QUERYING ONTOLOGIES ON THE BASE OF SEMANTICS OF BUSINESS VOCABULARIES AND BUSINESS RULES

Algirdas Sukys, Lina Nemuraite, Edvinas Sinkevicius, Bronius Paradauskas

Kaunas University of Technology, Department of Information Systems, Studentu 50-308, Kaunas, Lithuania, sukys.algirdas@gmail.com, lina.nemuraite@ktu.lt, edvinas.sinkevicius@ktu.lt, bronius.paradauskas@ktu.lt

Abstract. Today information systems more and more often rely on ontologies that are able to represent meaningful concepts and complex relationships among them relevant for business models and their supporting software systems. However, ontology development and access to ontological data is only possible on deep technological level that is not friendly for business experts. The goal of the paper is to present a possibility of querying OWL ontologies using semantic formulations of Semantics of Business Vocabulary and Business Rules (SBVR), expressed in SBVR Controlled English. We introduce the initial approach for specifying ontology queries as SBVR questions and transforming them to SPARQL.

Keywords: ontology, SBVR, OWL, SPARQL.

1 Introduction

Semantics of Business Vocabulary and Business Rules (SBVR) is the OMG created metamodel and specification that defines the vocabulary and rules for describing the business semantics – business concepts, business facts, and business rules using some kind of Controlled Natural Language [18]. The SBVR specification considers SBVR Structured English (SSE) as the concrete language for this purpose though other languages are possible. Formal SSE sentences cannot represent all possible constructions of natural English language because they are precise, based on predicate logic with a small extension in modal logic. Nevertheless, SSE language is understandable to business and information system experts as it looks like natural language, and is interpretable by machine in the same breath. The linguistic analysis is outside the scope of SBVR specification; however, SBVR supports linguistic analysis of text for business vocabularies and rules.

In our previous work, we were focused on generating UML&OCL models from SBVR specifications. We proposed the methodology for specifying information system requirements on the base of SBVR business vocabularies and business rules related with business process models, and implemented the prototype of tool VeTIS* supporting that methodology [3], [14]. VeTIS tool is capable to recognize SBVR concepts (object types, roles, fact types, fact type roles, individual concepts) and business rules (various kinds of semantic formulations) that make foundations for conceptualizing business and correspond to knowledge and metaknowledge level of knowledge triangle [16]. However, the complete specification of business conceptual schema and conceptual model needs to include business facts – instances of fact types (ground facts) and propositions (instances of complex semantic formulations based on several fact types) comprising the bottom knowledge level (i.e. fact level) [17]. Therefore, the business semantics editor should recognize instances of concepts and logical formulations needed for specifying ground and complex facts that make sense for various purposes:

- Validating conceptual schema, and conceptual model;
- Validating business facts;
- Representing business facts for different groups of people;
- Formulating SBVR questions about business and answers to them.

The last issue is the focus of the current paper, which presents the idea of transforming SBVR questions into SPARQL queries [9] that can be executed against Web Ontology Language (OWL) ontologies obtained from SBVR vocabularies and business rules. Consequently, it would be possible to describe business vocabularies and business rules, and querying software systems generated on the base of these vocabularies and rules using a single language and terminology.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 outlines the overall process of creating SBVR business vocabularies and business rules from SSE style texts, formulating SBVR questions, transforming them into SPARQL queries and executing them against ontologies generated from SBVR vocabularies and rules. Section 4 is devoted to formulating SBVR questions. Section 5 presents

transformation of SBVR questions into SPARQL and example of such transformation. Section 6 draws conclusions and outlines the future work.

2 Related works

The contributors of SBVR emphasize that SBVR specification is devoted for business people and for business purposes independently of information systems designs, i.e. for using business vocabularies and business rules as guidance for the business and different groups of people: workers, customers, suppliers, partners. OMG also proposes the XMI schema for the interchange of business vocabularies and business rules among organizations and between software tools. SBVR metamodel and XMI schema may be used for developing software tools for managing business vocabularies as well as for automating development of software for managing business on the base of business semantics, i.e. in the way different of previously existed approaches, e.g. [15], [19]. SBVR business vocabularies are transformable into UML&OCL [3], [14], and vice versa [2]; BPMN [1], RDB schemas [13], OWL [4], Web services [5], [7] etc. Besides automating development of software models and code [11], [12], SBVR Structured English may serve for creating semantic specifications of legacy information resources, integrating these resources, implementing contextualized and multilingual information systems, etc. The power of SBVR is nicely disclosed by the fact that SBVR specification itself is formally written in SSE [10].

Applying SBVR in practice, various limitations become obvious. For example, SBVR lacks the larger collection of data types and patterns for constructs needed for expressing arithmetic operations, data and time, past and future events and similar. SBVR specification is easy extensible, however, it would be desirable having standard constructs for most frequent cases. Spreeuwenberg and Anderson notice more deficiencies of SBVR: lack of inference; lack of references (rules should be stated in one sentence); necessity to introduce concepts before referencing to them; impossibility to express directives etc [22], [23]. Surely, SBVR should be extended in the future.

From the other side, until now only part of SBVR was used. For example, SBVR questions provide a nice capability of querying business models and their implementations but they attained a little attention in SBVR specification and related research. Kriechhammer in 2006 has noticed about possibilities of SBVR questions [9] for business people to query systems for business modeling without the support of programmers. SBVR questions are based on logical projections that are much more comfortable for business people than various query languages that are platform-specific and suitable for IT specialists having the experience of working with special tools.

Realizing the idea of querying business in SBVR requires a creation of a whole infrastructure including tools for authoring SBVR business vocabularies and rules, transforming them into various software models and code, including OWL, SQL, Web Services, business process execution languages and so on. Several EU projects are devoted for this purpose: OPAALS* (2006-2010, generating Web services and data models from SBVR specifications [13], [21]), ONTORULE** (2009-2012, aiming at integrating knowledge and technologies needed for extracting ontologies and business rules from various documents, including natural language texts; managing them and implementing in software systems). The commercial tool suite for Business Semantics Management Collibra*** presents capabilities for authoring SBVR vocabularies and rules, generate ontologies and various models of information systems.

None of the mentioned works addresses SBVR questions (except [9], but no further research in that direction was done). As interest in SBVR is growing and corresponding infrastructures are arising, we present the initial methodology for transforming SBVR questions into SPARQL.

3 Process of transforming SBVR Structured English questions into SPARQL

Process of transforming SBVR Structured English questions and executing them as SPARQL queries is presented in Figure 1. First of all, we need to define SBVR business vocabulary and business rules, and specify individual concepts and facts of the problem domain under consideration. Business concepts and rules are specified in SSE style text that consists of terms, verbs, names and keywords [18], and needs to be parsed for transforming them into SBVR schema. Our parser [14] is based on Antlr grammar and recognizes strict sentence structures. The development of a flexible parser is also a problem investigated in [8]. In a general case, an input into the parser may come from linguistic analysis tools.

In the next step, the SBVR model should be transformed into OWL ontology, which will be a data source to execute query. After SBVR model is defined, a question can be specified, using keywords, terms, verbs and names of the problem domain. The formulated question should be parsed to generate SBVR model, which

* http://www.opaals.eu/
** http://ontorule-project.eu/
*** http://www.collibra.com/
eventually can be transformed into SPARQL query. In real life applications, SPARQL queries will be executed on ontologies that may be stored in relational databases [24] and may be related with various software applications. These ontologies can specify business knowledge including business data, services and processes. The potential possibilities of applying SBVR questions are invaluable for business people but implementing such capabilities require many efforts from the research community.

Figure 1. Process of transforming and executing SBVR questions

In Figure 1, the dashed rectangular marks artifacts concerned in the current paper. Figure 1 also includes artifacts that are not considered here. Functionality of specifying business vocabularies and transforming them into SBVR schema was implemented in our first prototype [14]. Transforming SBVR schema into OWL is under development by other members of our research group (please look at the paper of Karpovic, Nemuraite in the current IT’2011 proceedings).

4 Modeling SBVR questions

The basic SBVR concept types that can be presented in a business vocabulary are object types (or general concepts), fact types (or verb concepts), and roles. For formulating SBVR questions, a vocabulary should be capable to represent individual concepts and facts. SBVR individuals and facts are essential elements for representing OWL ontology individuals and their relations.

SBVR defines individuals using logical formulations instantiation formulation, which relates an individual concept with the object type. The instantiation formulation means a classification of things that are individual concepts of some object type. It is also necessary to use the assortment fact type that is defined with respect to a given general concept and a given individual concept such that each instance of the fact type is an actuality that the instance of the individual concept is an instance of the general concept. We present a fragment of SBVR metamodel, specifying an individual concept, and an example of SBVR model of fact type “Ktu is university” in Figure 2 and Figure 3.

Figure 2. SBVR metamodel fragment specifying the individual concept
SBVR question is meant by closed logical formulation – question nominalization (projecting formulation). SBVR questions are based on fact types, defined in a business vocabulary. For example, question “What journals are referred in the database Web of Science?” is based on the fact type “journals are referred in the database”. Complex questions consider complex facts based on several fact types. Complex facts are constructed using keywords “and”, “that” or “which”. The example of a complex question is “What journals are referred in the database Web of Science and have an impact factor, which is greater than 0, and are published by the university that is in city Kaunas”.

After specifying a question using SBVR Structured English, we parse it using Antlr grammar and generate SBVR model of the question. Model is based on Eclipse EMF SBVR 1.0 metamodel. We present a fragment of SBVR metamodel for formulating questions in Figure 4.

For demonstrating how questions are formulated in SBVR let us take an example (Question1) – “What universities are in city Kaunas?”. SBVR model of this question is given in Figure 5. A core element of a question in SBVR model is the question, which is a meaning of interrogatory. A question is formulated by a closed projection, so the result of the projection answers the question. A question starts with the interrogative operator “what”, which means things that we want to see in the answer.

In the SBVR model, a closed projection that means a question is formulated by a question nominalization (projecting formulation). Projecting formulation has a projection on a variable that ranges over a concept that matches the operator. In our example we use variable1 which ranges over object type university. That means, we want to get a list of universities. Closed projection is constrained by the universal quantification, which has a variable2 that ranges over the object type university.

Universal quantification is restricted by a logical formulation aggregation formulation. The aggregation formulation is used to associate a variable with a set of things that satisfy some condition. It formulates natural language expressions of the form: “let <variable> be the set of all things t such that <some condition involving t>” [18] so that <variable> can then be used in other formulations regarding the set. In our
example *aggregation formulation* set restrictions for university objects. These restrictions are expressed by a projection, which has *atomic formulation*, based on a fact type university is_in city. Object type of the second fact type role of this fact type is connected with individual concept 'Kaunas'.

5 Transforming SBVR questions into SPARQL queries

SBVR vocabulary should be represented as OWL ontology for transforming SBVR questions into SPARQL queries where SPARQL is a query language for ontologies. Ontologies are stored using Resource Description Framework (RDF) data model that is considered to be the most relevant standard for data representation and exchange on the Semantic Web [6]. RDF statements are expressed in the form of subject-predicate-object. A set of RDF statements compose directed labeled graph. SPARQL query contains a set of triple patterns called a basic graph pattern. Triple patterns are similar to RDF triples, except that each of this triple can be a variable [20]. Query statements are compared with RDF graph statements and the results of such comparison are returned for the user. SPARQL query consists of three main parts that are described in Table 1.

<table>
<thead>
<tr>
<th>Query part</th>
<th>Definition</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>Defines four different kinds of SPARQL queries. All of them use the same graph matching approach, but they differ in result presentation.</td>
<td>select - the most common query which returns variables and their bindings; ask – returns answer, if query pattern meets RDF graph. Only yes or no values can be returned; describe – returns all of the relations of the certain resource; construct – returns a single RDF graph specified by a graph template.</td>
</tr>
<tr>
<td>solution modifiers</td>
<td>Enable to modify information output for user.</td>
<td>projection - choose certain variables to output; distinct – remove duplicate solutions; order – sort results by specified variable. Asc and desc keywords can be used to define sort order; limit – specify maximum rows in a result set; offset – specify number of the first row in the result output.</td>
</tr>
<tr>
<td>pattern matching part</td>
<td>„where” part of the query to describe graph matching pattern. It consists of collection of triple patterns, which are compared with RDF graph.</td>
<td>optional – does not eliminate the solution, if optional part does not meet RDF graph; union – merge different result sets into one; filter – filter by data type values.</td>
</tr>
</tbody>
</table>
We will provide an example of transforming a SBVR question into SPARQL query under the process model, which is presented in Figure 1. First of all, we have to define a business vocabulary – terms, names, fact types and facts of the problem domain. We will not go deep into this process, as it is described in [14]. Here we will only present ontology, which is generated from business vocabulary to execute query in. Ontology is presented in Figure 6 as UML class diagram.

Figure 6. UML model of the problem domain for formulating Query2

The Query2 we will analyze is designed to get names of universities, names and impact factors of the journals, that are referred in the database ‘Web of Science’, have impact factor greater than 0, and are published by university that is in city Kaunas. Written in SBVR Structured English, this query will look like this:

What are university_names of universities, journal_names of journals and journal_impact_factors of the journals that are_referred_in the database 'Web_of_Science' and have an journal_impact_factor, which is_greater_than 0, and are_published_by the university that is_in city 'Kaunas'?

This query is based on facts and fact types:

- journals are_referred_in database 'Web_of_Science'
- journals have journal_impact_factor, which is_greater_than 0
- journals are_published_by university
- university is_in city 'Kaunas'

When a query is based on several facts it is difficult to analyze it, so predefined keywords as “that”, “which”, “and” are used. These keywords help to recognize object types and individual concepts of the question. Model of Query2 is based on the same principles as Query1, presented in Figure 5, but is much more complex.

For this question nominalization we must apply a projection 1 on three variables that range over the object types “names of universities”, “names of journals”, and “journal impact factors”. The projection 1 has an auxiliary variable 4 that is ranged over the object type “journal”. The variable 4 is restricted by a projecting formulation that is constrained by an atomic formulation based on the fact type “journal is in city”. The projecting formulation 2 has a projection on a variable 5 that ranges over the object type “journal” and has an auxiliary variable that is restricted by the instantiation formulation “Kaunas is city”. The projection 1 is on a variable 5 that ranges over a concept “proposition”. The proposition is meant by a closed logical formulation – conjunction of projecting formulations “journals are_referred_in database 'Web_of_Science', journals have journal_impact_factor, which is_greater_than 0, journals are_published_by university that is_in city 'Kaunas'. These projecting formulations are formulated similarly as in Figure 5.

The first step of designing SPARQL query is to define solution modifiers – ?university_name, ?journal_name and ?journal_impact_factor. These variables are defined by query fact types and pattern matching part is generated from facts. Transformations of example query are presented in tables Table 2 and Table 3.
### Table 2. Composing solution modifiers

<table>
<thead>
<tr>
<th>SBVR fact types</th>
<th>Solution modifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>university names of universities</td>
<td>?university_name</td>
</tr>
<tr>
<td>journal names of journals</td>
<td>?journal_name</td>
</tr>
<tr>
<td>journal impact factors of journals</td>
<td>?journal_impact_factor</td>
</tr>
</tbody>
</table>

### Table 3. Composing triple patterns

<table>
<thead>
<tr>
<th>SBVR facts</th>
<th>Triple patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>journal has journal impact factor, which is greater than 0</td>
<td>FILTER(?journal_impact_factor &gt; &quot;0&quot;^^xsd:double)</td>
</tr>
<tr>
<td>journal is published by university</td>
<td>?journals journal:isPublishedBy ?university</td>
</tr>
<tr>
<td>university is in city 'Kaunas'</td>
<td>?university journal:isIn ?city . ?city journal:name &quot;Kaunas&quot;^^xsd:string</td>
</tr>
</tbody>
</table>

After transforming SBVR into SPARQL, we get such SPARQL query:

```sparql
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX journals_ontology: <http://Data.JournalsOntology/>
SELECT ?university_name ?journal_name ?journal_impact_factor WHERE {
  ?journals journals_ontology:isReferredIn ?database .
  FILTER (?journal_impact_factor > "0"^^xsd:float) .
  ?journals journals_ontology:isPublishedBy ?university .
  ?university journals_ontology:isIn ?city .
  ?city journals_ontology:name "Kaunas"^^xsd:string .
  ?university journals_ontology:name ?university_name .
  ?journals journals_ontology:name ?journal_name .
}
```

Results of the Query 2 are presented in Figure 7.

<table>
<thead>
<tr>
<th>universityName</th>
<th>journalName</th>
<th>journalImpactFactor</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Ku&quot;</td>
<td>&quot;InformationTechnologyAndControl&quot;</td>
<td>&quot;0.495&quot;</td>
</tr>
<tr>
<td>&quot;Ku&quot;</td>
<td>&quot;ElectronicsAndElectricalEngineering&quot;</td>
<td>&quot;0.435&quot;</td>
</tr>
<tr>
<td>&quot;Lu&quot;</td>
<td>&quot;SaltyForestry&quot;</td>
<td>&quot;0.369&quot;</td>
</tr>
</tbody>
</table>

**Figure 7. Results of Query 2**

### 6 Conclusions and future works

Currently, the growing attention to SBVR seems most focused on engineering or verbalizing various software models and applications. Such attention is not paid to SBVR questions that are worth to be considered for solving some topical problems e.g. querying about business data and processes in a language understandable to business people.

Our research and implemented prototype in principle have shown the feasibility of transforming SBVR Structured English questions into SPARQL. Question transformations process starts from parsing it and analyzing related fact types and facts. The most difficult part is to recognize links among facts, so predefined keywords must be used. After fact types, facts and their links are discovered, a question in SBVR Structured English style text can be transformed into SBVR schema and later – into SPARQL solution modifiers and pattern matching parts.

Our future work will be concentrated on thorough analysis of various patterns for formulating SBVR questions and corresponding SPARQL patterns as well as integrating the obtained solution into the wider context where such questions could be executed for extracting answers from ontologies and underlying data sources.
References


