format.
7. Reverse parses – to convert business rules from JIO format into a format required by the user.

In the present framework, the user can query the knowledge bases using POSL or N3 formats, and receive the answer in the same format as query.

III. JIO, POSL AND N3

The technical details of JIO (Java Interoperation Object) architecture and the use of JIO within the KRLIT are explained in [5]. The Java Interoperation Object is the basis of methods for business rules interoperation. The objective of this component is to capture as many aspects of the various knowledge representation paradigms available. This concept is used to translate supported POSL and N3 rule bases. JIO uses atoms to represent chunks in a rule base (e.g. for POSL this is a relation, for N3 this is a subject).

POSL provides object-centered instance descriptions via binary properties, taxonomies over classes and properties, class-forming operations and class/property axioms, and derivation, integrity, transformation, and reaction rules [3]. POSL is a more human-readable language than the XML-based RuleML [2], but has the same language constructs. POSL has two representation paradigms which it can use, depending on what the user requires. The first of which is *positional*; this means that slots are not used to represent relation contents. The second option is *slotted*; this means that property names are associated with every element in a relation. The latter best suits our JIO framework, and so this paradigm has been chosen.

Notation 3 is a compact, rule-extended version of RDF's XML syntax [4]. In this way, RDF's complex machine understandable language becomes more readable to humans. RDF facts and rules are still written with triples (subject – property - object) and so this language is expressive in nature, but also good for human comprehension.

In order to deal with any input and fetch the answers from the available Knowledge, the system should re-present this input in order to convert it to intermediate JIO representation (RuleML building blocks) which from-and-to the system can be interoperated to the target language. The interoperation process using the JIO representation can be done by implementing *Parser* and *ReverseParser*.

The main goal of Parser is to take a query or answer of a query of a language from a File, URL or as a String in the form of *InputObjectCollection*, and then parses it (breaks down) to RuleML building blocks, which will compose a single *AtomCollection* as JIO representation in order to provide it to *ReverseParser* as input. Similarly, *ReverseParser* takes this JIO *AtomCollection* as input and reverse parses it (translates) to the target language as output with option of returning the result as an answer or query. Details of parser's implementation are presented in [5].

IV. Interoperation between POSL and N3 Business Rules

The present work describes POSL-N3 interoperation using an example of an insurance company *Farm Insurance* and two on-line insurance brokers, which are insurance companies *Mainland Insurance* and *Healthy Life*. The companies use different knowledge representation languages, but use the same schema for their facts and rules. Assume the *Farm Insurance* has a business rules set describing automobile insurance Age-Class discounts as follows:

Age From	Age To	Customer Class	Discount Value
16	20	Economic	0.0
21	25	Economic	0.1
26	30	Economic	0.2
16	20	Gold	0.2
21	25	Gold	0.3
26	30	Gold	0.4
16	20	Platinum	0.5
21	25	Platinum	0.6
26	30	Platinum	0.7

This rule base is not accessible by individual users because it is not Web-based, but can be accessed by insurance brokers through some internal communications.

Mainland Insurance focuses primarily on *Economic*-class customers, and prefers to use an N3 knowledge base as follows (policy for providing a discount of *10%* to an *Economic* customer who is between the age of *21* and *25)*:

```
{ ?Client
                :type           :client;
                :clientID       ?ClientID;
                :age            ?Age;
                :name           [:type :fullname; :first ?FName; :last ?LName];
                :class          ?b.
     ?Age math:notLessThan       21 .
     ?Age math:notGreaterThan   25 .
     ?b     log:equalTo             :Economic. }
     =>
     {?resultDiscount
     :type            :Discount;
                    :company       :MainlandInsurance;
                          :clientID           ?ClientID;
                          :age                ?Age;
                          :class              ?b;
                          :discount           0.1.}.
```

Healthy Life focuses on *Gold* and *Platinum*-class customers, and prefers to use a POSL knowledge base as follows (policy for providing a discount of *20%* to a *Gold* customer who is between the age of *16* and *20)*:

```
Discount(company->HealthyLife; clientID->?ClientID; age-> ?a:Integer;
           class-> ?b;discount-> 0.2:Real) :-
              client(clientID->?ClientID;age->?a:Integer;
                   name->fullname[
                                first->?FName;
                                last->?LName];
                   class-> ?b),
                     greaterThanOrEqual(?a, 16 : Integer),
                     lessThanOrEqual(?a, 20 : Integer),
                                      equal(?b, Gold).
```

Suppose the customer is familiar with POSL only, but wants to find discount policies of both insurance brokers. The query is:

```
Discount(company->?All; clientID->?clientID; age->?age;
           class-> ?b;discount->?discount).
```

If business rules interoperation is not enabled, the only *Healthy Life* database is accessible. During query processing time, this query is transformed by POSL parser into JIO, and the N3 reverse parser class accepts the transformed query as input. This provides the N3 representation of this POSL query as follows:

```
        ?subject
                    :type                  :Discount;
                    :company:MainlandInsurance;
                   :clientID   ?ClientID;
                    :age                   ?Age;
                    :class                 ?Class;
```

This query in POSL is given to OOjDREW, whose answers are returned in POSL. Since now the equivalent N3 query is available, it can be given to Euler as input, whose answers are given in N3. The answer is used by N3 parser and stored in JIO. It is then sent to the POSL reverse parser to generate the POSL representation of the N3 answers. This answer is combined with the OOjDREW answer resulting in the following combined POSL answer:

```
Discount(company->MainlandInsurance;
        clientID->1:Real;age->19:Real;class->Economic;discount->0.0:Real).
Discount(company->MainlandInsurance;
        clientID->6:Real;age->17:Real;class->Economic;discount->0.0:Real).
Discount(company->MainlandInsurance;
        clientID->2:Real;age->22:Real;class->Economic;discount->0.1:Real).
Discount(company->HealthyLife;
        clientID->3:Real;age->19:Real;class->Gold;discount->0.2:Real).
Discount(company->HealthyLife;
        clientID->5:Real;age->30:Real;class->Gold;discount->0.4:Real).
```

```
Discount(company->HealthyLife;
          clientID->4:Real;age->29:Real;class->Platinum;discount->0.7:Real).
```

The answer is consistent with the business rules maintained by the Farm Insurance.

V. Interoperation between MS Excel and POSL usiness Rules

Assume *Healthy Life* would like to update its rule base automatically using data from Excel sheets created by *Farm Insurance*. The business rules below need to be converted from the user-friendly Excel format into POSL:

Age From	Age To	Customer Class	Discount Value
16	20	Gold	0.2
21	25	Gold	0.3
26	30	Gold	0.4
16	20	Platinum	0.5
21	25	Platinum	0.6
26	30	Platinum	0.7

Using rule transformation templates, the table was automatically converted to POSL syntax, parsed, and loaded into the OO jDREW reasoning engine [8]. Examples POSL rules derived from the rules above are:

```
Discount(?a : Integer, ?b : Customer, 0.2 : Real) :-greaterThanOrEqual(?a, 26 :
Integer), lessThanOrEqual(?a, 30 : Integer), equal(?b, Economic : Customer).

Discount(?a : Integer, ?b : Customer, 0.2 : Real) :- greaterThanOrEqual(?a, 16 :
Integer), lessThanOrEqual(?a, 20 : Integer), equal(?b, Gold : Customer).
```

As a test, the following query was issued to OO jDREW:

```
Discount(25 : Integer, Gold : Customer, ?discount : Real).
```

The query asks “what is the discount value for a customer with age 25 and type Gold?” The results of query, issued using the OO jDREW Top-Down reasoning engine, are as follows:

```
?discount = 0.3 of type Real.
```

The solution presented allows automatically updating the rule base in POSL using rules created in Excel. A similar solution can be developed for N3. This allows business analysts to work with user friendly formats rather than to use heavily human readable Semantic Web languages.

Conclusion and Future Work

The present paper has described the business rules interoperation framework as a solution to the Web-based interoperation gap issue. The work has focused on interoperation between business rules created in two different Semantic Web languages. The usage examples have been presented. The second focus of the paper is a methodology to partially automate the process of converting human-readable business rules stored in the form of MS Excel tables to machine-processible POSL, with the goal of combining the ease of use of Excel-based rule tables with the semantically-rich queries supported by reasoning engines. Although the work in current state covers Excel, POSL and N3 formats only, it can extend to other business rules representations.

Acknowledgment

The present work includes business rules interoperation techniques which had been implemented as part of students' projects conducted at the University of New Brunswick under supervision of the author. The students were Girish R Ranganathan (Dr. Ranganathan now), J Anthony Brown, Taylor Osmun, and Patrick Thébeau. The main funding sources for this work are NSERC Discovery Grant and ACOA AIF.

References

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

[2] H. Boley, The RuleML Family of Web Rule Languages, Invited Talk. *Proc. Fourth Workshop on Principles and Practice of Semantic Web Reasoning,* Budva, Montenegro, LNCS 4187, Springer-Verlag (2006) 1-15.

[3] H. Boley, POSL: An Integrated Positional-Slotted Language for Semantic Web Knowledge (2004). Available: http://www.ruleml.org/submission/ruleml-shortation.html.

[4] T. Berners-Lee et. al. Notation (N3), A readable RDF Syntax. Available: http://www.w3.org/TeamSubmission/n3/

[5] T. M. Osmun, P. Thébeau, Y. Biletskiy. Knowledgebase Representation Language Interoperation Tool. *In Proc RuleML America*, LNCS 7018, Springer-Verlag (2011) 58-65.

[6] Y. Biletskiy, G. R. Ranganathan, J. A. Brown. Representing User-Friendly Business Rules in a Semantic Web-Based Format. *ISAST Transactions on Computers and Software Engineering* 2(1) (2008) 8-12.

[7] OpenL Tablets, Available: http://openl-tablets.sourceforge.net/index.html

[8] Ball M., Boley H., Hirtle D., Mei J., and Spencer B. The OO jDREW Reference Implementation of RuleML. *In Proc. Rules and Rule Markup Languages for the Semantic Web (RuleML-2005)*, LNCS 3791, Springer-Verlag (2005) 218–223.

[9] Euler. Available: http://eulersharp.sourceforge.net/

.