KAUNAS UNIVERSITY OF TECHNOLOGY

ALGIRDAS ĠUKYS

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

Doctoral dissertation
Technological sciences, Informatics Engineering (07T)

2017, Kaunas

This doctoral dissertation was prepared at Kaunas University of Technology, Faculty of Informatics, Department of Information Systems during the period of 2010ï 2017. The studies were supported by Research Council of Lithuania.

Scientific Supervisor:

Prof. Dr. Lina NEMURAITĂ (Kaunas University of Technology, Technological Sciences, Informatics Engineering, 07T).

Doctoral dissertation has been published in:
http://ktu.edu

Editors:
Antony Richard Bexon, Vilija Celieğienă.

ISBN 978-609-02-1336-0

The bibliographic information about the publication is available in the National Bibliographic Data Bank (NBDB) of the Martynas Mağvydas National Library of Lithuania

KAUNO TECHNOLOGIJOS UNIVERSITETAS

ALGIRDAS ĠUKYS

VEIKLOS ĢODYNO IR VEIKLOS TAISYKLIö SEMANTIKA GRINDĢIAMOS ONTOLOGIJö UĢKLAUSOS

Daktaro disertacija
Technologijos mokslai, informatikos inĝinerija (07T)

2017, Kaunas

Disertacija rengta 2010-2017 metais Kauno technologijos universiteto Informatikos fakultete Informacijos sistemÔ katedroje. Mokslinius tyrimus rǟmǟ Lietuvos mokslo taryba.

Mokslinǟ vadovǟ:

Prof. dr. Lina NEMURAITǞ (Kauno technologijos universitetas, technologijos mokslai, informatikos inģinerija, 07T).

Interneto svetainǟs, kurioje skelbiama disertacija, adresas:
http://ktu.edu

Redagavo:

Antony Richard Bexon, Vilija Celieģienǟ.

ISBN 978-609-02-1336-0

Leidinio bibliografinǟ informacija pateikiama Lietuvos nacionalinǟs Martyno Maģvydo bibliotekos Nacionalinǟs bibliografijos duomenÔ banke (NBDB)

ACKNOWLEDGEMENT

This research would not have been possible without the support of many people. First, I would like to thank my scientific supervisor Prof. dr. Lina Nemuraitė for her guidance, support, patience, and gained knowledge during the research and writing of the thesis.

I would also like to thank my reviewers for their insightful comments and advice, who helped me to correct and improve the dissertation. I am also thankful to all colleagues from the Department of Information Systems for their suggestions and advice, which I have received through all the years of research.

A special thanks to my family and friends for their continuous support and encouragement.

TABLE OF CONTENTS

TERMS AND ABBREVIATIONS

Term	Description
ANTLR	Another Tool For Language Recognition, is a parser generator that uses LL(*) parsing.
API	Application programming interface.
AST	Abstract Syntax Tree. In-memory representation (object graph) of any parsed text file (Xtext definition).
ATL	ATL Transformation Language is a model transformation language and Eclipse toolkit, used in the field of Model-Driven Engineering where ATL provides ways to produce a set of target models from a set of source models.
BPMN	Business Process Model and Notation. A graphical representation for specifying business processes in a business process model.
CNL	Controlled Natural Language. CNLs are subsets of natural languages, obtained by restricting the grammar and vocabulary in order to reduce or eliminate ambiguity and complexity.
EBNF	Extended Backus-Naur Form is a notation for formally describing syntax.
Effectiveness	Usability measurement criteria, typically evaluated in terms of precision and recall in the area of NLIs.
EMF	Eclipse Modelling Framework. A modelling framework and code generation facility for building tools and other applications based on a structured data model.
Habitability	Term, often used in context of natural language interfaces. It defines, how easy and naturally a user can express his thoughts using language restrictions.
HTML	Hypertext Markup Language is the standard markup language used to create web pages.
IRI	Internationalized Resource Identifier, sequence of characters from the Universal Character Set (Unicode / ISO 10646) used to identify resources.
Lexicon	Vocabulary of natural language interface, used to formulate, analyse, and transform questions to queries.
MDD	Model Driven Development
NLI	Natural Language Interface.

OCL	Object Constraint Language. A declarative language for describing rules that apply to UML models developed at IBM and now part of the UML standard.
OMG	Object Management Group, the consortium for the wide range of technology standards, originally aimed at setting standards for distributed object-oriented systems
OWL	Web Ontology Language, the family of knowledge representation languages for authoring ontologies, endorsed by W3C.
Portability	A feature of natural language interface, allowing to adjust (i.e., configure) it for questioning in different business domains.
QVT	Query/View/Transformation. A standard set of languages for model transformation defined by the OMG.
RDB	Relational database.
RDF	Resource Description Framework is a standard model for data interchange on the Web.
SBVR	Semantics of Business Vocabulary and Business Rules, is a publicly available specification from the OMG intended to be the basis for a formal and detailed natural language declarative description of business, its rules and policies
SBVR SLE	SBVR structured language editor
SPARQL	Simple Protocol and RDF Query Language ï a query language for the Semantic Web and ontologies.
SQL	Structured Query Language for managing data held in a relational database management system.
SSE	SBVR Structured English ï notation and language used for SBVR vocabulary and rules description.
SWRL	Semantic Web Rule Language. A proposed language for the Semantic Web that can be used to express rules.
UML	Unified Modelling Language is the OMG standardized general-purpose modelling language used in a very broad scope that covers a large and diverse set of application domains including the field of software engineering and object-oriented software-intensive systems.
Usability	Term, defining the quality of the appropriateness to a purpose of any particular artefact.

W3C	World Wide Web Consortium is the main international standards organization for the World Wide Web.
XMI	XML Metadata Interchange is an OMG standard for exchanging metadata information via Extensible Markup Language.
XML	Extensible Markup Language ï a markup language, that defines a set of rules for encoding documents in format, which is human- and machine-readable.
XSD	XML Schema Definition. A recommendation of the W3C, which specifies how to describe elements formally in an XML document.

FIGURES

TABLES

INTRODUCTION

The amount of information on the Web grows constantly nowadays. Information overload makes a Web search process tedious. Traditional keyword based search engines analyse HTML documents that are intended to render information for humans but does not represent semantics, which a computer can understand. Even though such search engines help to find information, they give redundant or incomplete results based on keyword matches, leaving a lot of work for users to find relevant information. For example, it would be a difficult task to find all heads of the European Union states using keyword based search. The user would have to put additional effort into completing this search.

The Semantic Web idea [4] is based on understanding the meaning of published information and processing it by machines. The backbone of a Semantic Web is ontologies that store entities, representing real world objects (i.e., persons, vehicles, organizations), their relations, properties, etc. The search across ontologies is called semantic search. Due to the capability to understand the intent of the userôs queries and even complex questions, semantic search returns results that are more precise.

One of the challenges of developing a system with a semantic search function is the implementation of a usable and convenient user interface. A number of interfaces to ontologies were introduced after the Semantic Web idea spread: Semantic Crystal [54], Ginseng [6], QuestIO [22], FREyA [19], ORAKEL [12], PANTO [129], Querix [56], etc. They vary from simple interfaces for SPARQL queries to more sophisticated natural language interfaces (NLIs) and differ in their usability. The study of E. Kaufmann and A. Bernstein [54] was carried out to compare keyword-based search, graphical query language, natural language and menu guided interfaces. It was found, that users prefer querying ontologies using full sentences in natural language. The research revealed the potential of NLIs for end-user access to the Semantic Web, as this type of interface proved the most useful and best-liked query interface.

1.1 Motivation

In this work, it was decided to create a new NLI. The first reason was the desire to write questions in multiple languages. It is important, because 25.9% of internet users use the English language, while the other users use other languages [83]. Existing NLIs show good results answering questions in English. However, authors do not discuss about adapting their solutions for other languages, i.e., which components are independent from language, and which should be replaced or adjusted, what source code modifications are required. Certainly, simple replacement of standard linguistic libraries (e.g., Stanford parser, WordNet, etc.) would not be enough, it would require a significant source code modification.

Another reason is about mapping questions with ontology resources (i.e., classes, properties, etc.). This is a critical function of NLI, required for translating questions to queries. Ontologies in the Semantic Web are processed and understood by machines. The problem is that their structure can differ from how people think about data and formulate questions. It is obvious, that people desire writing simple questions, while data in ontologies can be stored using complex structures (e.g., using n-ary relations). Therefore, straightforward mappings (i.e., question to a single ontology resource) is not enough, NLI must be able to perform complex mappings (i.e., question to a combination of resources). The analysis of existing NLIs to ontologies revealed that most of them extract lexicon directly from ontologies. As a result, they allow only straightforward mapping and understand only those questions that correspond to the structure of the ontology.

Therefore, the basic principles of the systemôs architecture was formulated: NLI must be adjustable for different languages and the lexicon must allow relating complex ontological constructions with simple questions. It was decided to use SBVR in order to achieve this. This standard is intended to specify business vocabularies and business rules using structured natural language. The foundation of SBVR is a semiotic/semantic triangle, which is the theoretical basis for SBVRôs linguistic based architecture that separates expression from meaning [98]. It allows the expressing of the same things differently as well as in different languages. Therefore, a question, written in different languages, has the same model of meaning and can be transformed to an ontology query regardless of the language it is written in.

SBVR vocabulary concepts can have definitions given as rules that describe derivations of those concepts. Such definitions formally specify the derivation of concepts from other concepts and can support inferences [64]. This suggests that SBVR definitions could be used to bridge the gap between the way in which a particular item of data is stored (i.e., the ontology scheme) and the way of, how a user thinks about the data and formulates questions.

Although the SBVR metamodel supports questions and allows querying software models, it was not previously used for semantic search. This work should answer, whether or not SBVR can be used as a basis for NLI, which is multilingual and allows mapping simple questions with complex ontology structures (i.e., combinations of ontology resources).

1.2 Object and scope of research

The object of this research is a process of querying ontologies using natural language questions. The scope of the research includes the following topics:

- Natural language interfaces to knowledge bases;
- Most advanced knowledge and data models, their representation and query languages (SBVR, OWL 2, RDF, SPARQL), related tools and technologies;
- Model driven transformation technologies.

1.3 Problem statement and research questions

The lack of usable and convenient user interfaces to ontologies, allowing questioning in natural languages ï is the problem inspired by this research. When solving this problem it was important to fulfil such requirements:

- Adjustability to questioning in different languages (i.e., languages, investigated in this work: English and Lithuanian; and grammatically similar languages: German, Czech, Polish, etc.);

- Ability to map questions with combinations of ontology resources;
- Portability (i.e., ability to question in different domains);
- Effectiveness of answering questions similar to other NLIs.

This research intends to answer the following questions:

1. Is it possible to use SBVR questions for querying ontologies and relating natural language questions with combinations of ontology resources?
2. How natural language questions can be transformed to SPARQL using SBVR?
3. Is it possible to achieve portability without compromising the correctness of NLI to ontologies using SBVR?
4. Can SBVR based NLI to ontologies be adjusted to different languages and what components are language specific?

1.4 Goal and objectives

The main goal of this work is to extend semantic search capabilities, allowing users to write natural language questions in different languages, also including such cases when mapping of questions to ontology is complex (i.e., questions must be mapped with combinations of ontology resources). Research tasks are the following:

1. To analyse literature related with OWL ontologies and ontology query language SPARQL; existing NLIs to databases and ontologies; SBVR knowledge model and tools to write SBVR specifications;
2. To define the conception of NLI to ontologies and algorithms for transforming natural language questions to SPARQL queries;
3. To define the conception of a SBVR tool for writing business vocabularies, rules, and questions;
4. To create prototypes for evaluating the relevance of the solution;
5. To conduct an experiment and evaluate research results.

The scope of this work can solve just a limited set of problems, related with natural language questions, sufficient for proving the concept. The main quality criteria for the solution are as follows:

- Ability to question ontologies in different languages;
- Ability to map questions with combinations of ontology resources;
- Portability of the solution;
- Effectiveness of answering questions.

1.5 Research methodology

The research was carried out using the methodology of Design Science (also called constructive) research. This paradigm seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artefact [45]. This artefact can be algorithm, framework, model, etc. In this research, it is Semantic search solution. The methodology and steps of the research are presented in Table 1.

Table 1. The research methodology

1. Selecting a practically relevant research problem.	It was found that the process of traditional keyword based search leaves a lot of manual work for users to find relevant information from results of keyword matching.
2. Analysing existing solutions to find out the potential for the research.	The search process can be facilitated performing semantic search over ontologies. The most convenient interface for ontologies is NLI. The analysis of existing NLIs to ontologies showed that existing solutions have limitations. Therefore, it was decided to create a new Semantic search solution.
3. Analysing the domain to understand the problem and create the solution to solve it.	First, ontologies and ontology query languages were analysed to understand, how ontologies are modelled and quered. To understand the area of creating NLIs (e.g., problems that are faced creating and using such systems, their main features, requirements, etc.), research, related with NLIs to databases and ontologies were analysed. Finally, having an insight to solve the shortcomings of existing NLIs, SBVR standard and capabilities to use it for Semantic search solution were analysed.
4. Creating the original solution	The conception of SBVR based Semantic search solution was created. The solution consist of NLI to ontologies and SBVR structured language editor (SBVR SLE). The solution was theoretically described by defining rules to transform natural language questions to SPARQL queries and grammar for creating SBVR structured language editor.
5. Implementing the prototype of the solution and evaluating it	Two prototypes were implemented: SBVR SLE and NLI to ontologies. Experiments were conducted for evaluating the applicability of the solution and comparing it against other similar solutions.
6. Feasibility to apply the solution in practice	The created solution can be applied to implement the semantic search in the

	Web. It is also expected that the created SBVR SLE will create conditions for other SBVR related research.
7. Relations of the solution with theorethical studies	The created solution will complement the set of available solutions of NLIs to ontologies. This work also contributes to the research of SBVR and presents the applicability of this standard for querying.

1.6 Defended propositions

Propositions defended by this thesis are the following:

1. SBVR allows using language independent rules to transform questions to semantic queries and achieve multilingualism of NLI to ontologies. However, such a solution also requires language dependent components to perform syntactic and morphological analysis of questions.
2. SBVR derivation rules allows describing the relations between natural language questions and complex ontology structure. As a result, ontologies can be queried written questions that do not directly correspond to their structure.
3. SBVR based NLI is portable. Portability is achieved specifying business vocabulary and rules (i.e. lexicon) of certain domain and mapping it with ontology. Mappings are performed labelling ontology resources with representations of corresponding vocabulary concepts.

1.7 Major contributions and novelty

The major contribution of this work is the solution of querying ontologies using natural language. The solution consists of the following two parts:

- NLI to ontologies, which allows writing, analysing natural language questions and transforming them to SPARQL queries.
- The SBVR SLE that is used for configuration of NLI. It allows specifying business vocabularies, business rules and writing questions using structured language. In addition, specifications can be transformed to the SBVR XMI model.

The novelty of the research is as follows:

1. It was not found in any research, published by other authors, the using SBVR questions for querying ontologies.
2. The created NLI to ontologies has clearly defined parts, which have to be replaced for querying in different languages.
3. The created NLI to ontologies allows questioning when the structure of ontologies is complex and it is needed to map questions with combinations of ontology resources. These mapping are defined using SBVR derivation rules or formal definitions of SBVR concepts.
4. The created SBVR SLE is used for writing specifications of business vocabularies and business rules to formulate questions. This tool allows

the splitting of specifications into separate parts and hide some vocabulary entries (i.e., those that are used only for derivations) from user. In addition, this feature allows the creating of metavocabularies and applying editor for other purposes (e.g., transformation to OWL 2, specification of BPMN business processes, etc.).

1.8 Practical significance

The presented solution offers a way to implement semantic search writing questions in natural language. It can be configured to questions for ontologies of different domains and in different languages. The ability to define SBVR derivations allows the answering of simple questions in complex ontologies that are often used in practice. The solution can be applied for a semantic search on the Web or in business applications.

The results of this work were applied in the SemantikaLT project [99], creating a semantic search service that allows writing questions in the Lithuanian language. The SBVR SLE was used for creating SBVR business vocabularies, business rules and configuration of NLI. The transformations, created in this work, were used to transform Lithuanian questions to SPARQL for querying semantically annotated Lithuanian Internet corpora for Politics, Business and Economy, and Public Administration domains.

The created SBVR SLE can be used to other research, related with SBVR to create specifications of business vocabulary and business rules.

1.9 Scientific approval

Results of this research were presented at five international conferences and one Lithuanian conference. Two articles were published in scientific journals referred in the Thomson Reuters ñISI Web of Scienceò Master Journal List with impact factor. Two articles were published in publications that are referred in the Thomson Reuters ñISI Web of Scienceò Conference Proceedings. Four articles were published in other scientific publications ï proceedings of the conference. The detailed list of publications is presented in section seven.

1.10 Thesis structure

The second section introduces ontologies and the analysis of NLIs, including challenges and requirements of creating such systems. This section also introduces SBVR specification and SPARQL query language. In the third section, the Semantic search solution, including models and algorithms created in this work, is presented. The fourth section demonstrates details of implementing prototypes. The fifth section is dedicated to experimental evaluation of the solution. The sixth section presents conclusions. Seventh and eighth sections present references to the literature and a list of authorôs publications on the dissertation theme.

2 ANALYSIS OF NATURAL LANGUAGE INTERFACES AND MOST ADVANCED KNOWLEDGE MODELS

The analysis of related scientific literature contains two parts. In the first part, the definition of ontology is introduced. Then, the analysis of NLIs describes; how these systems evolved, what main obstacles are encountered creating and using them, what are the main requirements of NLIs, etc. Finally, a comparative analysis of existing NLIs to ontologies is presented.

In the second part, a SBVR knowledge model and its capabilities for using as a basis for NLI to ontologies are analysed. This part also contains the analysis of SPARQL query language.

2.1 Ontologies and natural language interfaces

2.1.1 What is ontology

The term *ontology* came from Greek word *onto* (being) and *logia* (science). It came from the discipline of philosophy ï *metaphysics*, dealing with the nature and the organization of reality. This discipline tries to answer questions, such as ***ñ**What are meanings of being?**ò**, **ñ**Into what categories, if any, can we sort existing things?**ò***, etc. The traditional goal of ontological inquiry is discovering fundamental categories or kinds into which the worldôs objects fall [3]. From here, the main concepts of ontology came, i.e., kinds, properties, attributes, relations, parts and wholes, and processes. This traditional understanding of ontology has many examples in natural and abstract sciences ï physics, chemistry, biology, mathematics. For example, in biology; the purpose of ontology is to classify living organisms into kingdoms, phylums, classes, etc. In mathematics, ontology is used to classify theoretical objects. For instance, in number theory; natural numbers are classified into prime and composite numbers. In geometry, triangles are classified by their angles and sides.

In computer and information science, ontology is a technical term that means an artefact, which allows modelling knowledge of some domain, which can be real or imagined [35]. The term was adapted by Artificial Intelligence researchers and in the 1980ós this term started to refer to both a theory of a modelled world and knowledge systems [35].

The essential definition of ontology came from 1993, when Tom Gruber defined the term *ontology* as an *explicit specification of conceptualizations* [34]. That is, ontology is objects, concepts, and other entities that are presumed to exist in some area of interest and the relationships that hold among them [35]. This definition has two essential points [35]: ontology defines concepts, relationships, and other features that are relevant for modelling a domain; specification defines the form of ontology (i.e., it can contain classes, data properties, etc.). The definition also claims that all concepts and their features should be explicitly stated. This can be done in two ways [37]: extensional, when all concepts and their features are listed or intentional, when conceptualization is specified by constraining interpretations using axioms (e.g., transitive, symmetric properties, etc.). However, this definition is broad, allowing a range of specifications from simple glossaries to logical theories couched in predicate calculus [102]. Therefore, researchers sort to clarify the definition of ontology.

In 1997, Borst defined an ontology as a *formal specification of a shared conceptualization* [8]. This definition is similar, but emphasizes the fact, that there must be an agreement about conceptualization to allow reusing ontology and general acceptance. The specification should also have formal representation to be processed by machines. Therefore, natural language texts do not satisfy this definition.

In 1998, Studer et al. [15] merged these two definitions stating that ontology is *a formal, explicit specification of a shared conceptualization*.

In 1998, Guarino clarified the original definition of ontology by formalizing it [36] [37].

In 1999, Smith and Welty [102] proposed the classification of ontologies. Different information artefacts were classified as ontology and all of them satisfy Gruberớs definition:

- Simple catalogue of products;
- A set of natural language texts;
- Glossary of terms and their natural language descriptions;
- Thesaurus, consisting of terms, formatting common hierarchy;
- Taxonomies with property inheritance from more general to more specific classes;
- Frame-based systems, having a taxonomic structure and relations between objects and restrictions, how objects can relate one with another;
- Ontologies, using axioms of first order, higher order, or modal logic ï this type of ontologies is the most complex and expressive.

Hence, ontologies can be represented in different ways. Consider ontology that stores knowledge about students, taking exams, and teachers, who organizes exams. Teachers can be also specialized as lecturers, associate professors, and professors. The graphical representation of this ontology is presented in Figure 2.1.

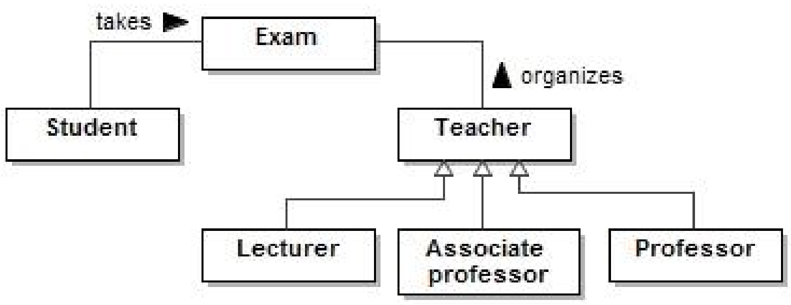


Figure 2.1. Graphical representation of ontology

Figure 2.2 presents an example of individual in this ontology (i.e., students taking the exam, which is organized by teacher).

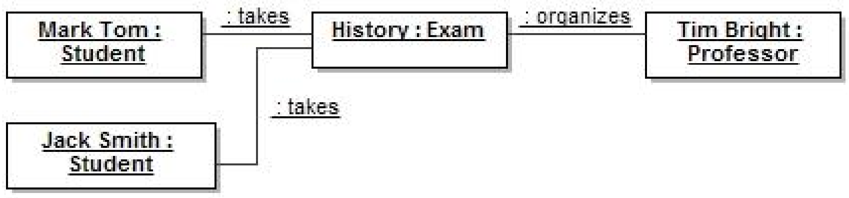


Figure 2.2. Example of individuals in the ontology

Besides explicit knowledge that is stored in ontologies, implicit knowledge can also be found. In this example, the fact that teachers examine students is not explicitly stated. However, it can be derived from other facts (i.e., students take exams and exams are organized by teachers).

Ontology must have a machine processable format to be able to use it as an engineering artefact. The most popular language for defining and instantiating ontologies is Web Ontology Language, OWL. It includes descriptions of classes, properties and their instances [101]. OWL ontologies are based on the RDF data model. It is considered the most relevant standard for data representation and exchange on the Semantic Web [42]. The RDF data model is based on statements (i.e., triples) that are expressed in the form of subject, predicate and object. A set of statements compose the directed labelled graph. RDF Schema is a semantic extension of RDF. It allows defining classes and properties of RDF resources, semantics of generalizations of classes and properties.

OWL is a set family of three increasingly expressive sublanguages, which can be chosen according to the needs:

- OWL Lite. Provides all semantics of RDF Schema, plus simple constraints, such as cardinalities, equalities, property characteristics, etc. It also allows importing ontologies.
- OWL DL. Supports maximum expressiveness with computational completeness and computing in finite time. It is named so due to the correspondence with description logics.
- OWL Full. Provides richer expressiveness, but it is not handled by software tools, because no computational guarantees are provided.

OWL ontology can be serialised using various syntaxes (e.g., RDF/XML, OWL/XML, Turtle, etc.) for storing or exchanging ontology among tools and applications [89]. The primary syntax is RDF/XML. However, OWL/XML is easier processable by XML tools, functional syntax simplifies reading of formal structures, and Turtle syntax represents the triple based nature of OWL ontologies more clearly and represent it as a set of triples. The example ontology of Turtle syntax is presented in Table 2.1.

Table 2.1. Ontology serialization using Turtle syntax

```
:Student rdf:type owl:Class .
:Teacher rdf:type owl:Class .
:Exam rdf:type owl:Class .
:Lecturer rdf:type owl:Class ;
          rdfs:subClassOf :Teacher .
```

```
:Professor rdf:type owl:Class ;
           rdfs:subClassOf :Teacher .
:Associate_professor rdf:type owl:Class ;
           rdfs:subClassOf :Teacher .
:organizes rdf:type owl:ObjectProperty ;
           rdfs:range :Exam ;
           rdfs:domain :Teacher .
:takes rdf:type owl:ObjectProperty ;
       rdfs:range :Exam ;
       rdfs:domain :Student .
```

In this work, ontologies are treated as an engineering artefact and used to perform a semantic search on the Web. According to the dependence classification of N. Guarino [38], domain ontologies (i.e., describing the vocabulary related to a specific domain) are considered. Ontologies are defined using OWL language (precisely OWL 2, a latest version of OWL). By semantic expressiveness, OWL Lite or OWL DL sublanguages can be selected.

2.1.2 Introduction to natural language interfaces

The simplest user interfaces allows to perform semantic search over ontologies writing queries (e.g., SPARQL [92] [43]). However, dedicated query languages are complex and unfamiliar for users. To allow querying ontologies - more convenient, different types of interfaces were created: NLIs, KIM form-based interface [91], faceted search, where knowledge is grouped and represented through taxonomies [19], graphical tools, menu-guided, and keyword-based, etc. Although these tools hide the complexity of underlying query languages, they still require the user to be familiar with the structure of queried knowledge [19]. The usability research of E. Kaufmann and A. Bernstein [54] has shown that users prefer querying ontologies using natural language sentences. This study showed the potential of NLIs for end-user access to the Semantic Web.

The first research of NLIs were started in the early 1960ś [79]. Their goal was to simplify the search process in databases and reduce the learning time to work with them. NLI was thought to be a promising interface, allowing users to communicate with information systems and extract the required data using natural instead of specific querying or programming language. Four main reasons were distinguished for using NLI [73]:

- No need to learn special purpose languages;
- The range of database queries that can be formulated in natural languages is potentially the same as in any formal query language;
- Possibility to query about the domain structure;
- Possibility to refine queries using dialogues.

In general, NLIs have the following components: linguistic component, lexicon, and data storage component. Linguistic component is used to parse natural language questions and transform parse trees into queries. This component also formulates answers to questions. Data storage component stores data (e.g., relational database, ontology, etc.) and lexicon is used to link syntax elements of question with structures of data storage. It helps to find objects (e.g., tables, attributes, relations, etc.) corresponding to certain natural language formulations. Lexicon can also be used for answer generation or formulating questions.

The key task of NLI systems is a bridge between two views: the way in which that data is stored (i.e., the knowledge base view) and the userǿs view ï the way how he or she thinks about data. The problem is that data is stored using a strict model, while a user thinks about it in a more abstract way, using real world knowledge. Someone has to ñteachò the system to identify available words of questioning in certain domains and determine the relations of those words with database structures [1]. This process is called configuration or customization. It is not a straightforward task and requires extensive efforts of natural language processing specialists and knowledge experts. The first of the configurable NLIs were configured by hand [33], therefore the economics of configuring them was one of the factors that blocked the uptake of NLIs [73]. Another reason was linguistic and syntactic problems that aroused processing natural language questions. Despite that, many efforts were made to create a commercially successful NLI. In the middle of 1980ǿs, when NLI research reached its peak, it was predicted that this type of interface will be one of the alternatives for working with information systems. Unfortunately, after some time this area has been abandoned for previously mentioned difficulties that prevented from reaching practical success of NLIs. Form based graphical interfaces were sufficient and more attractive for organizations, because they had no difficulties, which are inevitable using NLIs. However, a number of NLIs to databases were created for working in limited domains [84].

2.1.3 Natural language interfaces to databases

Research of NLIs to databases started in the late sixties and early seventies in the context of research into artificial intelligence. Early systems have been developed for certain domains and the portability was not important. One of the first systems was LUNAR [133]. It was created to query the database for chemical analysis of lunar rocks and facilitate the search process for scientists. Data from different scientific articles were stored in a single repository. The purpose was not a simple search of articles, but understanding and analysing data that was published: counting averages, ratios, comparing results of different scientists, etc. Authors decided to use this type of interface for several reasons:

- People use natural language in everyday life, so it is not necessary to learn any computer language to work with the system. It was important, because many people were not computer literate in those days;
- People think using terms of natural languages. Translation of ideas and thoughts to specific computer languages takes time and aggravates communication between people and computers.

LUNAR allowed formulating questions in a flexible way, using abbreviations and synonyms. They were automatically translated into terms, corresponding to the database structure. LUNAR used the advanced language analysis system. First, syntactic analysis produced a parse tree. Following, semantic analysis was performed to identify meaning of elements of the parse tree using derivation rules. Finally, the result of the semantic analysis was used to generate a query. The following example question of the LUNAR system is presented [133]: *What is the average concentration of aluminium in high alkali rocks?*

A number of later systems used semantic grammars. This kind of architecture allowed analysing natural language questions easier. RENDEZVOUS accepted relatively unrestricted natural language questions and used clarification dialogs when having difficulties with parsing user input. It also used query paraphrasing which allowed reformulating a question if it was interpreted incorrectly [72]. LADDER interface was designed to work with large databases [44]. PLANES was created to answer questions, related to airplane maintenance and flight history [128]. However, when portable systems started to be create, semantic grammars were abandoned.

In the early eighties, one of the most prominent NLI to databases was CHAT-80 [130]. It used intermediate question representation as Prolog expressions that were executed in a Prolog database. This system was widely used. Additionally, it became a base for other experimental systems, for example, MASQUE.

In the mid-eighties, much attention has been paid to solving the portability problem. For example, the TEAM interface [33] was designed to be easily configurable by database administrators without any specific knowledge. ASK [117], [116] had expandable vocabulary that could be filled by users. This NLI had an internal database, but could also connect to external databases and applications, communicating with them using the same interface and hiding the complexity of those systems. JANUS [93] also had a similar interface with external systems, integrating them and providing a single interface to question their data. JANUS was one of the few systems, supporting temporal questions (questions with specifying time). Another system of the mid-eighties was STEP. The important feature of this system was the feedback mechanism, based on the intermediate representation of the question. It allowed users to check, whether the system understood the question correctly [72]. Some of the other interfaces that appeared in the mid-eighties were DATALOG, EUFID, LDC, TQA, TELI, etc. Commercial NLIs were also developed (e.g., INTELLECT, Symantec Q&A, Natural Language of Natural Language Inc., etc.). However, they were not commercially successful as was expected and interest in this type of interface started to decline. Therefore, this area of research was abandoned.

Research on NLIs were resumed when the Semantic Web idea spread. However, most of the challenges remained the same for NLIs to ontologies.

2.1.4 Challenges of developing natural language interfaces

Development of NLI is inseparable from the challenges that are centred on [20]:

- Understanding the natural language;
- Understanding the data, which is being questioned;
- The way, in which userôs information needs are verbalised into questions.

Understanding natural language is challenging, because natural language is complex and ambiguous. There are morphologically rich languages that makes them even harder to understand. In general, the natural language understanding problem includes the syntactic analysis, solving the ambiguity and expressiveness of the language. Methods of dealing with these problems are presented in subsection 2.1.9.

The important feature of NLI is the ability to relate user thoughts and their expressions in natural language with the data storage view. Knowledge bases have different structures and also the same things can be stored differently. For example, the number of countries that river flows through can be stored as data property or rivers can be connected with countries using object property. In addition, some words might have different meanings in two different domains or contexts. For example, *How the word big* can refer to *height*, *length*, *area*, or *population* ï depending on the question context. Therefore, a NLI needs to have a configuration mechanism to relate natural language phrases with structures of knowledge base and ensure portability.

Another problem is that knowledge bases store data in optimized structures, which differs from how people formulate questions. For example, n-ary relations are a natural and convenient way to represent meaning in the Semantic Web [82]. Consider ontology, containing classes *person* and *organization* with object property *works_in*. It allows questioning for persons working in organizations. However, if one wants to store the period when a person worked in a certain organization, a n-ary relation with intermediate class is required and the structure of underlying ontology becomes different from the language formulations of the question. To avoid this problem, the configuration mechanism of NLI must be able to relate simple phrases of natural language with complex structures of knowledge bases.

Another group of challenges is focused on users and translation of their information needs to questions. NLIs have only one textual query box that can pose difficulties for users who need to express their information needs through natural language queries. In order to address this problem, several usability enhancement methods have been developed. These methods either assist users with query formulation, or communicate with the user by asking them to confirm the interpretation of the questions. The purpose of such methods is to increase the habitability of the system [20]. These methods are overviewed in section 2.1.9.

2.1.5 Questions in natural language interfaces

Before discussing the main syntactic problems of understanding natural language questions, an introduction of the most commonly used groups of questions. These groups were identified working on project [99] and analysing question sets of Mooney Natural Language Learning Data [75].

The simplest questions retrieve objects of a certain type, for example, *Find organizations*. Another group contains questions with very similar syntax, but their meaning is changed using superlative adjectives, for example, *Find largest organization*. It is more difficult to answer these questions. The system needs to know, what criteria should be used to answer them and avoid a nominal compound problem (e.g., largest by sales, number of employees, etc.) and perform calculations.

Questions with modifier attachments belongs to another group of questions. The modifier attachment phenomenon occurs, when modifiers are used to modify meaning of syntactic elements (i.e., constituents). For example, in questions *Find persons that work in organizations* and *Find persons that work in KTU,* the meaning of *person* is modified by retrieving only those persons that work in some or specific organization. Questions can have more than one modifier (e.g., *Find persons that work in organizations that are located in Kaunas*). Questions with modifier attachments can be additionally restricted using cardinalities, for example, *Find persons that work in at least two organizations.*

Another group contains questions using count functions. Their syntax is very similar to the syntax of questions, using modifier attachment (e.g., *How many persons work in KTU*). These questions are distinguished by counting keywords (e.g., *How many*, *What is the number of*, etc.).

It should be noted, that NLI should also answer combined questions, for example, *How many large organizations are in Kaunas.*

Certainly, everyday language contains many more constructions of questions. The presented groups are the most commonly used working with NLIs and is considered in this work. More complex questions, such as ones, using conjunctions or disjunctions, are more difficult to interpret and answer correctly. In addition, it is unlikely, that internet users will be happy writing long and complex questions.

2.1.6 Understanding natural language questions

In this subsection, it is presented the most common linguistic problems of NLIs and the natural language understanding in general. It also presents possible solutions of how to cope with such problems.

The overview starts with elliptical questions. These questions do not have all the required words, but they can be recovered from the context. The example of the elliptical question is presented following [1]:

- *Who is the manager of the largest department?*
- *John.*
- *The smallest department?*
- *Jack.*

Elliptical questions are interactive: after the most comprehensive question, the shorter elliptical questions are submitted. Users often want to submit a group of similar questions. In this way, long and repetitive questions are avoided. To implement such a solution, a robust discourse analysis mechanism is needed. The easiest way to avoid a problem of long and repetitive questions is to allow users to edit and resubmit previous questions [1].

Anaphora is another difficulty that is faced when analysing natural language questions. In linguistics, it is a phenomenon when pronouns (e.g., she, he, they, etc.), possessive determiners (e.g., her, his, their), or noun phrases (e.g., these people) are used for implicitly denoting entities mentioned in the discourse [1]. Anaphora is a common problem in natural language processing. Anaphoric expressions make questions more natural, but these expressions must be resolved by identifying denoted concepts, called antecedents. The example of a question with anaphora is as follows:

- *Find city where John lives.*
- *Kaunas.*
- *What organization does he work?*

In the second question, the pronoun *he* means person *John*. The easiest way to resolve anaphoras is to keep a list of all available concepts in the discourse and select the most recently mentioned concept, which corresponds to syntactic and semantic constraints of the anaphoric object and its antecedent [1].

In everyday language, people make grammatical mistakes. Nevertheless, other people usually correctly understand what they mean. Although NLIs are foremost designed to understand correctly formulated questions, it is desirable to understand ill-formed texts (or at least partially), especially in cases when mistakes are not critical for correct interpretation of questions (e.g., missing articles, keyword mistakes, disagreement between cases or numbers, capitalization, or punctuation errors, etc.). The lexicon of NLI can be used for finding or automatically correcting mistakes.

Another linguistic problem is related with questions, using modifier attachments. Sometimes it is not easy to determine, which constituent is modified. Consider a question, taken from [1]: *List all employees in the company with a driving licence.* For people it is obvious, that modifier *with a driving licence* refers to *employees*. However, this question is ambiguous for a computer, because it can also refer the modifier to *company*. To resolve such ambiguities, some NLIs use heuristic rules. For example, in [26] it is assumed that a relative clause modifies the rightmost available constituent. Another solution is using clarification dialog, where the user is asked to determine the correct interpretation.

Another linguistic problem is related with interpretation of determiners (e.g., *a, each, some, every, several,* etc.). They are usually interpreted as logic quantifiers. However, the interpretation is not always obvious. Consider the question, taken from [60]: *Which dish did every boy make?* It is assumed that the quantified noun phrases in questions can be interpreted in three different ways: the narrow-scope reading (i.e., *every boy made pasta*); the functional reading (i.e., *every boy made his favourite dish*), and the pair-list reading (i.e., *All made pasta*, *Bill made salad*, and *Carl made pudding*). However, some quantifiers do not allow pair-list reading (e.g., *Which dish did most/several/a few/no boys make?*). In addition, some questions do not have pair-list interpretations (e.g., *Which boy made every dish?*) [60].

The linguistic problem of determining quantifier scope arises when multiple quantifiers are used in a single question. It leads to an ambiguous situation when it is not easy to determine which quantifier should be given to the wider scope. Consider the question *Has every student taken some course?* [1]. This question has two different interpretations:

∀student ∃course taken (student, course)
∃course ∀student taken (student, course)

The first interpretation means that each student has taken any course. The second interpretation means that each student has taken the same course. For resolving such ambiguities, the heuristic method is proposed in [90]. It suggests that quantifiers preserve the same order as for the determiner in the question.

The conjunction can also cause ambiguities that are difficult to resolve. In some situations, the word *and* does not necessarily have the meaning of conjunction. Consider the question *List applicants that live in California and Arizona* [115]. In this question, the word *and* is used, but the question should be interpreted using disjunction to get persons from either *California* or *Arizona*.

The nominal compound problem arises when a noun is modified with another noun or adjective creating an ambiguous meaning. For example, the compound phrase *major river* can be interpreted in a number of ways: a river that traverses through a certain number of countries, a river of a certain length or width, etc. Another example *large company* can be understood as a company with high sales or a company that employs a large number of employees, etc. If the system contains only data about sales of companies, this phrase can be easily disambiguated. However, in some situations it is hard to determine the right interpretation. To address this problem, some systems require defining each possible noun-noun or adjective-noun compound in a configuration phrase [1].

2.1.7 Portability of natural language interfaces

Portability is the desired feature of NLIs and is considered as one of the most challenging problems of developing a NLI [13]. This term was defined when NLIs to databases were being developed. It is considered, that NLI to database is knowledge-domain portable, if it can be configured for use in a wide variety of knowledge domains [1]. However, many NLIs to databases were not portable and were created to work only in a particular domain. The portability problem remains relevant for NLIs to ontologies too. In addition, ontologies has a flexible data model, which is changing and evolving over time and therefore the NLI should be adaptable to these changes.

The most important issue here is to have a mechanism for specifying, how natural language words and formulations map to structures in the knowledge base [13]. Portability is closely related with correctness. Correctness is the criterion of how correct results are and is expressed using precision and recall parameters. It is always trying to keep the balance between these two features. If a system is portable with little configuration efforts, the correctness is often worse and vice versa, if a system is more tied with a certain domain and is not portable, it returns more correct results. Easily portable systems, such as Querix [56], QuestIO [22], or NLP-Reduce [55] require no or little configuration efforts. However, additional configuration can improve the correctness [30].

A person, responsible for configuring NLI, creates a lexicon of a particular domain and map linguistic formulations with the knowledge base [22]. This work is often carried out by domain experts. A configuration can be time expensive, so it is desirable to reduce the amount of effort and make a NLI easier to be portable to different domains. Therefore, some systems use techniques to achieve portability without additional efforts. For example, in QuestIO or FREyA [19], knowledge is extracted from ontologies and lexicon is created automatically. Certainly, the correctness depends on the quality of lexical information (e.g., labels, descriptions, etc.) available in the ontology [22].

2.1.8 Habitability of natural language interfaces

Habitability is a term, proposed by Watt in 1968 [132]. This term is often used in context of NLIs. It defines how easy and naturally a user can express his thoughts using language restrictions. For example, if a certain question can be expressed in five ways, the habitable NLI should understand each of them. Misalignment of system capabilities and user expectations is called the habitability problem [118].

According to Epstein [26], a language is habitable if users are able to construct expressions of the language, which they have not previously encountered, without significant conscious effort. In addition, users are able to avoid constructing expressions that are not part of the language.

Ogden and Bernick [84] describe the habitability in the context of four domains: conceptual domain, functional domain, syntactic domain and lexical domain. NLIs try to cover each domain by meeting expectations of the users. The habitability is determined by how well these expectations are met.

The conceptual domain means the conceptual coverage of NLI. It means that the interface can understand only those concepts that have relations with data stored in the knowledge base. For instance, one cannot formulate questions to find persons and organizations they work in, if there is no information about that. However, the conceptual coverage of natural language and knowledge base can disagree. If conceptual coverage of the knowledge base is wider than the conceptual coverage of natural language, some information will not be found. Otherwise, users will be able to formulate questions that will not be answered. The system should react properly in such cases and inform users that certain question could not be answered (e.g., *Information about persons and their organization is not stored*). To sum up, the conceptual domain determines *what* can be queried by the system [20].

The functional domain is determined by a number of functions and knowledge that a system has for searching and inferring information. Same questions can be formulated in many ways, depending on knowledge and experience of the users. This domain defines *how* questions can be formulated. Consider the following example: *What is a salary of Johnŝ Smithŝ manager?* The procedure of getting the answer to this question is complicated and if the system is not sophisticated enough, this question cannot be answered. However, it can be formulated in a different way by writing two questions [84]:

- *Who is the manager of John Smith?*
- *Mary Jones.*
- *What is the salary of Mary Jones?*
-*1200$*

A habitable system should also provide functions that the user expects from the natural language interface: counting, comparison, search for minimum and maximum values, etc. This type of questioning should be determined automatically by specific words and phrases (e.g., *largest*, *smallest*, *greater than*, *how many*, etc.). Consider the following question: *How many people live in France?* The habitable system should properly understand it and use a counting function. Another field of functional criterion ï determining types of proper names. When talking about persons or organizations, people naturally understand types of proper names without stating it explicitly. It is also expected from a habitable NLI. Consider the following question: *Where John Smith works?*. The habitable system should understand that the question is intended to find information about *person John Smith*.

The syntactic domain is defined as a set of different phrases, having the same meaning that can be understood by the system. It defines how flexible questions can be formulated. For example, if the system is not sophisticated enough, it cannot understand and interpret both of the presented questions in the same way [84]:

- *What is the salary of John Smith's manager?*
- *What is the salary of the manager of John Smith?*

The lexical domain is defined as a set of available words in the lexicon, including synonyms. It allows formulating more diverse question. Consider the following question: *What are the earnings of John Smith?* If the lexicon does not have some of its word, the question will not be answered.

Habitability of NLI directly correlates to its usability [21]. This term is more general and defines quality of the appropriateness to a purpose of any particular artefact [9]. With particular reference to information systems, ISO 9241 standard suggests that one of the areas that usability measures should cover is effectiveness [9]. It defines the ability to complete tasks using the system, and the quality of the output of those tasks. From the end users point of view, effectiveness of NLIs is typically evaluated in terms of precision and recall [20].

2.1.9 Usability enhancement methods

NLIs use methods to avoid the habitability problem and, therefore, improve usability. The main requirement of a habitable NLI is that its vocabulary must be aligned to that of the user. This means that the user must formulate questions using terms, understandable by the system. However, at the beginning of using a NLI, the user is not familiar with its vocabulary and does not even know, what can be asked. At this stage, methods help to familiarise a user with the vocabulary of the system and adapt it to that of the user [21]. These methods are presented further.

Guided interface is the method, which helps to familiarise the user with lexicon, language restrictions, and supported functions of the NLI. It guides user to write correct questions showing available words. The method is especially useful when the user is not familiar with the vocabulary and does not know what can be asked and how questions can be formulated. One of the options implementing guiding interface is autocompletion. Traditional autocompletion is based on matching input strings with a list of the words in a vocabulary. In NLIs to ontologies and other ontology-based systems, autocompletion can be extended to semantic autocompletion, when ontology resources are used to predict further available words [21] [47].

In [118], guided interface is implemented as a contextual menu, which is used to formulate natural language questions and commands. Users can write the question either by typing it or selecting items from menus, driven by a grammar. This helps to restrict the user to formulate only those questions that are understandable by the system and avoid the habitability problem.

Feedback is a technique to present the interpretation of question in a form, understandable by the user [21]. It familiarizes the user of how questions are interpreted, what mistakes are made, and what criterion of habitability is violated. Therefore, users start formulating questions in a way understandable by the system more quickly [136], [100]. Feedback allows users to confirm, whether the question is correctly interpreted and reformulate it if needed. Feedback is a desirable feature of the NLI, because it increases the confidence in the system and its returned results.

Controlled natural language (CNL) is a subset of natural language, based on restricted vocabulary and grammar rules that have to be followed [18]. On the one hand, CNL allows retrieving data without extensive training, whilst on the other; has less expressiveness than formal languages, used for querying. However, NLIs to ontologies usually use CNLs to avoid the complexity and ambiguity of natural language.

Synonyms, hypernyms, and hyponyms are used to extend the lexical domain of habitability. Synonyms are different words that have the same or nearly the same meaning. Hypernyms and hyponyms are used to show the relationship between more general word (i.e., hypernyms) and more specific words (i.e., hyponyms). To enrich the lexicon with synonyms, hypernyms, and hyponyms, external lexical resources, such as WordNet [28] or FrameNet [96] are often used.

Derivations are used to define concepts that are derived from other concepts. This activity is often performed in a configuration phase. Derivations extend both lexical domain, because more words are recognized, and functional domain, because some concepts can be derived based on their data properties (e.g., finding minimum, maximum values, etc.). For example, the term *large city* can be defined as the city having a minimum number of citizens.

Another group of usability enhancement methods are used when a question is already written. They help to resolve ambiguous questions and adjust the vocabulary of the user with the vocabulary of the system [21]. These methods are presented further.

Automatic ambiguity resolving is a method for resolving ambiguities using heuristics and knowledge base. NLIs to ontologies usually use ontology reasoning for mapping questions with ontology resources. For example, if a user did not explicitly identify some concept writing question, it can be resolved analyzing domains and ranges of object properties. In addition, morphological libraries, string metrics [22] [30] and other techniques could be used.

When ambiguities cannot be resolved automatically, **clarification dialog** involves the user into the process of interpreting questions to modify interpretations [20]. The user can supervise the process of a creating formal query to get results or make them more precise. This method helps mapping vague or ambiguous terms of questions with ontology concepts. For example, if a user enters the word *Mississippi*, it can be interpreted as a state or river. In such a situation, the system must show a clarification dialog, allowing the user to select the correct interpretation. The clarification dialog is in effective working with large knowledge bases, having many resources with identical names that leads to ambiguous questions. Another way to cope with ambiguities is using every possible interpretation of the question and showing all results. However, it is not feasible when there are too many interpretations [19].

Clarification dialog can be used together with **learning mechanism** to save userôs input and reuse it. It improves the userôs experience over time and reduces the cognitive overhead [20]. During the query interpretation, the learning mechanism can remember choices that were made previously in similar situations. For example, AquaLog system can remember novel phrases that a user entered and relate them with ontology structure. If the system faces the term *homepage* and the user relates it with ontology relation *has-web-address*, this decision will be remembered in the future.

Query refinement ï is changing the query in order to obtain results that are more relevant. It usually means adding more constraints to the query until the user obtains results that satisfies them [21]. This method is used to deal with ambiguities, which are caused by vague expression of the userôs information need [17].

In [106], query refinement assists the user to formulate a more precise question. When the user writes its initial question, the system tries to search for ambiguities analyzing the knowledge base and the structure of underlying ontology. To successfully refine question, it is important to know the information needs of the user. In this approach, userôs needs are obtained by analyzing its interaction with the portal. Finally, the refinement process derives more generic, more specific, similar (i.e., results are partially overlapped with initial results) or equivalent queries (i.e., results are the same, but the execution of query is optimized).

In QuestIO [17], query refinement is straightforward, allowing the user to select the set of returned documents.

2.1.10 Natural language interfaces to ontologies

In this subsection, the existing NLIs to ontologies, showing promising results of answering questions, are analysed. In order to find their commonalities and differences, the joint method of agreement and difference is used [71]. NLIs are compared on the following criteria: (1) portability; (2) possibility of automatic configuration; (3) capability to map language phrases with combinations of ontology resources; (4) capability to resolve language ambiguities automatically; (5) capability to resolve language ambiguities using clarification dialogs; (6) capability to refine queries; (7) clear adaptability for different languages. Algorithms of parsing and transforming questions to ontology queries are also analysed. In further text, the criterion number in parenthesis marks the mentions of analysed criterion.

The analysis starts from **QuestIO** [22]. This NLI does not require any user training and allows writing English questions of any length and form. QuestIO is portable (1), the lexicon is created automatically by generating a gazetteer list from morphologically normalized ontological lexicalizations. Therefore, the approach can be applied for different ontologies without configuration (2). QuestIO cannot map NL phrases with combination of ontology resources (3).

Questions are interpreted identifying key concepts and searching for relations between them based on object properties of the ontology. The algorithm of analysis and transformation of questions includes the following steps: linguistic analysis, ontological gazetteer lookup, transformation to SeRQL query, executing query and displaying results. In the first step, tokenization, POS tagging and morphological analysis is performed. In the second step, annotations for all mentions of ontological resources are created from the gazetteer list. In the third step, the most suitable interpretation is found. Finally, the question is transformed to a query which is executed against the ontology. Disambiguation (4) is performed using ontology reasoning in order to derive all potential valid interpretations of the question. To find the most suitable interpretation, fuzzy string distance metrics and similarity scores are used. Clarification dialogs (5) are not used in this approach. However, it uses query refinement, which allows to refine the set of returned documents or to provide an answer [17] (6).

The next NLI is **FREyA** [19]. It allows a flexible formulation of English questions, having no strict structures. FREyA is designed by the authors of QuestIO to have a better understanding of the semantic meaning of questions and provide concise answers. FREyA is a portable NLI (1), requiring no configuration. The lexicon is derived from the semantic repository by executing the set of SPARQL queries [17] (2). FREyA cannot map NL phrases with a combination of ontology resources (3).

The algorithm of translating question to query combines ontology reasoning and syntactic parsing. First, ontology based annotations, called ontology concepts (OC), are identified in the question. In the next step, a syntax tree is created. Certain words in the syntax tree (e.g., nouns, noun phrases, etc.) are identified as potential ontology concepts (POC). The algorithm iterates through all POCs and tries to map them to OCs automatically either (4) or engaging the user (5). If some POCs cannot be resolved, the algorithm finds the closets OC for that POC by walking through the syntax tree and generates suggestions using ontology reasoning. Suggestions are ranked using string similarity metrics, synonyms, and other algorithms. Clarification dialog is generated for the user to select the relevant suggestion [20]. When all POCs are resolved, the query is interpreted as a set of OCs and transformed to SPARQL.

To improve habitability, FREyA also uses query refinement together with feedback mechanism. It allows the user to confirm if the question is interpreted correctly or reformulate it if needed (6).

ORAKEL is a system, capable of understanding composite semantic constructions, such as quantifications, conjunctions, and negations [12]. ORAKEL is portable (1), but, unlike QuestIO and FREyA, it requires configuration (2). The mapping of NL phrases with ontology is defined creating linguistic structures, called subcategorization frames (i.e., verbs with their arguments). Part of the lexicon (including proper names) is automatically generated from the underlying ontology. WordNet [28] is used to append lexicon with synonyms. ORAKEL allows relating subcategorization frames with a combination of several relations in the ontology and answer questions that do not directly correspond to one relation in the ontology (3).

The parsing process includes syntactic analysis of the question and construction of semantic representation in terms of first order logic, enriched with query, count, and arithmetic operators. The syntactic analysis is performed using logical description grammar. First, parser selects elementary trees from the lexicon for each token. A parse tree is produced combining elementary trees. Then, the meaning of every word in the parse tree is analysed, semantic representation is created and translated into the query. In ORAKEL, ambiguities are resolved automatically (4) during the parsing process. The algorithm selects only those elementary trees that fulfil the ontological restrictions. Clarification dialogs (5) and query refinement (6) are not used in this approach.

PANTO [129] accepts English NL questions. It is designed to be portable (1) for different domain ontologies without manual configuration (2). The lexicon is built automatically from ontology entities. As well as in ORAKEL, proper names are written to the lexicon. Users can enter their own synonyms. This helps to adapt the system for specific domains. In this approach, NL phrases cannot be mapped with a combination of ontology resources (3).

The parsing and transformation to SPARQL is performed by the query translator. First, questions are parsed using the statistical Stanford Parser [58]. Nominal phrase pairs (i.e., phrases or words and their relationships expressed by verb phrases, prepositions, etc.) are extracted from the parse tree to form intermediate representation of the question, called query triples. Then, query triples are mapped to ontology triples using lexicon. Simultaneously, the parse tree is analysed to extract potential words for targets (i.e., variables after SELECT keyword) and modifiers (i.e., information for UNION and FILTER elements). Ontology triples, targets and modifies are finally used to generate a SPARQL query. Questions are disambiguated automatically (4), matching query triples to ontological triples. This step is performed employing semantic matching (i.e., using WordNet [28]) and morphological matching (i.e., using string metrics [14] and heuristic rules).

PANTO does not use clarification dialogs (5) or query refinement (6).

Querix [56] is a domain-independent NLI for the Semantic Web to answer NL questions in English. Querix is portable (1) and requires no manual configuration (2). The lexicon is constructed from the ontology automatically and is enriched using WordNet [28]. Querix does not have means to map NL phrases with a combination of ontology resources (3).

The algorithm of question analysis starts from creating the syntax tree using Stanford Parser [58]. Word categories of the syntax tree are used to compose the query skeleton. Then, a small set of heuristic patterns are used to identify triple patterns of the question. After finding possible triples in the skeleton and combining them with ontology resources, the SPARQL query is generated [56].

Querix does not try to resolve ambiguities of NL automatically (4), but asks the user for clarifications using dialogs (5). This approach does not use query refinement (6).

AquaLog [30] is a portable (1) question answering system, which interprets questions using terms and structure of the ontology. Although Garcia et al. [30] state that configuration time is negligible, AquaLog requires manual configuration (2). Therefore, it cannot be configured to map NL phrases with a combination of ontology resources (3).

The analysis of NL question starts with translating it into a set of intermediate representations ï query triples. Further, relation similarity service (RSS) is used to map query triples to ontology compliant triples. Ontology compliant triples are used to generate SPARQL query. Ambiguities are resolved automatically in RSS. The algorithm uses knowledge, encoded in the ontology and string metrics (4). Ambiguities can also be resolved by interacting with users using clarification dialogs (5). Query refinement (6) is not used in AquaLog.

The analysis is summarized in the comparison table in Table 2.2. It also includes SBVR based NLI to ontologies, created in this work.

Table 2.2. Comparison of NLIs to ontologies

Criterion	**QuestIO**	**FREyA**	**ORAKEL**	**PANTO**	**Querix**	**AquaLog**	**SBVR based NLI**
1. Portability	+	+	+	+	+	+	+
2. Automatic configuration	+	+	-	+	+	-	-
3. Mapping NL phrases with combination of ontology resources	-	-	+	-	-	-	+
4. Automatic ambiguity resolving	+	+	+	+	-	+	+
5. Clarification dialogs	-	+	-	-	+	+	+
6. Query refinement	+	+	-	-	-	-	-
7. Clear adaptability for different languages	-	-	-	-	-	-	+

The analysis of NLIs to ontologies led to some general conclusions. First, authors of the analysed approaches do not discuss adjusting them for writing questions in different languages, what components are language dependent, etc.

All of the analysed NLIs are portable. The lexicon of NLIs is often created semi-automatically. Part of the lexicon is generated from ontology lexicalization, and the rest is created manually by the user. However, most NLIs do not have a means to relate formulations of question with a combination of several ontology relations.

For improving the habitability, most NLIs use algorithms to solve ambiguities automatically using heuristic rules or ontology reasoning. When ambiguities cannot be resolved automatically, the system generates clarification dialogs and the users become involved.

The desire to write questions in different languages and the need to relate questions with combinations of ontology resources led to the solution of creating a new NLI to ontologies. In order to make transformation rules of questions independent from language, it was decided to use intermediate representation of question, instead of transforming a syntax tree directly to query. Further, SBVR metamodel and its capabilities to use an intermediate representation of NLI are analysed.

2.2 SBVR knowledge model and SPARQL query language

2.2.1 Overview of SBVR knowledge model

Semantics of Business Vocabulary and Business Rules (SBVR) is OMG specification to define the vocabulary and rules for describing business semantics (i.e., business concepts, business facts, and business rules) using SBVR Structured English (SSE) or other CNL [98]. SBVR is one of the first OMG standards for creating detailed business focused natural language specifications. The first version of SBVR specification was introduced in January 2008. The latest version was released in November 2013. SBVR is a synthesis of four disciplines [11]: terminology science, natural language grammar structures, formal logic, and business rules approach. The foundation of SBVR is a semiotic/semantic triangle, which is the theoretical basis for the SBVRǿs linguistic based architecture.

The idea of SBVR is to raise the process of information system development into a more abstract level by creating information system specifications using some kind of structured language. Therefore, SBVR specifications are understandable for business people and interpretable by software tools. By providing business-oriented language for specifying business knowledge and capability to transform them to software models and artefacts, SBVR bridges the gap between business requirements and formal models of information systems [23].

SBVR is an approach that enables people and organizations to treat business, legal, and educational knowledge in a productive way not seen before [78]. However, applying SBVR in practice has various limitations. For example, SBVR lacks the larger collection of data types and patterns for expressing arithmetic operations, date, time, past, future events, etc. Although SBVR specification is extensible, standard constructs for the most frequent cases are desired. Spreeuwenberg and Anderson noticed more deficiencies of SBVR: lack of inferences, lack of references (i.e., rules should be stated in single sentences), necessity to introduce concepts before referencing to them, impossibility to express directives, etc. [104], [105]. However, the main drawback of SBVR is the complexity of its metamodel for production of suitable vocabularies [46]. In reality, sophisticated SBVR tools should handle the complexity. Users should work with SBVR Structured English language and create business specifications with the help of business experts.

Researchers have introduced transformations of SBVR specifications to various software models: UML&OCL [16], [76] and vice versa [10], BPMN [7], [122], RDB schemas [70], SQL [74], OWL [24], OWL 2 [52], [89], [51], [53], Web Services [31], [50], etc. In addition, several EU projects were devoted to create tools for authoring SBVR business vocabularies and rules, and transforming them into various software models and code. OPAALS (2006-2010, generating Web services and data models from SBVR specifications [70], [77]), ONTORULE [85] (2009ï 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 [15] presents capabilities for authoring SBVR vocabularies and rules, generate ontologies and various models of information systems. Besides automating development of software models and their implementations [63], [65], 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 disclosed by the fact that SBVR specification itself is formally written in SSE [62].

SBVR mainly covers two aspects ï business vocabulary and business rules. Business vocabulary specification contains concepts and their relations while rules serve as elements of guidance that govern actions. Specification of business vocabulary and rules allows describing organization from static and dynamic points of view. SBVR also allows questioning business models and their implementations. However, this functionality attained only little attention from related research and SBVR specification [112]. Kriemacher in 2006 noticed the possibilities of SBVR questions [59] for business people to query systems for business models without the support of programmers. SBVR questions 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. However, it is believedthat no further research in that direction was done.

2.2.2 Overview of SBVR metamodel

The metamodel of SBVR consists of three main parts: meaning, representation, and expression. Elements of meaning holds the shared sense that people assign to real world objects. Metamodel allows defining meaning of business concepts, their relations, business rules and questions. The meaning is not a text, graphical symbol, or some other kind of representation. The metamodel of SBVR meaning is presented in Figure 2.3.

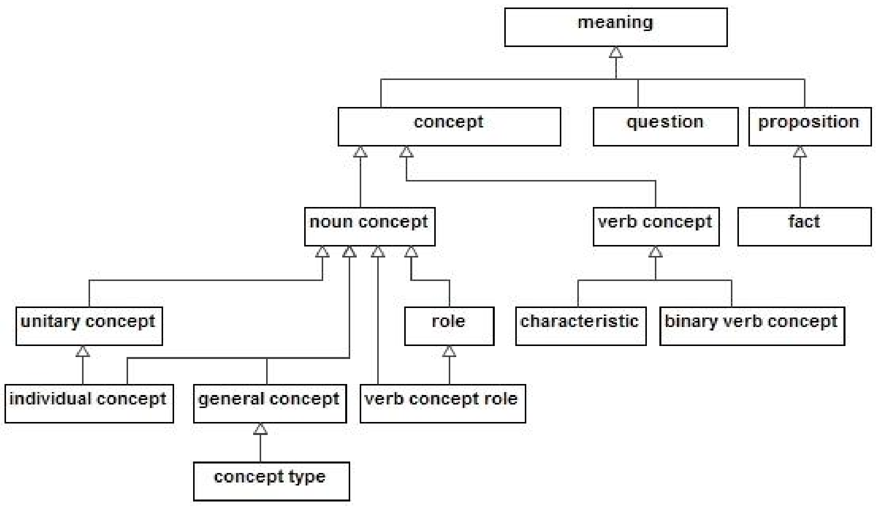


Figure 2.3. Metamodel of SBVR meaning [98]

Concepts are used to formulate meaning of business vocabulary. Noun concepts and verb concepts describe business entities and their relations. They are used to compose a conceptual schema that is a basis for defining propositions and questions.

The *question* element is used to define meaning of interrogatory. The word ñquestionò has two common meanings: written or spoken expression of inquiry and the meaning of such inquiry. Using the second definition, a question can be asked in different languages. However, the first definition results different expressions for different languages. The concept ñquestionò here should be understood in the sense of meaning, without confusing it with expression or representation of question [98]. The meaning of question is defined by semantic formulations ï structures based on formal logic, as presented in the further subsection.

The last type of SBVR meaning is proposition. Proposition is the meaning of a declarative sentence and is used to express meaning of a business rule. Facts are propositions that always have meaning *true*.

Elements of representation relate meaning with expression (i.e., elements used to communicate, such as text, sounds, diagrams, etc.) as presented in Figure 2.4. Meaning can have many representations, therefore they can be expressed in different ways. It is necessary for two reasons [94]: allows using different types of representations and multilingualism (i.e., sharing and talking about the same concepts in different languages).

Vocabulary concepts can have definitions. Definition is a representation of concept by a descriptive statement, which serves to differentiate it from related concepts. Definitions can be formal (i.e., expressed by logical formulations) and informal (i.e., expressed in natural language sentences having no underlying SBVR logical formulations).

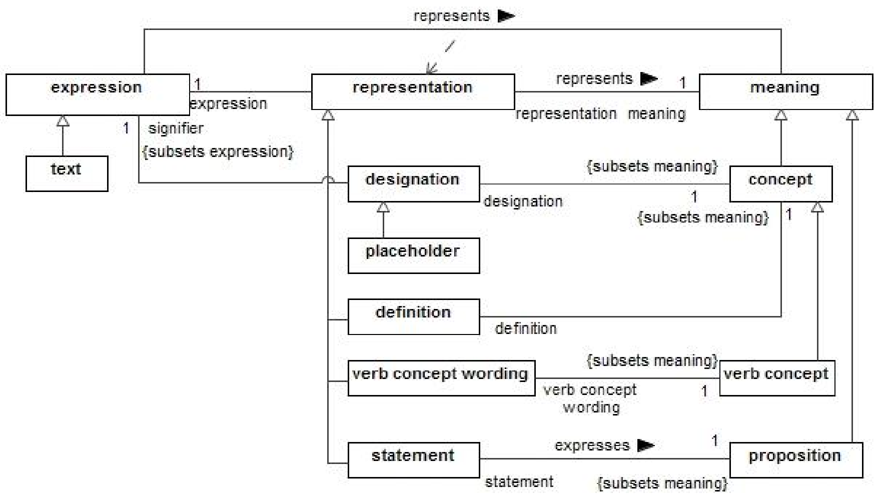


Figure 2.4. Metamodel of SBVR meaning and representations [98]

2.2.3 Formulating the meaning of question

The meaning of a question is formulated using specific SBVR semantic formulation ï projection. According to the SBVR specification [98], a projection returns a set of things that satisfy a projectionῶs constraints. The projection introduces one or more variables to represent results of a question. Types of results are defined by general concepts that variables range over. For example, if one wants to see a list of persons in results, the question introduces a variable that ranges over the general concept person. For introducing other types in results (e.g., organizations that person works in), the corresponding variables are introduced.

A projection is constrained by logical formulation, which projects variables, using first-order logic. The constraining logical formulation scopes over the atomic formulation, which is based on a verb concept. Depending on formulations of questions, verb concept roles can be bound to particular bindable targets ï variables or individual concepts. SBVR metamodel fragment for representing questions is presented in Figure 2.5.

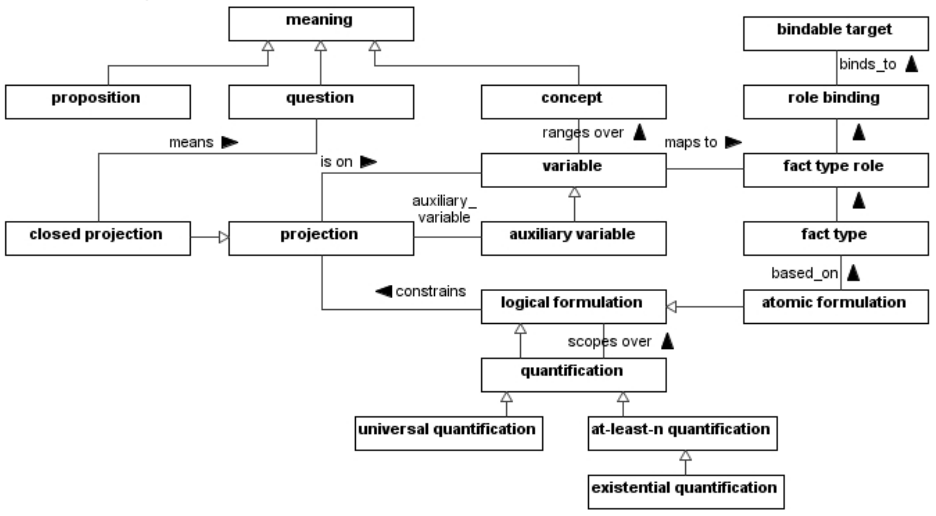


Figure 2.5. SBVR metamodel fragment for representing questions [98]

2.2.4 Using SBVR for NLI to ontologies

There are several reasons, why SBVR could be used for NLI to ontologies. First, the SBVR metamodel allows to model questions and could be used as an intermediate representation of meaning for transforming questions to queries. SBVR separates meaning from expression and allows expressing the same things differently and in different languages. For example, questions will have the same meaning, even though they are written in different languages.

SBVR specifications allow describing semantics of the business domain, required for semantic search (hypernyms, hyponyms, synonyms, etc.). In addition, SBVR allows specifying derivations of concepts from other concepts and can support inferences [64]. SBVR derivation rules seem to have a potential for bridging the gap between the ways in which specific data is stored (i.e., the ontology scheme) and how a user thinks about data. It is an important function of NLI, because facts questioned using simple and more abstract language formulations can be stored in more complex ontology structures (i.e., expressed through several object properties, derived from values of data properties, etc.).

2.2.5 Overview of SBVR editing tools

An important prerequisite to use SBVR for NLI to ontologies is a robust SBVR editing tool that would be used in the configuration phase to specify vocabulary and rules (i.e., lexicon) of NLI to ontologies. Such an editor should support SBVR questions and be capable to serialize them to XMI models for further transformations to SPARQL queries. In this section, the overview of freely available SBVR editing tools is presented. An important feature of a SBVR tool is its capability to formalize business vocabularies, business rules, and questions in accordance with SBVR metamodel. SBVR specifications should be serialized using SBVR XMI format to allow transferring them to other tools or transformation engines.

The first overviewed SBVR tool is an open source SBVR editor SBeaVeR. It was created in 2006 as an Eclipse plugin [23]. SBeaVeR is capable to specify and validate business vocabularies and business rules using SSE. Specifications can be transformed to formal models (i.e., XML Schema format) to support the interchange of specifications between software tools. SBeaVeR editor has syntax highlighting following SBVR Structured English style, hierarchical vocabulary navigation panel, and embedded WordNet [28] dictionary for support of synonyms, hypernyms, hyponyms, meronyms, and informal definitions. Specifications are parsed and validated using LL parser. It is a top-down parser for context-free languages, constructing a leftmost derivation. Unfortunately, the tool is not further supported.

Another open source tool is VeTIS [76], [123]. It was created on a base of SBeaVeR. It provides all functionality of SBeaVeR plus better capabilities of specifying business vocabulary (i.e., support of individual concepts) and recognizes a wider variety of business rules. In VeTIS, business vocabularies and business rules are serialized into SBVR XMI models and further transformed to UML models with OCL expressions. However, both tools lack the flexibility for further extensions to realize the full potential of the SBVR knowledge model.

Marinos et al. presented a SBVR editor with syntax highlighting and auto-completion functions [69]. Unlike SBeaVeR and VeTIS, it allows writing terms or verbs consisting of multiple words without joining them with a dash. This tool also automatically recognizes terms in plural form even if they were declared in a singular form. It was implemented using the Active Support for JavaScript tool that uses a number of well-known English patterns for determining plural nouns from singular ones [69]. The grammar was written in OMeta, an object oriented language for matching patterns, based on expression grammars [131]. Expression grammar is similar to context free grammar, except that the ambiguity eliminated by prioritizing alternatives [29]. The editor was implemented to work on a Web browser using CodeMirror library. It is capable to specify terms, verb concepts with attributes, complex business rules (i.e., based on several fact types), and quantifiers.

RuleXpress [95] is a commercial framework for developing and managing business vocabularies and business rules. RuleXpress allows creating conceptual models that comply with the SBVR metamodel using a graphical interface. It does not use any controlled language for writing business rules. Instead, it allows using a natural language and recognizes only those concepts that are specified in a vocabulary. RuleXpress is not an execution tool, but rather a tool to help people to organize and understand their business.

The SBVR lab 2.0 is a commercial SBVR editor working on a Web browser [97]. This tool allows specifying business vocabulary and business rules in SSE. Each vocabulary term and business rule should be indicated by specifying their type and features. The SBVR lab 2.0 allows using binary verb concepts and characteristics. Like in Marinos et al. editor [69], verbs of verb concepts can consist of several words without joining them. The editor has a unique function of graphical visualization of business vocabulary and publishing it to the Web.

Table 2.3 presents the comparison of the analysed SBVR tools concerning several essential features.

Table 2.3. Comparison of SBVR editors

	SBeaVeR	**VeTIS**	**Marinos et al. SBVR Editor**	**RuleXpress**	**SBVR Lab 2.0**
Language used in business vocabulary and business rules	SSE	SSE	SSE	Graphical interface	SSE
Automatic highlighting and auto completion	+	+	+	–	+
Language parserĝs grammar	Context-free grammar	Context-free grammar	Extended expression grammar	Uncontrolled natural language	Unknown
Formalization in compliance with SBVR metamodel	+	+	–	–	+
Model interchange format	XMI XSD	XMI model	–	–	–
Possibility to access generated XMI	+	+	–	–	–
Possibility to extend tool capabilities	Limited	Limited	Unknown	Unknown	Unknown
Support of SBVR questions, including XMI serialization	-	-	-	-	-

The analysis leads to some general conclusions. First, most of the SBVR tools provide special functionalities (e.g., automatic text highlighting, using style and colours of SSE, autocomplete, etc.) to relieve the creation of business vocabularies and business rules. The SBVR Lab 2.0 is distinguished by its graphical interface to visualize SBVR specifications. The analysed tools provide different formalization levels. Some of them, such as Marinos et al. editor or RuleXpress, preserve text structure only, whereas others are able to transform text to formal SBVR specifications and analyse their compliance. The analysed tools use different interchange formats: XMI XSD allows interchange of SBVR conceptual schema only, whereas the XMI model allows interchange of the overall SBVR model ï its conceptual schema, individuals, facts, and semantic formulations. SBeaVeR and VeTIS tools allow accessing generated XMI schema; this function is useful for developers, creating model transformations.

Unfortunately, none of SBVR editors supports questions. Further extensions of them are prevented because of an inability to access the required functionality or such extensions are extra complicated. Therefore, a new SBVR editor is required.

2.2.6 Ontology query language SPARQL

The analysis continues with ontology query language SPARQL. It covers SELECT query type and main syntactic elements, required to answer types of questions, described in section 2.1.5. This subsection is prepared according to the specification of SPARQL 1.1 [43]. Models are created in accordance with the syntactic structure of SPARQL 1.1, presented in the specification.

After RDF data model was introduced in 1998, when a problem of data extraction arose. Many query languages were suggested (RQL, SeRQL, TRIPLE, RDQL, N3, Versa, SPARQL [42]). Finally, SPARQL became an official recommendation of W3C and now is the *de facto* ontology query language [103]. SPARQL is based on analysis of matching RDF graph patterns. SPARQL query contains a set of triple patterns called a basic graph pattern. Triple patterns are similar to RDF triples, except that triple pattern can contain variables [7]. During execution of a query, triple patterns are matched with RDF graph triples, variables of the graph pattern are bounded with RDF elements, and query results (a solution set) are provided.

The original version of SPARQL lacked many features for querying ontologies. The SPARQL working group received many requests from the community to complement SPARQL with new features. As a result, SPARQL 1.1 was released, offering many new features that facilitated writing queries and making them simpler. This version is an official recommendation of W3C from March 2013. The most important features of SPARQL 1.1 are aggregate functions, subqueries, negations, property paths, assignment of value to variables [57] [43].

Although SPARQL has four types of query (i.e., SELECT, ASK, DESCRIBE, and CONSTRUCT), only SELECT queries are used in this work. Figure 2.6 presents top-level elements of this query type: SELECT clause, dataset clause, WHERE clause, and solution modifier.

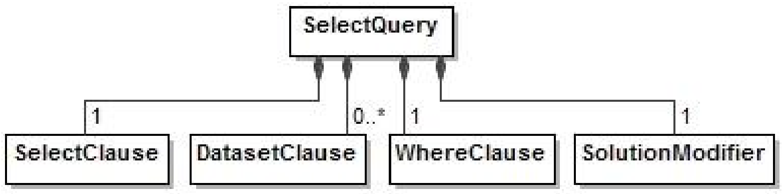


Figure 2.6. SPARQL 1.1 SELECT query

SELECT clause is a mandatory part of the SELECT query (Figure 2.7). It is used to declare, what results the query should give and includes a list of variables from a pattern matching. It can also include *Expression* elements, to represent results from SPARQL 1.1 aggregate functions, such as counting, finding minimum or maximum values, etc. The SELECT clause has optional DISTINCT and REDUCED modifiers. DISTINCT is used to eliminate duplicate solutions. For large datasets, where DISTINCT can be too slow, the REDUCED operator can be used to permit solutions to be eliminated. In some situations, REDUCED can perform only straightforward deduplication, i.e., to remove immediately repeated results and to leave duplicates that are expensive to remove. In certain situations, it can be good enough.

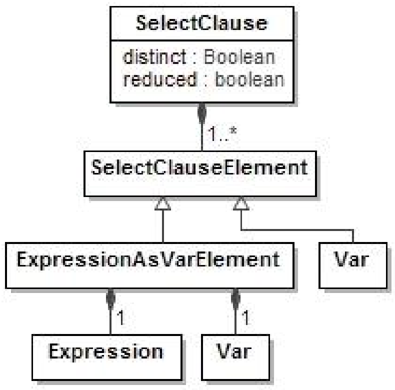


Figure 2.7. SPARQL 1.1 SELECT clause

Queries are executed against RDF datasets. The RDF dataset is a set of graphs that always contains one default graph and zero or more named graphs. Each of the named graphs are identified by IRI. If a graph is not specified in the dataset clause, the query is executed against the default graph. The Dataset clause can be defined using two keywords. FROM keyword is used to execute the query against the default graph and specified named graphs. While FROM NAMED is used to execute query only against specified named graphs. The structure of dataset clause is presented in Figure 2.8.

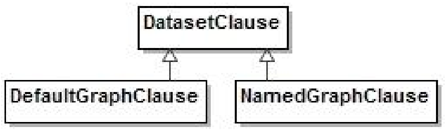


Figure 2.8. SPARQL 1.1 dataset clause

The WHERE clause is used to define graph patterns (Figure 2.9). Graph patterns are used to produce a result set by matching them with RDF graphs. Graph patterns contain variables and results are formulated when each variable of the graph pattern has binding to a certain RDF element. If at least one variable does not have a binding, such results are rejected.

The WHERE clause is expressed by a graph pattern element, which contains a number of triple block elements. The triple block element contains a number of graph patterns, expressed by *TriplesSameSubjectLeft*. This element has a structure of subject, predicate, and object. In positions of subject and object, variables or graph elements (i.e., IRI references, blank nodes, RDF literals, numeric literals, Boolean literals, etc.) can be used. In a position of predicate, variables or IRI references, expressed by a *VerbPath* element, are available.

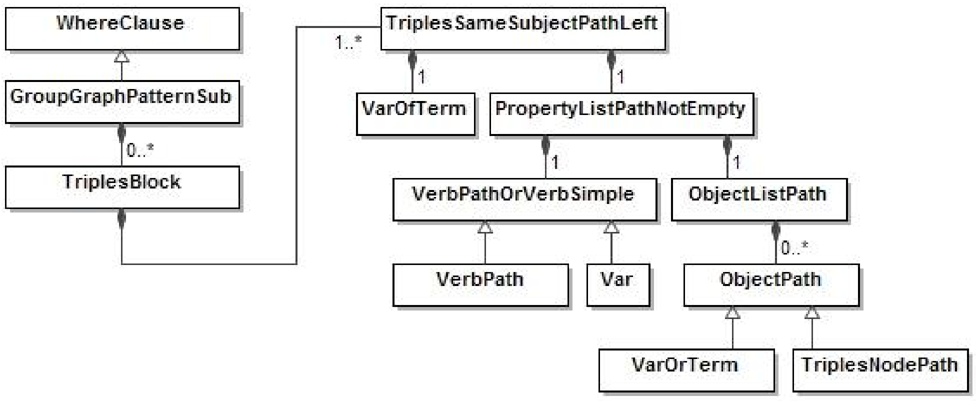


Figure 2.9. SPARQL 1.1 WHERE clause

The WHERE clause can also contain FILTER elements. They are expressed using *GraphPatternNotTriples* and *FilterConstraint* elements as presented in Figure 2.10. Note that SPARQL 1.1 specifications has many other types of filter constraints, such as functions for data type conversions, IN, NOT IN operators, rounding functions, etc. In this model, only constraints for numeric comparison and string matching are included.

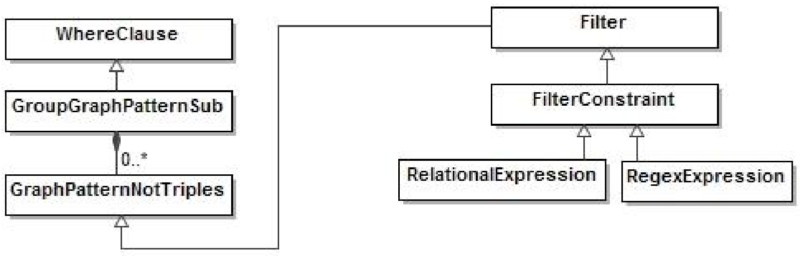


Figure 2.10. SPARQL 1.1 FILTER

The last part of the SELECT query is a solution modifier (Figure 2.11). This part is used after pattern matching for the following reasons: divide results into smaller groups with GROUP BY modifier to calculate aggregate values; filter grouped solution sets using HAVING modifier; order results using ORDER BY modifier; slice results using LIMIT and OFFSET modifiers.

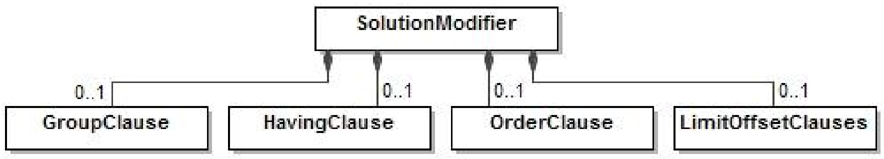


Figure 2.11. Structure of SPARQL 1.1 SELECT query solution modifier part

This analysis also revealed that SPARQL syntax metamodel and SBVR metamodel of questions have conforming elements and suggests that transformation of SBVR questions to SPARQL is feasible. For example, verb concepts of SBVR closed projections conforms to SPARQL triple patterns, variables of closed projection conforms to variables of SELECT clause, etc. Therefore, model based transformations of SBVR questions to SPARQL will be developed and used for NLI to ontologies.

2.3 Analysis of related works of Lithuanian researchers

Research on Knowledge Models, Information Systems, and Model Driven Development (MDD) has a long history in Lithuania and currently is following the newest trends. It is worth mentioning theresearch done in the Department of Information Systems (ISD) at Kaunas University of Technology (KTU); Information Systems Department (ISD) of Vilnius Gediminas Technical University (VGTU); the joint project of ISD (KTU) and ISD (VGTU) VeTIS [123]; the joint project of ISD (KTU) and Vytautas Magnus University (VDU) SemantikaLT [99]; research in Kaunas Faculty of Humanities of Vilnius University (VUKHF), Klaipãda University (KU), and others.

The main directions of ISD (VGTU) are business rule approach [80], [81], [121] and ontologies [119], [120], [49]. All mentioned research work is related with Model Driven Development.

Related work of Kaunas Faculty of Humanities of Vilnius University and Klaipãda University are mainly directed towards Knowledge based engineering of Information systems [39], [40], [41], [66], [67], [68], [114].

The closest relation of this research is with research work that is done or is being carried out in the ISD (KTU), specifically, for creating the Semantic Search Framework based on SBVR (Figure 2.12). The Semantic Search Framework was applied in the SemantikaLT project, but separate parts of this framework have a broader value and applicability. Research work, directly related with this research, are transformations of SBVR business vocabularies and rules to OWL 2 ontologies [52], [51], [53], creation of SBVR business vocabularies and rules from OWL 2 ontologies [5], [61]. This is because SPARQL 1.1 queries, obtained from the particular SBVR business vocabulary and rules, must be formulated using concepts of the ontology, which strictly corresponds to the same SBVR business vocabulary and rules.

Other parts of the Semantic Search Framework (e.g., process of semantic annotation, including anaphora and co-reference resolution, linguistic processing of text corpus, etc.) are beyond the scope of this research, which is limited to the obtaining of SPARQL 1.1 queries that can be executed in OWL 2 ontologies.

Besides that, this work is related with the experience of the Department of Information Systems in the fields of Conceptual Modelling, SBVR Business Vocabularies and Business Rules, Ontology Engineering, MDD [126], [125], RDB [88].

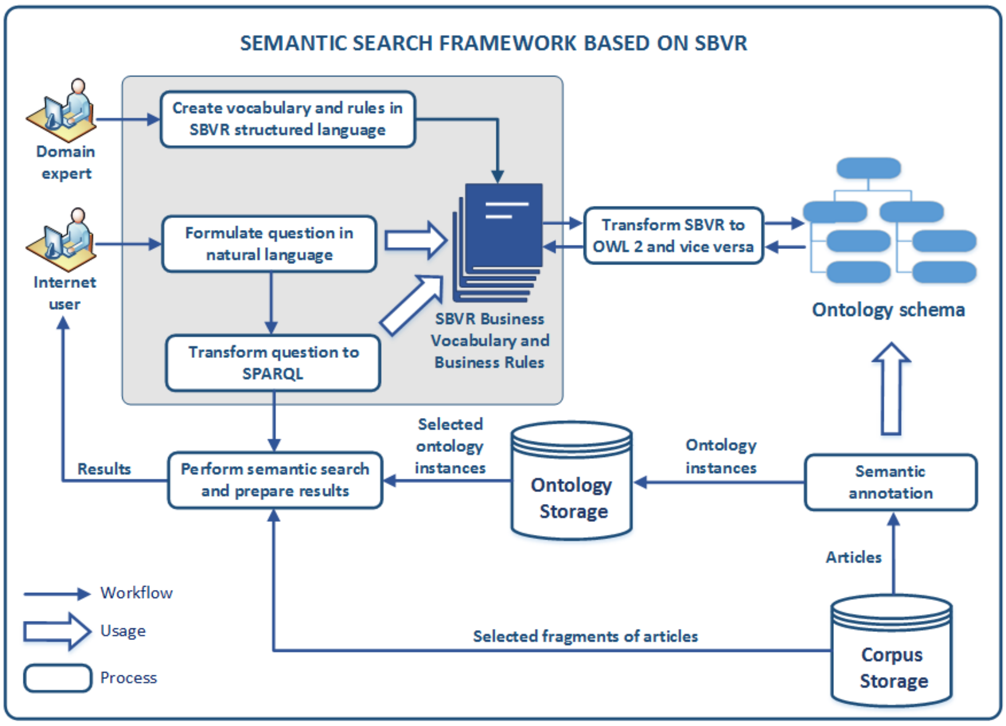

SEMANTIC SEARCH FRAMEWORK BASED ON SBVR
Domain expert
Create vocabulary and rules in SBVR structured language
Internet user
Formulate question in natural language
Transform question to SPARQL
SBVR Business Vocabulary and Business Rules
Transform SBVR to OWL 2 and vice versa
Ontology schema
Results
Perform semantic search and prepare results
Selected ontology instances
Ontology Storage
Ontology instances
Semantic annotation
Articles
Corpus Storage
Selected fragments of articles
Workflow
Usage
Process

Figure 2.12. Semantic search framework based on SBVR [99] (parts, distinguished by darker background, were created on the base of this research)

2.4 Analysis summary

The huge knowledge resources accumulated on the Web, organization documents, and data storages require new ways for search and analysis of knowledge. Ontologies seem the most relevant means for coping with this problem, but they need human friendly interfaces. The conclusions of the analysis are as follows:

1. Analysis of scientific publications has shown that the preferable interface for querying ontologies is a natural language interface (NLI). The most important requirements of such NLI are adjustability to different languages, ability to deal with complex structure of ontology resources, portability, and habitability.
2. Existing NLIs to ontologies allow questioning only in English and usually in the form when formulation of question directly correspond to the structure of

ontology. On the other hand, existing solutions are portable and can be configured for questioning in different domains.

3. Habitability of NLI expresses how naturally and easy a user can write questions. To improve habitability, NLIs use various techniques that can be divided into two groups: (1) methods intended to familiarize users with lexicon and help formulating questions (e.g., guided interface, feedback, synonyms, etc.); (2) methods intended to help interpreting and disambiguating questions (e.g., clarification dialog, query refinement, etc.).
4. Analysis of SBVR standard has shown that the distinguishing feature of SBVR metamodel to separate the meaning from representation allows achieving multilingualism. This means that a question can have the same model of meaning regardless of language it is written. It allows using language-independent rules to transform questions to semantic (SPARQL) queries. The language dependent components of NLI should only be those that help writing and interpreting questions. In order to achieve multilingualism, only these components should be replaced or adjusted for questioning in certain language. The architecture of NLI will be designed in pursuance of these ideas.
5. An important prerequisite to use SBVR for NLI to ontologies is a robust SBVR editing tool for specifying vocabulary and rules (i.e., lexicon) of NLI to ontologies. The tool should be capable of generating XMI models of questions for further transformations. The analysis of existing tools showed that none of them meets the requirements and their further improvements are complicated. Therefore, it was decided to create the new SBVR editor.
6. The analysis revealed that metamodels of SBVR meaning and SPARQL syntax has conforming elements, expressing information needs and restrictions of query. This led to the assumption that transformation of SBVR questions to SPARQL is feasible. Therefore, it was decided to describe detailed mappings between these metamodels and create transformation rules of SBVR questions to SPARQL.

3 THE SEMANTIC SEARCH SOLUTION BASED ON SEMANTICS OF BUSINESS VOCABULARY AND BUSINESS RULES

This section presents the Semantic search solution using SBVR. First, requirements of the solution are introduced. Following, the solution is presented in detail, revealing created algorithms, and models.

3.1 Requirements of Semantic search solution

The solution consists of two parts: SBVR structured language editor (SBVR SLE) and SBVR based NLI to ontologies. Their functional requirements are presented in Figure 3.1. SBVR SLE is a tool for a domain expert, who is responsible for the configuration of NLI. This tool integrates SBVR transformations to OWL 2, created by J. Karpovič and described in [51], [52], and [53]. SBVR SLE can also be used as a developer tool to write specifications and generate SBVR XMI models of questions for developing and testing transformations to SPARQL.

NLI to ontologies is intended for internet users to write natural language questions and perform a semantic search over OWL 2 ontologies.

The solution (both SBVR SLE and NLI to ontologies) uses an external morphological library to analyse words (i.e., get lemma and other morphological features, such as case, number, gender, etc.) and generate words in certain morphological forms.

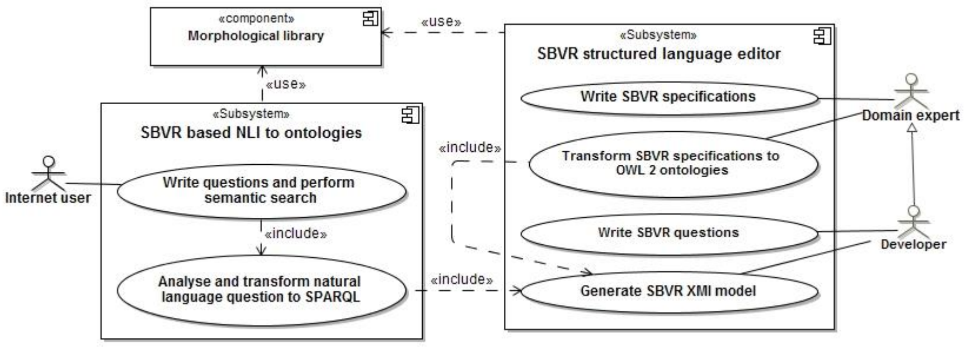

«component»
Morphological library
«use»
«Subsystem»
SBVR structured language editor
Write SBVR specifications
Transform SBVR specifications to OWL 2 ontologies
Write SBVR questions
Generate SBVR XMI model
Domain expert
Developer
«use»
«Subsystem»
SBVR based NLI to ontologies
Internet user
Write questions and perform semantic search
«include»
Analyse and transform natural language question to SPARQL
«include»
«include»

Figure 3.1. Functional requirements of Semantic search solution using SBVR

Non-functional requirements of SBVR SLE are as follows:

- Usability of writing SBVR specifications providing autocomplete, highlighting syntax following SBVR Structured English style, automatically marking errors of specifications, and checking cross-references between concepts;
- Ability to adjust the editor for writing specifications in different languages;
- Ability to write SBVR specifications in different domains;
- Ability to split SBVR specifications into separate parts;
- Ability to adjust the editor for different transformations (e.g., OWL 2 ontologies, BPMN business processes, etc.).

Non-functional requirements of NLI to ontologies are the following:

- Ability to adjust NLI for writing natural language questions in different languages;
- Portability to allow questioning in different domains;
- Ability to relate formulations of questions with combinations of ontology resources;
- The effectiveness must be similar as other NLIs to ontologies: *f-measure* above 80% in Mooney knowledge base of geography and above 70% of restaurants.

The process of the solution is also split into two parts: configuration of NLI and questioning ontologies (see Figure 3.2). NLI to ontologies is configured creating SBVR specification and synchronizing it with the ontology. Then, the user can start questioning. It is important to note, that questions should correspond to SBVR specification. During the analysis of a question, it is identified the SBVR concept (s) that the question is based on. This concept is further used to create intermediate representation of the question (i.e. SBVR model of meaning) and transform it to query. Finally, query is executed against the ontology.

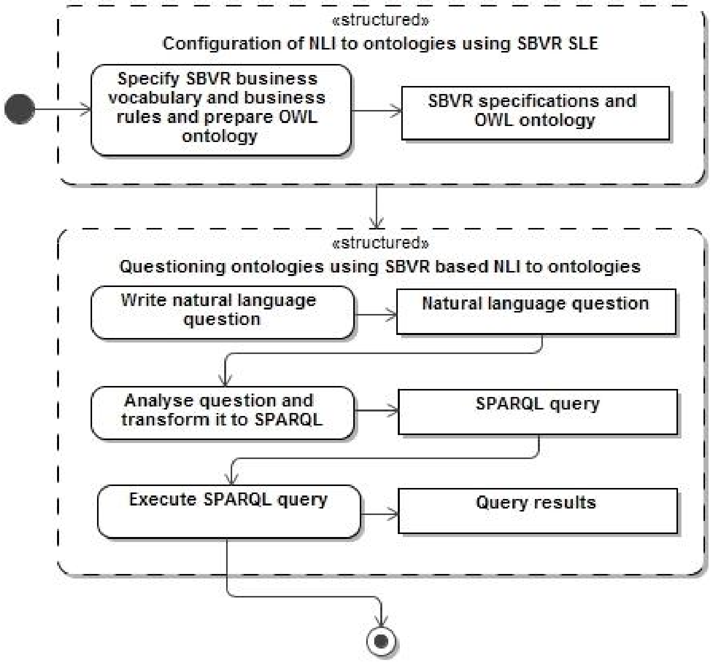

«structured»
Configuration of NLI to ontologies using SBVR SLE
Specify SBVR business vocabulary and business rules and prepare OWL ontology
SBVR specifications and OWL ontology
«structured»
Questioning ontologies using SBVR based NLI to ontologies
Write natural language question
Natural language question
Analyse question and transform it to SPARQL
SPARQL query
Execute SPARQL query
Query results

Figure 3.2. Process model of Semantic search solution using SBVR

3.2 Configuration of NLI to ontologies

Configuration is performed to adjust NLI to a certain domain. The results of this process are SBVR specifications, containing business vocabulary, business rules, and OWL 2 ontology that corresponds to SBVR specifications. SBVR specifications are used as a lexicon for writing and interpreting natural language questions. It is important to ensure the conceptual coverage of specifications and ontology (i.e., ontology classes and properties must be specified as SBVR concepts to allow questioning them). For alignment of SBVR specifications and OWL 2 ontologies, relevant mappings between SBVR concepts and OWL 2 resources should be holdcompleted. Following recommendations from [52], [51], [5], [61], [53], it was decided to use the following mappings:

- SBVR general concepts to OWL classes;
- SBVR binary fact types to OWL object properties;
- SBVR *is_property_of* verb concepts to OWL data properties;
- Categorizations of SBVR concepts to OWL *SubclassOf*, *SubObjectProperty* or *SubDataProperty*;
- SBVR individual concepts to OWL individuals;
- SBVR logical formulations to OWL 2 axioms and restrictions.

There are three ways to prepare SBVR specifications and OWL 2 ontology. The first one is specifying SBVR vocabulary and rules, and transforming them to OWL 2 ontology using the automatic transformation, created by J. Karpovič [51], [52], [53]. For this purpose, transformations of J. Karpovič are integrated into SBVR SLE. However, many more ontologies than SBVR business vocabularies and rules are created nowadays. Fortunately, the opposite transformations are under development and in the future it will be possible to transform OWL 2 ontology to SBVR specifications [5], [61]. In both cases, the compliance between SBVR specifications and OWL 2 ontologies is ensured automatically, so the possibility of errors is lower. The third way is creating SBVR specifications and OWL 2 ontology manually, ensuring compliance by domain expert. However, this way of configuration requires a greater effort from the domain expert.

From a technical point of view, the alignment of SBVR specifications and OWL 2 ontology is ensured by labeling ontology resources with expressions of corresponding SBVR concepts. For example, SBVR general concept *person* must have the corresponding ontology class with label *person*. In an analogous way, verb concepts must have corresponding labelled object properties.

Another important activity of the configuration process is writing derivation rules. This allows extending the set of allowed questions or making them shorter. Derivation rules are used to specify, how new facts are derived from existing ones. Two types of derivation rules are allowed: rules for deriving general concepts and rules for deriving verb concepts. Rules of the first type are used to define concepts that correspond to ontology classes, deriving them from their properties. For example, general concept *large_city* can be derived from its population or area.

Derivations of the second type are used to shorten questions, when their formulations do not meet the structure of the ontology. Such situations are common using n-ary relations. For example, the question *What did person say?* is based on verb concept *person said speech_content*, which may not have the corresponding object property in the ontology. However, it can be deriver from other object properties (i.e., *person gave speech and speech has speech_content*). Such derivations can be specified in the ontology (e.g., using SWRL rules to define semantic relations [110]). In this solution, they are specified as SBVR derivation rules and used transforming questions into SPARQL queries. Therefore, queries hold required derivations and the amount of derivation rules in the ontology is reduced.

3.3 The conception of SBVR structured language editor

Configuration is performed using SBVR SLE. The conception of this tool is defined as structured language grammar for specifying business vocabulary, business rules, and writing questions. Grammar rules are specified using Xtext syntax (EBNF like) and presented in the following subsections. Rules correspond to the representation part of SBVR metamodel. They are specified following metamodel of SBVR representations, examples of Structured English from SBVR specification ([98]) and practice. The presented grammar is intended to write SBVR specifications in English. However, it can be adjusted to other languages, appending rules with keywords of a certain language. Since no domain specific words are hard coded, grammar rules are independent from the business domain.

3.3.1 Grammar rules for specifying SBVR business vocabulary

Structured language grammar contains terminal (i.e., lexer) and production rules. Terminal rules are fundamental building blocks and are used to produce tokens. In this grammar, the main terminal rules are as follows: TERM_OR_VERB_SYMBOL (i.e., words, starting with lowercase letter) and NAME (i.e., words, starting with uppercase letter or digit). These rules are used to formulate basic SBVR structures, such as terms, names, and verb symbols. Additional terminals (e.g., NEWLINE, ML_COMMENT, etc.) are used for editing purposes. Xtext grammar allows using terminal fragments (i.e., reusable parts). They are not counted as tokens, but makes grammar simpler and more readable. Fragments are used to define available lexer symbols: Latin, Lithuanian letters, and digits. Terminal rules are presented in Table 3.1.

Table 3.1. Terminal rules of the grammar

```
terminal TERM_OR_VERBSYMBOL:
  LOWERCASE(LOWERCASE|DIGIT|'_'|'-')*;
terminal NAME:
  (UPPERCASE|DIGIT)(LOWERCASE|UPPERCASE|DIGIT|'_'|'-')*;
terminal NEWLINE:
  ('\r'|'\n')*;
terminal ML_COMMENT:
  '/*' -> '*/';
terminal fragment LOWERCASE:
  ('a'..'z')|'ą'|'č'|'ę'|'ė'|'į'|'š'|'ų'|'ū'|'ž';
terminal fragment UPPERCASE:
  ('A'..'Z')|'Ą'|'Č'|'Ę'|'Ė'|'Į'|'Š'|'Ų'|'Ū'|'Ž';
terminal fragment DIGIT:
  ('0'..'9');
```

Production rules are used to produce syntactic trees. The smallest production rules of SBVR business vocabulary grammar are *Term*, *Name*, and *VerbSymbol* (Table 3.2). Rules *Term* and *VerbSymbol* share the same terminal rule defining their syntax. They can be applied, when a word starts with a lowercase letter. Syntax of *Name* is defined by another terminal and can be applied when a word starts with an uppercase letter or digit.

Table 3.2. Smallest production rules of SBVR business vocabulary grammar

```
Term:
  syntax=TERM_OR_VERBSYMBOL;
Name:
  syntax=NAME;
VerbSymbol:
  syntax=TERM_OR_VERBSYMBOL;
```

The smallest production rules are used to define rules of representing vocabulary concepts: *GeneralConcept*, *IndividualConcept*, and *VerbConcept* (Table 3.3). These representations consist of primary representation and optional specification, containing a set of captions. Captions provide additional information of vocabulary concept (e.g., textual descriptions, synonyms, synonymous forms, general concepts, definitions, etc.). Some captions are applicable only to particular concepts. For example, *Synonymous_form* can be used only for verb concepts. Therefore, different sets of captions for each concept type were defined.

Table 3.3. Rules to represent vocabulary concepts

```
GeneralConcept:
  primaryRepresentation=Term
  (captions+=CaptionForGeneralConcept)*;
IndividualConcept:
  primaryRepresentation=Name
  (captions+=CaptionForIndividualConcept)*;
VerbConcept:
  primaryRepresentation=VerbConceptWording
  (captions+=CaptionForVerbConcept)*;
```

VerbConceptWording is the general rule for representing verb concepts (Table 3.4). SBVR verb concepts can have different structures: binary verb concepts, characteristics, and noun forms. They use cross-references to representations of general concepts (i.e., terms). They help to ensure that verb concepts are specified using only those general concepts that are specified in the vocabulary. Cross-references are indicated by square brackets in Xtext syntax.

Table 3.4. Rules to represent verb concepts

```
VerbConceptWording:
  SententialForm|NounForm;
SententialForm:
  BinaryVerbConcept|Characteristic;
BinaryVerbConcept:
  placeholder1=[Term]
  verbSymbol=[VerbSymbol]
  placeholder2=[Term];
Characteristic:
  placeholder1=[Term]
  verbSymbol=[VerbSymbol];
NounForm:
  placeholder1=[Term]
  placeholder2=[Term];
```

The example of captionõs rule is presented in Table 3.5. *CaptionForGeneralConcept* is the rule to define available captions of representation of general concepts. The caption rule contains a label and type of value. For example, the synonym of general concept can be defined using syntax of term, whereas the description can be written using free text.

Table 3.5. Example of vocabulary caption rule

```
CaptionForGeneralConcept:
  Description|Synonym| ... ;
Description: "Description: " STRING;
Synonym: "Synonym: " Term;
```

3.3.2 Grammar rules for specifying SBVR business rules

Rules of SBVR business rules grammar are based on representations of atomic formulations that are described in subsection 3.3.4.

In this subsection, the basic structure of business rules and their types are presented. Grammar allows specifying structural, operative, and derivation rules. Structural and operative business rules are distinguished by modal operators. Structural rules are written using alethic (i.e., *It is necessary that*, *It is possible that*, and *It is impossible that*) and operative rules - deontic (i.e., *It is obligatory that*, *It is permitted that*, *It is prohibited that*) modal operators. Derivation rules are used for deriving new facts from existing ones. They are specified using implication.

Therefore, two templates are defined for representing business rules: rules based on statements of atomic formulations for specifying structural and operative rules and rules based on implications (Table 3.6). The first template contains two parts: modal operator and restricting statement. The second one has a modal operator and two statements as consequent and antecedent.

Table 3.6. Rules of representation of business rules

```
Rule:
  RuleBasedOnStatementOfAtomicFormulation|ImplicationRule;
RuleBasedOnStatementOfAtomicFormulation:
  modality=ModalOperator
  statement=RepresentationOfAtomicFormulation;
ImplicationRule:
  modality=ModalOperator
  consequent=RepresentationOfAtomicFormulation
  "if" antecedent=RepresentationOfAtomicFormulation;
```

```
ModalOperator:
  NECESSITY|POSSIBILITY|IMPOSSIBILITY|OBLIGATION|PERMISSION|
  PROHIBITION;
NECESSITY:
  "It is necessary that";
POSSIBILITY:
  "It is possible that";
IMPOSSIBILITY:
  "It is impossible that";
OBLIGATION:
  "It is obligatory that";
PERMISSION:
  "It is permitted that";
PROHIBITION:
  "It is prohibited that";
```

3.3.3 Grammar rules for writing SBVR questions

As well as rules for specifying business rules, rules for writing questions are based on representations of atomic formulations.

In this subsection, the basic structure and types of questions are presented (in accordance with types, analysed in this work and defined in 2.1.5). The two main types of questions are defined in the grammar, they are presented in Table 3.7. The simplest questions intended to find instances of certain type (e.g., *Find persons*, *Find largest_state*, etc.). Ordinary questions contain representation of atomic formulation (e.g., *What persons work_in KTU*).

Table 3.7. Rules of writing questions

```
Question:
  (QuestionToFindInstancesByType|OrdinaryQuestion);
QuestionToFindInstancesByType:
  startKeyword=FIND
  concept=[Term];
OrdinaryQuestion:
  startKeyword=(WH|FIND)
  statement=RepresentationOfAtomicFormulation ("?"|".");
WH:
  "What"|"When"|"Where"|...;
FIND:
  "Find"|"Search_for"|...;
```

3.3.4 Grammar rules for representing atomic formulations

Atomic formulations are logical formulations, used to formulate meaning of restrictions of business rules and questions. In this subsection, representations of atomic formulations are described. For example, in the business rule *It is necessary that person works_in at least 1 organization* its representation of atomic formulation means the following fragment of the rule: *person works_in at least 1 organization.*

Atomic formulations are based on verb concepts and have bindable targets, which allow binding roles to individual concepts or variables. Variables can be further restricted by other logical formulations, for example, cardinality formulations.

Representation of atomic formulation correspond to representation of verb concept with placeholders for each role. A placeholder is a place for an expression that can be replaced by representation of an individual concept (i.e., name), quantifier, or quantity comparison. For example, the representation *event is_organized_by organizer* has two placeholders: *event* and *organizer*. If placeholder *organizer* is replaced by name *Events_Ltd*, statement is as follows: *event is organized_by Events_Ltd*. It represents the atomic formulation with the second role bound to an individual concept, which means that each event is organized by organizer *Events_Ltd*. If the placeholder is not replaced (i.e., *event is_organized_by organizer*), it represents atomic formulation with both roles bound to variables, which formulates the meaning that each event is organized by some organizer. In this case, representation of the atomic formulation correspond to the representation of the verb concept.

Atomic formulations can be restricted by several logical formulations that are related using conjunction or disjunction operators. In such cases, the representation is compound. For example, *event is_organized_by Events_Ltd and event has number_of_sold_tickets greater_than 100* is the representation of two atomic formulations related with conjunction. This representation can also be written in a way that is more elegant, by omitting on of the placeholders: *event is_organized_by Events_Ltd and has number_of_sold_tickets greater than 100.*

Compound representations are also used to represent meaning, when atomic formulations are restricted by other logical formulations. For example, *organizer organizes event that takes_place_in sports_arena Snow* represents the meaning of when the second role (*event*) of first atomic formulation is restricted by the other logical formulation (represented by *event takes_place_in sports_arena Snow*).

Table 3.8 presents types of representations of atomic formulations. Each of them are further described in detail.

Table 3.8. Representation of atomic formulations

```
RepresentationOfAtomicFormulation:
  BothPlaceholdersNotReplaced|
  1stPlaceholderReplacedByName|
  2ndPlaceholderReplacedByName|
  BothPlaceholdersReplacedByName|
  2ndPlaceholderReplacedByQuantityRestriction|
  2ndPlaceholderReplacedByQuantification;
```

3.3.4.1 Representations with both placeholders not replaced

It is the simplest type of representation, which correspond to representation of verb concept. It represents atomic formulation, which is based on a verb concept, having both roles bound to variables. The meaning of this atomic formulation is *true*, when all referent things of the first variable have corresponding referents of the second variable. For example, representation *organizer, which organizes event*, represents the meaning that each organizer organizes at least one event. The grammar rule and the example of such representation are presented in Table 3.9.

Table 3.9. Rule and example of representation with both placeholders not replaced

```
BothPlaceholdersNotReplaced:
  placeholder1=[Term]? (PRONOUN)?
  verbSymbol=[VerbSymbol]
  placeholder2=[Term]
  ((conj=CONJUNCTION|disj=DISJUNCTION)?
  restriction=RepresentationOfAtomicFormulation)?;
```

```
event takes_place_in event venue
```

Rules of representations of atomic formulations allow writing compound representations using recursive restrictions, expressed by optional call of *RepresentationOfAtomicFormulation* (i.e., any type of representation).

3.3.4.2 Representations with placeholders replaced by names

Representations of this type intended for atomic formulations that have at least one role bound to an individual concept. The meaning of this atomic formulation is *true*, when each referent of the variable is an individual concept, bound to the role. For example, statement *event is_organized_by organizer Events_Ltd* represents the meaning, selecting events that are organized by the organizer *Events_Ltd*. Grammar rules and examples for this type of representations are presented in Table 3.10 - Table 3.12.

Table 3.10. Rule and example of representation when first role is replaced by name

```
1stPlaceholderReplacedByName :
  placeholder1=[Term]? role1Replacement=Name (PRONOUN)?
  verbSymbol=[VerbSymbol]
  placeholder2=[Term]
  (((conj=CONJUNCTION|disj=DISJUNCTION)?
  restriction=RepresentationOfAtomicFormulation)?)*;
```

```
organizer Events Ltd organizes event
Events Ltd organizes event
```

Table 3.11. Rule and example of representation when second role is replaced by name

```
2ndPlaceholderReplacedByName:
  placeholder1=[Term]? (PRONOUN)?
  verbSymbol=[VerbSymbol]
  placeholder2=[Term]? role2Replacement=Name
  ((conj=CONJUNCTION|disj=DISJUNCTION)?
  restriction=RepresentationOfAtomicFormulation)?;
```

```
event that takes_place_in sports arena Snow
event takes_place_in Snow
```

Table 3.12. Rule and example of representation when both roles are replaced by names

```
BothPlaceholdersReplacedByName:
  placeholder1=[Term]? role1Replacement=Name (PRONOUN)?
  verbSymbol=[VerbSymbol]
  placeholder2=[Term]? role2Replacement=Name
  ((conj=CONJUNCTION|disj=DISJUNCTION)?
  restriction=RepresentationOfAtomicFormulation)?;
```

```
organizer Events Ltd organizes event Tern
Events Ltd organizes Tern
```

3.3.4.3 Representations with the second placeholder replaced by quantity restriction

In this type of representations, the second placeholder is replaced by expression of quantity restriction. This type of representations is intended for atomic formulations, when the first role is bound to a variable and the second one is restricted by a logical formulation of quantity restriction. It formulates the meaning, which is *true* for each referent of the variable, which satisfies the restriction. For example, *event that has number_of_sold_tickets greater_than 100* represents the meaning, that each selected event has number of sold tickets greater than 100. The grammar rule and examples are presented in Table 3.13. Keywords to express various quantity restrictions are presented in Table 3.14.

Table 3.13. Rule and examples of representation when second role is replaced by expression of quantity restriction

```
2ndPlaceholderReplacedByQuantityRestriction:
  placeholder1=[Term]? (PRONOUN)?
  verbSymbol=[VerbSymbol]
  placeholder2=[Term]
  role2Replacement=QuantityRestriction;
```

```
event that has number of sold tickets that is_greater_than 100
event has number of sold tickets that is_less_than 100
number of sold tickets of event equals 100
```

Table 3.14. Keywords to express quantity restrictions

```
QuantityRestriction:
  GREATER_THAN|GREATER_OR_EQUAL|LESS_THAN|LESS_OR_EQUAL|EQUAL;
terminal GREATER_THAN:
  "is_greater_than" value=DIGIT;
terminal GREATER_OR_EQUAL:
  "is_greater_or_equal_to" value=DIGIT;
terminal LESS_THAN :
  "is_less_than" value=DIGIT;
terminal LESS_OR_EQUAL:
  "is_less_or_equal_to" value=DIGIT;
terminal EQUALS :
  "equals" value=DIGIT;
```

3.3.4.4 Representations with placeholders replaced by quantification representations

These types of representations are used to represent atomic formulation, when the second role is restricted by a cardinality restriction. Therefore, the second placeholder of this representation is replaced by an expression of quantification. For example, representation *organizer organizes at_least 2 events* represents the meaning, which is true for each organizer that organizes at least two events. Grammar rules for statements with quantifications and available quantifications are presented in Table 3.15. Keywords to express various quantification restrictions are presented in Table 3.16.

Table 3.15. Rule and example of representation when second role is replaced by expression of quantification

```
2ndPlaceholderReplacedByQuantification:
  placeholder1=[Term]
  verbSymbol=[VerbSymbol]
  role2Replacement=Quantification
  placeholder2=[Term];
```

```
event is_organized_by exactly 1 organizer
```

Table 3.16. Keywords to express quantification restrictions

```
Quantification:
  UNIVERSAL_QUANTIFICATION|AT_LEAST_N_QUANTIFICATION|
  AT_MOST_N_QUANTIFICATION| EXACTLY_N_QUANTIFICATION|
  NUMERIC_RANGE_QUANTIFICATION;
terminal UNIVERSAL_QUANTIFICATION:
  ("a"|"an"|"each")?;
terminal AT_LEAST_N_QUANTIFICATION:
  "at_least" valueN=DIGIT;
terminal AT_MOST_N_QUANTIFICATION:
  "at_most" valueN=DIGIT;
terminal EXACTLY_N_QUANTIFICATION :
  "exactly" valueN=DIGIT;
terminal NUMERIC_RANGE_QUANTIFICATION :
  "at_least" valueN=DIGIT "and_at_most" valueM=DIGIT;
```

3.4 Natural language interface to ontologies

Another part of the solution is NLI to ontologies is that it accepts natural language questions, analyses them and transforms to SPARQL queries. NLI can only answer question starting with *WH* or *find* keywords. *Yes/no* questions are not accepted. Answering such questions is complicated, because ontologies use open world assumption and the absence of required data does not give rise to a negative response. Such questions could only be used to check, if a certain fact is correct (e.g., *whether person had ever been a prime minister?*).

The conception of NLI to ontologies is presented in Figure 3.3. It contains components for writing natural language questions, analysing them and transforming to SPARQL queries. Components with a darker background (i.e., autocomplete suggestions, clarification dialog, heuristic algorithms of question analysis) and morphological library are dependant from language.

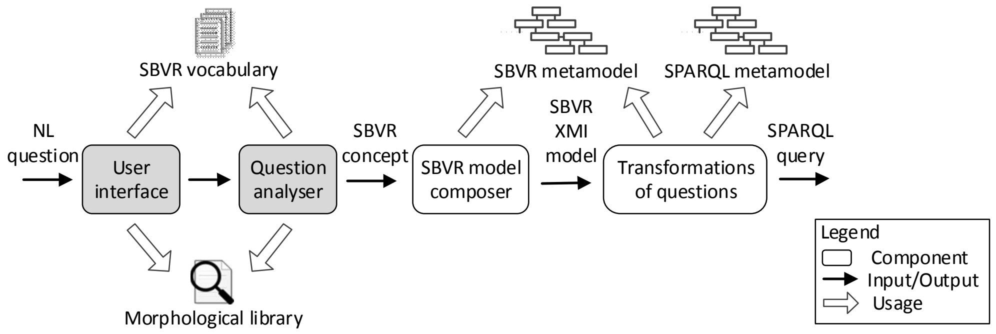


Figure 3.3. Components of NLI to ontologies

The user interface is intended for writing questions and presenting results. This component provides autocomplete and guides the user writing the question word after word. Suggestions of autocomplete are generated using general and verb concepts from the SBVR vocabulary. The SBVR vocabulary can contain synonyms to allow questions that are more varied. SBVR generalizations allows the writing of abstract or specific questions. The user interface also shows a clarification dialog, when a question is ambiguous. Generating suggestions of autocomplete and clarification dialog should be adjusted according to the used language.

When the structure of the ontology is complex and does not directly correspond formulations of questions, formal SBVR definitions are used to derive concepts used for questioning. As a result, SBVR concepts, which are required only in definitions and cannot be used to formulate any natural language question (e.g., *person gave speech*, *speech has speech_object*, etc.), often appears in the vocabulary. Such concepts (i.e., systemic) should be hidden from users and cannot appear in the suggestions of autocomplete. This can be done by splitting the SBVR vocabulary into two parts. The first part containing systemic concepts and the second one contains concepts that can be used to formulate questions and appear in autocomplete.

When a question is written, the analysis of question starts. This component is described in detail in the next subsection.

3.4.1 Analysis of questions

Since NLI interprets and answers only those questions that are based on SBVR vocabulary concepts, the goal of question analysis is to find the SBVR concept(s) that a question is based on. The analysis is performed using empirical rules that should be adjusted according to the language. Main steps of the analysis and their sequence are presented in Figure 3.4.

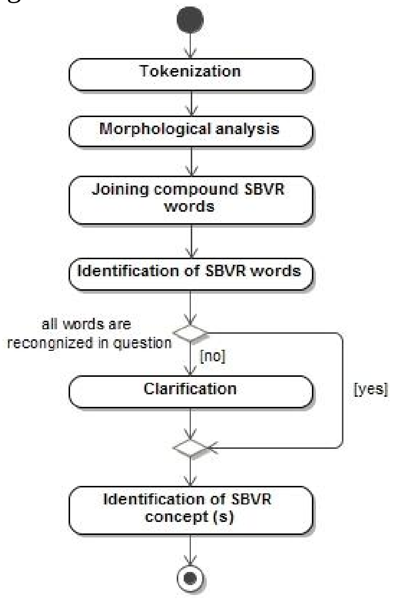

Tokenization
Morphological analysis
Joining compound SBVR words
Identification of SBVR words
all words are recongnized in question
[no]
Clarification
[yes]
Identification of SBVR concept (s)

Figure 3.4. Steps of question analysis

Steps are described further:

- *Tokenization* ï question is split into separate words;
- *Morphological analysis* ï tokens are analysed morphologically by finding the part of speech, lemma and other information using the morphological library. This step requires a specific morphological library, such as Stanford parser [58] for English;
- *Joining compound SBVR words* ï this step is performed matching tokens with words of SBVR vocabulary. Compound SBVR words (e.g., *large_state*, *works_in*, etc.) are searched and joined to a single token. If a question starts with a preposition (e.g., Lit.: *Su kuo*, Eng.: *With what*), the system attempts to connect such preposition with a verb (e.g., Lit.: *susitiko_su*, Eng.: *met_with*). In addition, it is verified, if such a compound is defined in the SBVR business vocabulary;
- *Identification of SBVR words* ï each token that is found in the SBVR vocabulary as term, verb, or proper name is marked as a SBVR word;
- *Clarification* ï clarification is used when some words of the question are not recognized as a SBVR word and morphological analysis does not provide any helpful information (e.g., word is name of place, surname, etc.). User

can clarify unrecognized word as: (1) synonym of another SBVR vocabulary word; (2) proper name of a certain type. User can write lemma of unrecognized proper name; (3) stop word that should be skipped in further interpretation. A clarification dialog is also generated in cases of ambiguities, when several equal interpretations of a questions are available. When possible, ambiguities are resolved automatically from the context. For example, although Mississippi can mean state or river, in the question *What states border Mississippi* it is obvious that the state is meant.

- *Identification of SBVR concept(s)* ï this step is required to find the concept(s) that a question is based on. For example, the question *What states border Illinois* is based on the SBVR verb concept *state borders state*, while question *Find cities* is based on the general concept *city*. When a question starts with a pronoun (e.g., Lit.: *Kas*, *KŃ*, *Kuo*, Eng.: *Who*, *What*, *Wherewith*) followed by a verb, the noun, representing the first role of the verb concept is skipped (e.g., Lit.: *KŃ kalbǟjo asmenys*, Eng.: *What did persons say*). In such cases, the correct verb concept should be found by selecting verb concepts with corresponding verb and analysing morphological features of their roles. When a question is ambiguous (i.e., several verb concepts available), a clarification dialog should be generated.

Question analyser also identifies if the question is for counting and passes it as a parameter for the query transformation component to use the appropriate transformation rules.

After analysing a natural language question and identifying the SBVR concept(s), a SBVR XMI model of the question is generated calling XMI generator of SBVR SLE. If the identified concept has definition rules, the XMI generator uses definition in the model of the question. Finally, the SBVR XMI model is transformed to a SPARQL query using model transformation rules that are presented in a further subsection. As the SBVR model holds the meaning of a question, transformation rules are independent from language.

3.4.2 Rules to transform questions to SPARQL

Transformation rules are based on generic transformation patterns presented in [109]. They are model driven, conforming to the SBVR metamodel (subsections 2.2.2, 2.2.3), and SPARQL syntax metamodel (subsection 2.2.6). The details of implementation of these rules are described in subsection 4.2.3.

There are six types of transformable questions, conforming types, presented in 2.1.5:

- Questions to find individuals of certain type (e.g., *Find persons*);
- Questions with modifier attachments (e.g., *What states that border Illinois?*);
- Questions to count values (e.g., *How many states border Illinois?*);
- Questions with cardinality restriction (e.g., *Find states that border at least 3 states.*);
- Questions with numerical comparison (e.g., *Find cities that have population greater than 100000.*);

- Questions to find minimum or maximum values (e.g., *Find state that has largest population.*).

To transform questions to SPARQL queries, nine transformation rules were defined; as described in the further subsections. Each rule creates a certain part of the query. They are called by different algorithms (Figure 3.6 ï Figure 3.10) depending on the type of question.

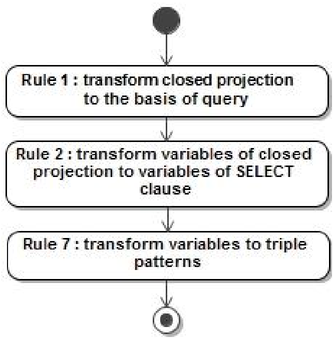

Rule 1 : transform closed projection to the basis of query
Rule 2 : transform variables of closed projection to variables of SELECT clause
Rule 7 : transform variables to triple patterns

Figure 3.5. Algorithm for transforming questions to find individuals of certain type

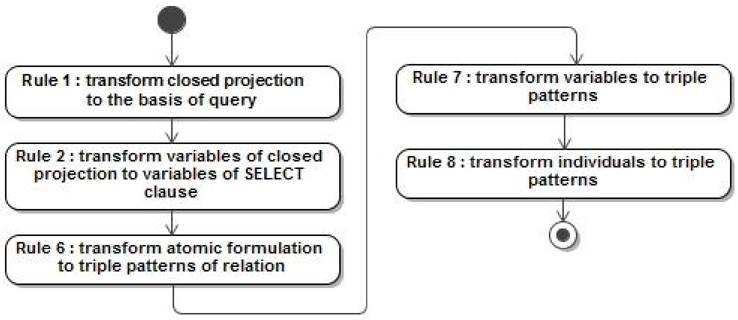

Rule 1 : transform closed projection to the basis of query
Rule 2 : transform variables of closed projection to variables of SELECT clause
Rule 6 : transform atomic formulation to triple patterns of relation
Rule 7 : transform variables to triple patterns
Rule 8 : transform individuals to triple patterns

Figure 3.6. Algorithm for transforming questions with modifier attachments

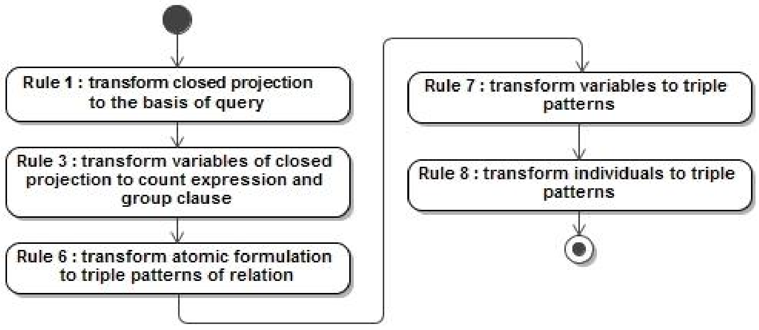
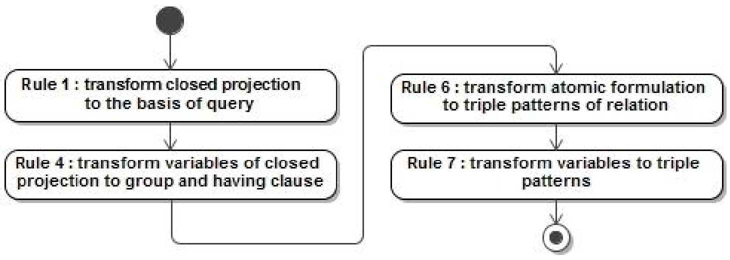
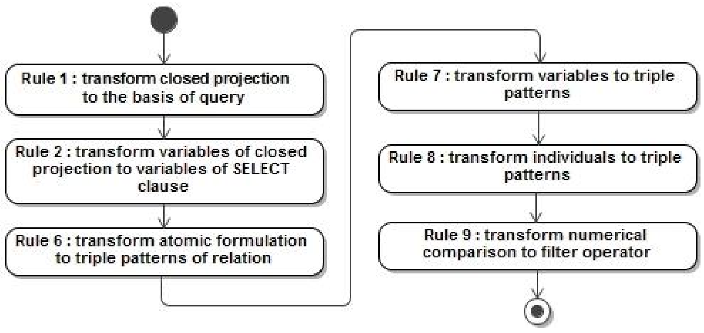

Rule 1 : transform closed projection to the basis of query
Rule 3 : transform variables of closed projection to count expression and group clause
Rule 6 : transform atomic formulation to triple patterns of relation
Rule 7 : transform variables to triple patterns
Rule 8 : transform individuals to triple patterns

Figure 3.7. Algorithm for transforming questions to count values



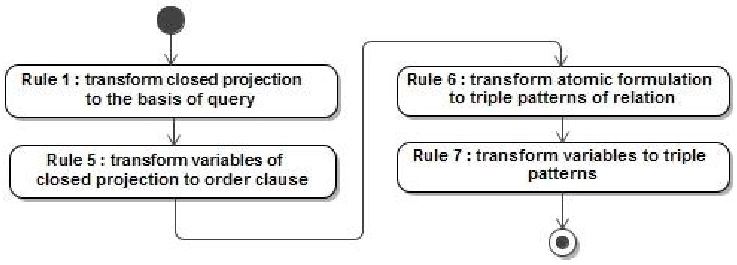
Rule 1 : transform closed projection to the basis of query
Rule 4 : transform variables of closed projection to group and having clause
Rule 6 : transform atomic formulation to triple patterns of relation
Rule 7 : transform variables to triple patterns

Figure 3.8. Algorithm for transforming questions with cardinality restrictions



Rule 1 : transform closed projection to the basis of query
Rule 2 : transform variables of closed projection to variables of SELECT clause
Rule 6 : transform atomic formulation to triple patterns of relation
Rule 7 : transform variables to triple patterns
Rule 8 : transform individuals to triple patterns
Rule 9 : transform numerical comparison to filter operator

Figure 3.9. Algorithm for transforming questions with numerical comparisons

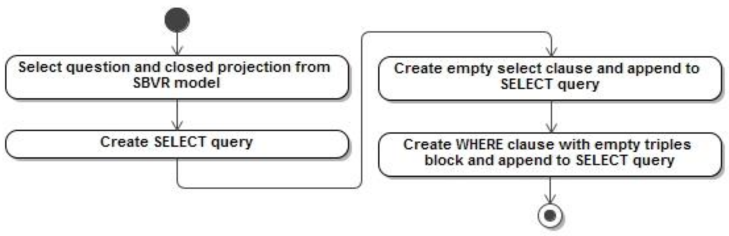


Rule 1 : transform closed projection to the basis of query
Rule 5 : transform variables of closed projection to order clause
Rule 6 : transform atomic formulation to triple patterns of relation
Rule 7 : transform variables to triple patterns

Figure 3.10. Algorithm of transforming questions with minimum or maximum restrictions

3.4.2.1 Rule 1: transform closed projection to the basis of query

This rule is called first for all types of questions. It uses closed projection to create the top-level element of the query with empty SELECT and WHERE clause elements that are filled when using subsequent rules. Steps of this rule are presented in Figure 3.11. Table 3.17 presents SPARQL model fragment created by this rule and the example.



Select question and closed projection from SBVR model
Create SELECT query
Create empty select clause and append to SELECT query
Create WHERE clause with empty triples block and append to SELECT query

Figure 3.11. Steps of Rule 1

Table 3.17. Model fragment and example created by Rule 1

```
in: SBVR:ClosedProjection
out: SPARQL:
 SelectQuery (
  SelectClause ( ),
  WhereClause (
   GroupGraphPatternSub (
    TriplesBlock ( )
   )
  )
 )
```

```
SBVR: What rivers run_through states?
SPARQL:
  SELECT
  WHERE { ... }
```

3.4.2.2 Rule 2: transform variables of closed projection to variables of SELECT clause

Rule 2 transforms variables of closed projection to variables of SELECT clause. Names are set by expressions of general concepts that projection variables range over. Steps of this rule are presented in Figure 3.12. Table 3.18 presents SPARQL model fragment created by this rule and the example.

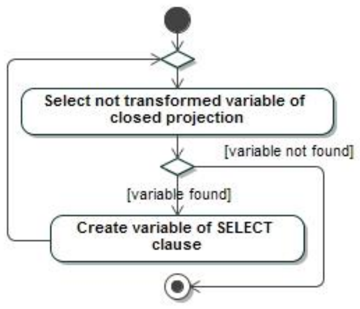

Select not transformed variable of closed projection
[variable not found]
[variable found]
Create variable of SELECT clause

Figure 3.12. Steps of Rule 2

Table 3.18. Model fragment and example created by Rule 2

```
in: SBVR:ClosedProjection
out: SPARQL:
 SelectClause (
  foreach in.variable as v
   Var(name=v.rangedOver.expr)
 )
```

```
SBVR: What rivers run_through states?
SPARQL:
  SELECT
    ?river
    ?state
  WHERE { ... }
```

3.4.2.3 Rule 3: transform variables of closed projection to count expression and group clause

This rule is used for transforming questions to count values. Since the SBVR metamodel is not capable to represent questions with counting, SBVR XMI models for such questions are created in the same way as simple questions with modifier attachments. Therefore, transformation accepts the parameter to indicate questions for counting and calls appropriate rules. The transformation creates the COUNT function from the first variable of a closed projection and solution modifier with GROUP BY operator from the second variable. Steps of this rule are presented in Figure 3.13. Table 3.19 presents SPARQL model fragment created by this rule and the example.

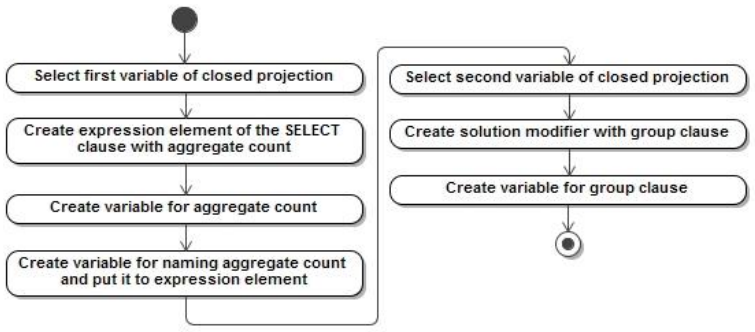

Select first variable of closed projection
Create expression element of the SELECT clause with aggregate count
Create variable for aggregate count
Create variable for naming aggregate count and put it to expression element
Select second variable of closed projection
Create solution modifier with group clause
Create variable for group clause

Figure 3.13. Steps of Rule 3

Table 3.19. Model fragment and example created by Rule 3

```
in: SBVR:ClosedProjection
out: SPARQL:
SelectClause(
 ExpressionAsVarElement1(
  AggregateCount(
   Var(name=in.Var[0].rangedOver.expr)
  ),
  Var(name=in.Var[0].rangedOver.expr + "_count")
 )
),
SolutionModifier(
 GroupClause(
  Var(name=in.Var[1].rangedOver.expr)
 )
)
```

```
SBVR: How_many rivers run_through Illinois?
SPARQL:
  SELECT (COUNT(?river_i) as ?river_count)
  WHERE { ... }
  GROUP BY ?state_i
```

3.4.2.4 Rule 4: transform variables of closed projections to group and having clauses

This rule is selected to transform variables of closed projection, restricted by cardinality formulation. It creates COUNT function and solution modifier with GROUP BY and HAVING operators from first and second variables of closed projection. Steps of this rule are presented in Figure 3.14. Table 3.20 presents SPARQL model fragment created by this rule and the example.

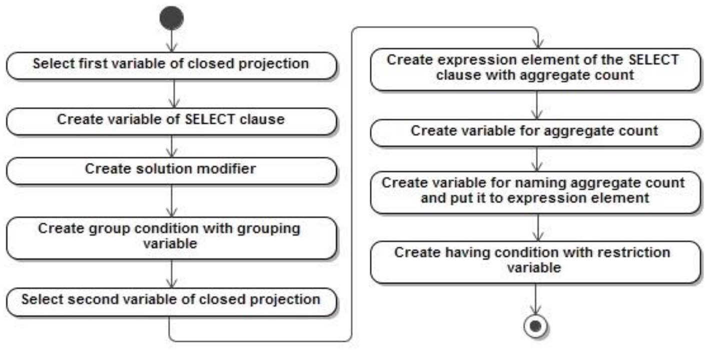

Select first variable of closed projection
Create variable of SELECT clause
Create solution modifier
Create group condition with grouping variable
Select second variable of closed projection
Create expression element of the SELECT clause with aggregate count
Create variable for aggregate count
Create variable for naming aggregate count and put it to expression element
Create having condition with restriction variable

Figure 3.14. Steps of Rule 4

Table 3.20. Model fragment and example created by Rule 4

in: SBVR: ClosedProjection
out: SPARQL:
 SelectClause(
  Var(name=in.Var[0].rangedOver.expr),
  ExpressionAsVarElement1(
   AggregateCount(
    Var(name=in.Var[1].rangedOver.expr + "_i")
   ),
   Var(name=in.Var[1].rangedOver.expr + "_count")
)
 ),
 SolutionModifier(
  GroupCondition(
   Var(name=in.Var[0].rangedOver.expr + "_i")
  ),
  HavingCondition(
   Var(name=in.Var[1].rangedOver.expr + "_count")
   ComparisonSign
   INTEGER
)
)

SBVR: Which rivers run_through at_least 3 states?
SPARQL:
 SELECT
  ?river_i
  (count(?state_i) as ?state_count)
 WHERE { ... }
 GROUP BY ?river_i
 HAVING(?state_count >= 3)

Depending on the type of cardinality quantifications, different comparison symbols are created in the HAVING condition. Cardinality quantifications and corresponding symbols are presented in Table 3.21.

Table 3.21. Cardinality quantifications and corresponding symbols

Cardinality quantification	**Comparison symbol**
`not AtLeastNQuantification`	`<`
`AtMostNQuantification`	`<=`
`EqualsNQuantification`	`=`
`not EqualsNQuantification`	`!=`
`AtLeastNQuantification`	`>=`
`not AtMostNQuantification`	`>`

3.4.2.5 Rule 5: transform variables of closed projection to order clause

This rule is called when the restricting atomic formulation is based on *is_property_of* verb concept (e.g., population of city), and such property is additionally restricted by the minimum or maximum formulation. This rule creates variables of the SELECT clause and the solution modifier with ORDER and LIMIT clauses from variables of a closed projection. Steps of this rule are presented in Figure 3.15. Table 3.22 presents SPARQL model fragment created by this rule and the example.

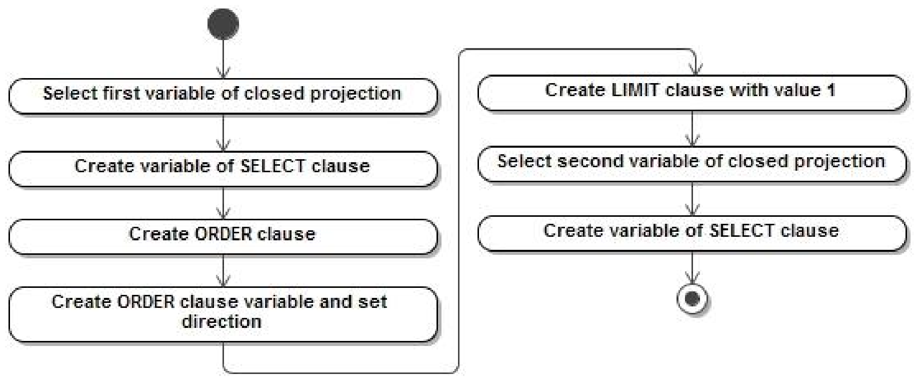

Select first variable of closed projection
Create variable of SELECT clause
Create ORDER clause
Create ORDER clause variable and set direction
Create LIMIT clause with value 1
Select second variable of closed projection
Create variable of SELECT clause

Figure 3.15. Steps of Rule 5

Table 3.22. Model fragment and example created by Rule 5

```
in: SBVR:ClosedProjection
out: SPARQL:
 SelectClause(
  Var(name=in.Var[0].rangedOver.expr),
  Var(name=in.Var[1].rangedOver.expr),
 ),
 SolutionModifier(
  OrderClause(
   OrderDirection,
   iriOrFunction(
    iri="xsd:float",
    argList=Var(name=
      in.Var[0].rangedOver.expr)
    )
  )
  LimitClause(
   integer=1
  )
 )
```

```
SBVR: What city has largest population?
SPARQL:
 SELECT
  ?city_i
  ?population_i
 WHERE { ... }
 ORDER BY DESC(xsd:float(?population_i))
 LIMIT 1
```

Depending on whether it is a minimum or maximum restriction, ordering is ascending or descending.

3.4.2.6 Rule 6: transform atomic formulation to triple patterns of relation

The atomic formulation is based on the verb concept and is used to express restrictions of question. Rule 6 transforms atomic formulation and its verb concept into two triple patterns to express the relation of the verb concept.

The first one is the main triple pattern expressing relation. It has variables in all three positions of subject, predicate, and object. The name of the predicateôs variable is set by the expression of verb conceptôs verb symbol. Names of variables in positions of subject and object are set by roles of verb concept and suffixed with ñ_iò.

The second triple pattern is used to identify the label of relation. It is set by the expression of verb conceptôs sentential form and language tag.

When both triple patterns are created, they are appended to triples block. Steps of Rule 6 are presented in Figure 3.16. Table 3.23 presents SPARQL model fragment created by this rule and the example. If questions use synonymous forms, the preferred verb concepts should be used [111].

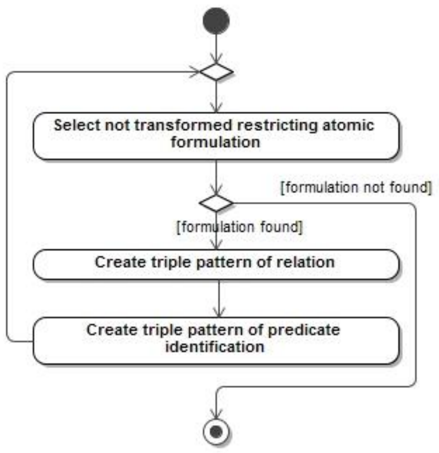

Select not transformed restricting atomic formulation
[formulation not found]
[formulation found]
Create triple pattern of relation
Create triple pattern of predicate identification

Figure 3.16. Steps of Rule 6

Table 3.23. Model fragment and example created by Rule 6

```
in: SBVR:AtomicFormulation
out: SPARQL:
WhereClause(
 GroupGraphPatternSub(
  TriplesBlock(
   TriplesSameSubjectPath(
    Var(name=in.verbConcept.role[0].expr +
     “_i“),
    PropertyListPathNotEmpty(
     Var(name=in.verbConcept.verbSymb.expr),
     Var(name=in.verbConcept.role[1].expr +
      “_i“)
   )
  )
 ),
 TriplesSameSubjectPath(
  Var(name=in.verbConcept.expr),
  PropertyListPathNotEmpty(
   IRIREF=”:label_sbvr”,
   STRING_LITERAL=in.verbConcept.
    sentForm.expr + “@” + lang
  )
 )
)
```

```
SPARQL:
 ?city_i ?is_in ?state_i .
 ?is_in :sbvr_label “city is_in state“@en .
```

3.4.2.7 Rule 7: transform variables to triple patterns

Roles of verb concept of atomic formulation can be bound to variables or individual concepts. If the role is bound to variable, it is transformed to two triple patterns using this rule. The first one is *rdf:type* relation between variable and its type (i.e., variable with suffix ñ_cò) and the second one identifies type by label. This rule is also called transforming questions for finding individuals of certain type to identify types of searched variables.

Rule 7 is presented in Figure 3.17. Table 3.24 presents SPARQL model fragment created by this rule and the example.

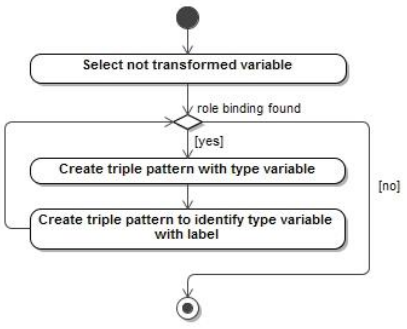

Select not transformed variable
role binding found
[yes]
Create triple pattern with type variable
[no]
Create triple pattern to identify type variable with label

Figure 3.17. Steps of Rule 7

Table 3.24. Model fragment and example created by Rule 7

```
in: SBVR:Variable
out: SPARQL:
 TriplesSameSubjectPath (
  Var(name=in.rangedOver.expr + "_i"),
  PropertyListPathNotEmpty(
   IRIREF="rdf:type",
   Var(name=in.rangedOver.expr + "_c"),
  )
 ),
 TriplesSameSubjectPath (
  Var(name=in.rangedOver.expr + "_c"),
  PropertyListPathNotEmpty(
   IRIREF="rdfs:label",
   STRING_LITERAL=in.rangedOver.expr+"@"+
    lang
  )
 )
```

```
SBVR: What rivers run_through states?
SPARQL:
 ?river_i rdf:type ?river_c .
 ?river_c rdfs:label "river"@en .
 ?state_i rdf:type ?state_c .
 ?state_c rdfs:label "state"@en .
```

3.4.2.8 Rule 8: transform individuals to triple patterns

Rule 8 is used to transform individual concepts, bound to roles. It creates three triple patterns with a filter operator. The first two triple patterns are the same as rule 7 creates. The third one defines the variable of a searched individual label and the filter element used to filter individuals by label. Figure 3.18 presents steps of this rule. Table 3.25 presents SPARQL model fragment created by this rule and the example.

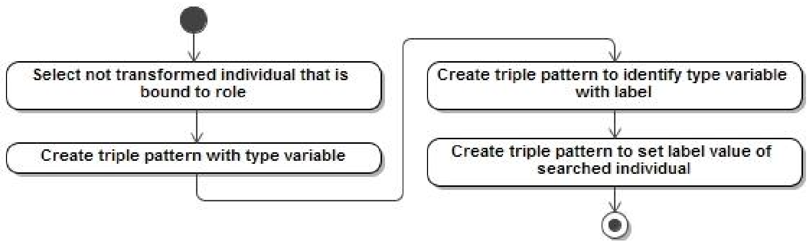

Select not transformed individual that is bound to role
Create triple pattern with type variable
Create triple pattern to identify type variable with label
Create triple pattern to set label value of searched individual

Figure 3.18. Steps of Rule 8

Table 3.25. Model fragment and example by Rule 8

```
in: SBVR:IndividualConcept
out: SPARQL:
 TriplesSameSubjectPath (
  Var(name=in.general.expr + "_i"),
  PropertyListPathNotEmpty(
   IRIREF="rdf:type",
   Var(name=in.general.expr + "_c"),
  )
 ),
 TriplesSameSubjectPath (
  Var(name= in.general.expr + "_c"),
  PropertyListPathNotEmpty(
   IRIREF="rdfs:label",
   STRING_LITERAL=in.general.expr+"@" + lang
  )
 ),
 TriplesSameSubjectPath (
  Var(name=in.general.expr + "_i"),
  PropertyListPathNotEmpty(
   IRIREF="rdfs:label",
   Var(name=in.general.expr + "_v"),
  )
 ),
 RegexExpression(
  Var(name=in.general.expr + "_v")
  pattern=in.expr
 )
```

```
SBVR: What rivers run_through Illinois?
SPARQL:
 ?state_i rdf:type ?state_c .
 ?state_c rdfs:label "state"@en .
 ?state_i rdfs:label ?state_v .
 FILTER regex(?state_v, "Illinois")
```

3.4.2.9 Rule 9: transform numerical comparison to filter operator

Rule 9 defines transformation of questions with quantity restrictions, expressed by numerical comparisons of values of data properties. In SBVR models, numerical comparisons are expressed by atomic formulations, based on particular verb concepts (e.g., *number1 is_greater_than number2*). This restriction is transformed to the FILTER operator in the WHERE clause. Steps of this rule are presented in Figure 3.19. Table 3.26 presents SPARQL model fragment created by this rule and the example.

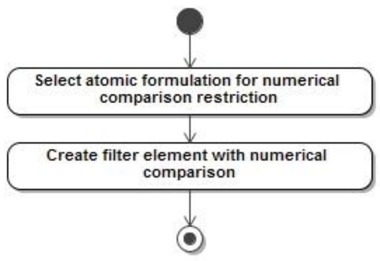

Select atomic formulation for numerical comparison restriction
Create filter element with numerical comparison

Figure 3.19. Steps of Rule 9

Table 3.26. Model fragment and example created by Rule 9

```
in: SBVR:AtomicFormulation
out: SPARQL:
 WhereClause(
  RelationalExpression(
   Var(name=in.Var[1].rangedOver.expr)
   ComparisonSym
   INTEGER
  )
 )
```

```
SBVR: What cities has population less_than 30000?
SPARQL:
 SELECT
  ?city_i
  ?population_i
 WHERE {
  ...
  FILTER(?population_i < 30000)
 }
```

4 IMPLEMENTATION OF SEMANTIC SEARCH SOLUTION

This section presents prototypes of the SBVR based Semantic search solution and details of implementation. The solution contains two prototypes: SBVR SLE and NLI to ontologies.

4.1 Implementation of SBVR structured language editor

4.1.1 Graphical user interface for SBVR specifications

The prototype SBVR SLE was implemented using the Xtext framework [27], [32], [134], [135] and grammar, presented in subsection 3.3. This framework was used because it allows generating ANTLR [2] based parser, AST metamodel and full-featured Eclipse editor from grammar descriptions without any programming effort. The generated editor is capable of recognizing text, compliant with the defined grammar, and has an auto completion function. Xtext framework allowed implementing the tool with the desired features: the possibilities of evaluating the context of business concepts, adapting the editor to other languages; and extending its capabilities by appending the grammar with new rules and rebuilding the editor [107]. The editor is easily configurable and extensible with new functions, such as syntax highlighting, colouring features, external linguistic libraries, etc.

Xtext framework is integrated with the Eclipse Modelling Framework (EMF) and Eclipse User Interface [25]. Xtext automatically creates EMF based AST from structured language text. In this editor, AST is further used to compose formal SBVR models of business vocabularies, business rules and questions. SBVR models are further exported to SBVR XMI schemas and can be used by other tools for transformations. In this work, the SBVR XMI schemas are transformed to SPARQL queries.

Xtext uses the context-free grammar, but allows analysing context following the two-phase process: parsing the text in the first phase and using a linking service to establish cross references in the second phase. Therefore, it allows the creation of links between vocabulary concepts (i.e., verb concept roles and general concepts), business rules, and questions. Incorrect references are automatically marked as errors and can be quickly noticed by the user. Furthermore, cross references can be established between concepts, specified in different files of the same SBVR project.

This feature allows the splitting of specifications into separate parts and uses vocabulary concepts in other vocabularies. It also allows the definition of vocabulary of metaconcepts, required for a particular purpose. For example, one can create metavocabulary for OWL 2 transformations, containing specific concepts, such as *transitive_verb_concept*, *symmetric_verb_concept*, etc. These concepts can be used in the main vocabulary to declare, how certain concepts should be transformed (i.e., to transitive or symmetric object properties in OWL 2 ontology). The example of using metavocabulary is presented in Table 4.1.

Table 4.1. Example of using metavocabulary for transformations of SBVR to OWL 2

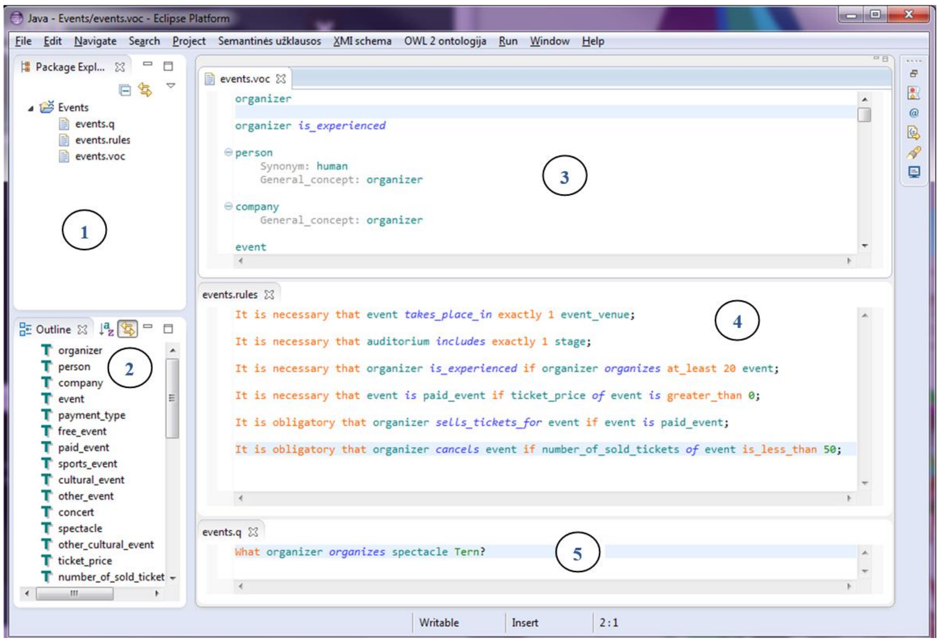
Metavocabulary :

```
verb_concept
transitive_verb_concept
   General_concept: verb_concept
```

Main vocabulary :

```
product consists_of product_part
   Concept_type: transitive_verb_concept
```

The graphical interface of SBVR SLE is presented in Figure 4.1. It contains the following components: (1) package explorer to manage SBVR projects and files; (2) outline block for showing the tree of vocabulary concepts; (3) editing area for business vocabularies, (4) business rules, and (5) questions. Generation of the SBVR XMI model and SBVR transformation to OWL 2 ontology are initiated from the main menu. Editor also contains API that is used by NLI to ontologies, for generating SBVR XMI models of questions.



```
organizer

organizer is_experienced

person
    Synonym: human
    General_concept: organizer

company
    General_concept: organizer

event
```

```
It is necessary that event takes_place_in exactly 1 event_venue;

It is necessary that auditorium includes exactly 1 stage;

It is necessary that organizer is_experienced if organizer organizes at_least 20 event;

It is necessary that event is paid_event if ticket_price of event is greater_than 0;

It is obligatory that organizer sells_tickets_for event if event is paid_event;

It is obligatory that organizer cancels event if number_of_sold_tickets of event is_less_than 50;
```

```
What organizer organizes spectacle Tern?
```

Figure 4.1. Graphical interface of SBVR SLE

4.1.2 Adjusting the editor to different languages

In this work, the editor was adjusted to the English and Lithuanian languages. However, it can also be adjusted to other grammatically similar languages. The adjustment requires appending the grammar with keywords of that language and adjusting the morphological library. The Hunspell based library of JSC Fotonija (created in SemantikaLT project [99]) for Lithuanian was used and Stanford Parser of English language.

The adjustment is more complex for morphologically rich, highly inflected languages. Cross-references between concepts are established analysing syntax. However, concepts can be written using different cases, numbers and other morphological features but they are not syntactically identical. For example, general concepts are usually written using nominative case, but they can be used in genitive or other cases specifying verb concepts. Therefore, the editor must be capable of finding lemmas (i.e., canonical or citation forms) and relate concepts using them. Consider the example representation of a verb concept in Lithuanian *spektaklis vyksta auditorijoje*. The second role in this representation (*auditorijoje*) is in a locative case, while the representation of general concept is in nominative (*auditorija*). By finding the lemma of the locative case (*auditorija*), the verb concept role can be syntactically related with the general concept.

However, finding lemmas proved to be challenging in some cases. The first, difficulties are caused by words, having several parts of speech. For example, the Lithuanian word *kasa* can be considered as a noun or a verb. Lemmas of adjectives and nouns are written in a singular form and a nominative case (*kasa*), whereas lemmas of verbs are in the infinitive form (*kasti*). It creates ambiguous situations, where it is important to decide, which lemma should be used.

Problems also arise tying to relate lemmas with general concepts that are specified in plural forms. Normally, lemmas are in singular forms and cannot be syntactically related with concepts, specified in plural. This problem frequently occurs when analysing compound phrases. Consider the representation of the general concept *scenos_dekoracijos*. The second word of this representation is in the plural form. While the second word of its lemma (*scenos_dekoracija*) is in singular. Therefore, words cannot be syntactically related and concept recognition error occurs. To cope with such errors, a morphological generator is used. This tool generates words in the required morphological forms (e.g., numbers, genders, pronominal forms, etc.). Further, generated words are used to establish cross references between concepts.

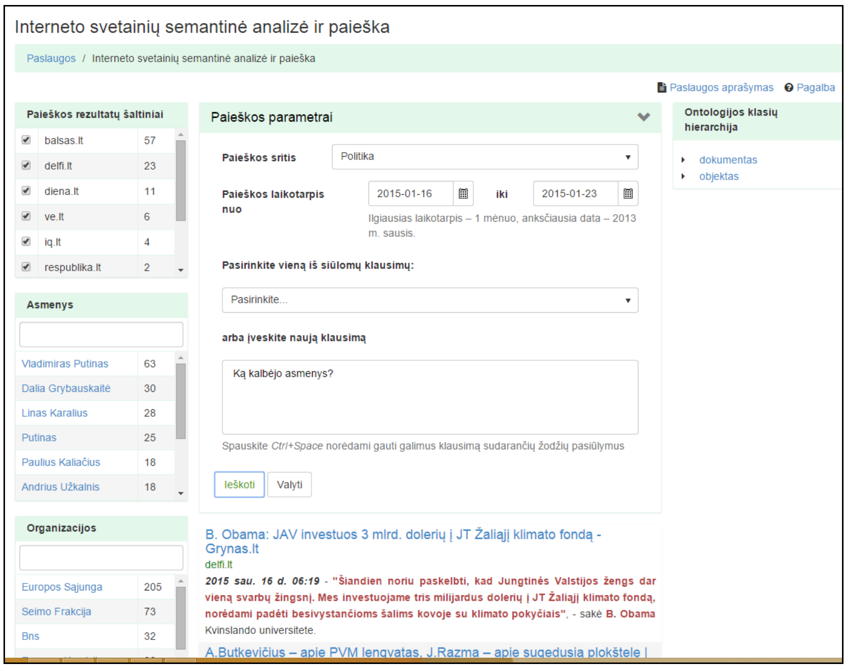
4.2 Implementation of natural language interface to ontologies

The implementation of NLI to ontologies consists of several main components: graphical user interface; component for checking questions and generating autocomplete suggestions; component for analyzing and transforming questions to SPARQL queries and component for executing SPARQL queries and presenting search results to users. It should be noted, that graphical user interface, execution of SPARQL queries and presentation of results are implemented by other authors and is not part of this work. SPARQL queries are executed against ontologies that are filled with individuals using a semantic annotation component. This component annotates articles, extracted from various news portals and stored in corpus storage. However, it is also not a part of this work and is not described in detail.

Results of this research were implemented as two Web services: the first one checks questions and generates autocomplete suggestions; the second one transforms questions to SPARQL queries. These services were used to implement a sematic search service in the SemantikaLT project [99]. The semantic search service is described in more detail in [124]. This subsection describes only parts that were implemented using results of this work.

4.2.1 Graphical user interface

The graphical user interface of NLI to ontologies is presented in Figure 4.2. It is implemented as an internet page and works from a web browser. The interface allows the writing of questions in the Lithuanian language and performs a semantic search in the Lithuanian news corpus from areas of politics, economy, and public administration. Users can write questions to find out about utterances of persons, positions of persons, information related with currencies, unemployment, organizations, etc.


Interneto svetainių semantinė analizė ir paieška

Paslaugos / Interneto svetainių semantinė analizė ir paieška

Paslaugos aprašymas Pagalba

Paieškos rezultatų šaltiniai

✓	balsas.lt	57
✓	delfi.lt	23
✓	diena.lt	11
✓	ve.lt	6
✓	iq.lt	4
✓	respublika.lt	2

Asmenys

Vladimiras Putinas	63
Dalia Grybauskaitė	30
Linas Karalius	28
Putinas	25
Paulius Kaliačius	18
Andrius Užkalnis	18

Organizacijos

Europos Sajunga	205
Seimo Frakcija	73
Bns	32

Paieškos parametrai

Paieškos sritis Politika

Paieškos laikotarpis nuo 2015-01-16 **iki** 2015-01-23

Ilgiausias laikotarpis – 1 mėnuo, ankščiausia data – 2013 m. sausis.

Pasirinkite vieną iš siūlomų klausimų:

Pasirinkite...

arba įveskite naują klausimą

Ką kalbėjo asmenys?

Spauskite *Ctrl+Space* norėdami gauti galimus klausimą sudarančių žodžių pasiūlymus

Ieškoti Valyti

B. Obama: JAV investuos 3 mlrd. dolerių į JT Žaliąjį klimato fondą - Grynas.lt
delfi.lt
2015 sau. 16 d. 06:19 - **"Šiandien noriu paskelbti, kad Jungtinės Valstijos žengs dar vieną svarbų žingsnį. Mes investuojame tris milijardus dolerių į JT Žaliąjį klimato fondą, norėdami padėti besivystančioms šalims kovoje su klimato pokyčiais"**, - sakė **B. Obama** Kvinslando universitete.

A.Butkevičius – apie PVM lengvatas, J.Razma – apie sugedusią plokštelę |

Ontologijos klasių hierarchija

- dokumentas
- objektas

Figure 4.2. Graphical interface of NLI to ontologies in Lithuanian language

To start the search, the user needs to select one of the areas and time interval of publication. Then, they select one of the suggested questions or write their own question. If the question contains errors, the user receives an error messages. Errors can be of two types: syntax errors, when some of words of the question are not recognized or vocabulary errors, when some of words are not found in the vocabulary. The user can correct the question and try to search once again. If the question contains no errors, it is transformed to a SPARQL query, which is executed against the ontology and results are presented to the user.

Currently, the implementation of a semantic search uses tools that store the whole ontology in memory [86]. However, to improve performance, it is worth considering to store ontology individuals in a relational database and query data using SPARQL together with SQL using the method described by Vyġniauskas et al. [127], [108].

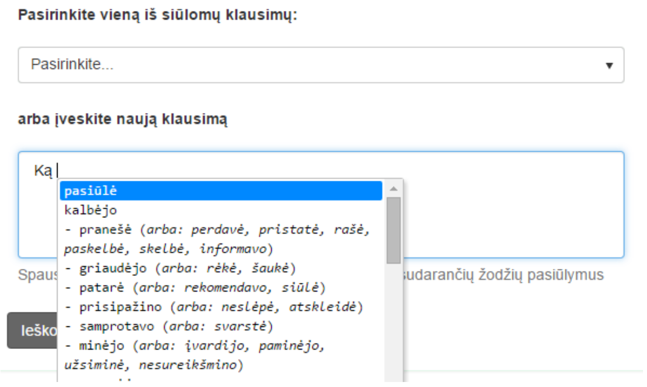
4.2.2 Generating autocomplete suggestions

Autocomplete shows the available words and phrases and helps to formulate questions word after word. It familiarizes users with the structure of questions and concepts that can be questioned.

NLI uses SBVR vocabulary as lexicon. Vocabulary is split into two parts, as described in subsection 0. General concepts and verb concepts from the second vocabulary are used to create autocomplete suggestions. Appropriate morphological forms of words are generated using a morphological generator. For example, if a question starts with a word *Kokie* (Eng. *What*), the second word should be a masculine plural noun (e.g., Lit. *agentai*, *asmenys*; Eng. *agents*, *persons*). These forms are generated from the SBVR vocabulary concepts that are usually specified in singular and nominative case.

Suggestions are generated considering synonyms and synonymous forms. SBVR vocabulary concepts always have the main form of representation and can have several synonyms. During the generation of autocomplete suggestions, the main form is written first and synonyms are listed in parentheses. This gives more information of what results to expect choosing a certain word.

Attention to the hierarchy of concepts is also considered when generating autocomplete. If the concept has child concepts, they are presented below the parent concept; indented by spaces and dashes. This allows users to formulate general or specific questions. The screenshot of autocomplete is presented in Figure 4.3.


Pasirinkite vieną iš siūlomų klausimų:

Pasirinkite...

arba įveskite naują klausimą

Ką

```
pasiūlė
kalbėjo
- pranešė (arba: perdavė, pristatė, rašė,
paskelbė, skelbė, informavo)
- griaudėjo (arba: rėkė, šaukė)
- patarė (arba: rekomendavo, siūlė)
- prisipažino (arba: neslėpė, atskleidė)
- samprotavo (arba: svarstė)
- minėjo (arba: įvardijo, paminėjo,
užsiminė, nesureikšmino)
```

Spaus

sudarančių žodžių pasiūlymus

Figure 4.3. Autocomplete of the semantic search interface

4.2.3 Implementation of transformation rules to SPARQL

Question transformation rules (subsection 3.4.2) are implemented as a model-to-model transformation in accordance with the principles of model-driven development, when a series of transformations are executed over models, usually to decrease the level of abstraction of models [48]. Although transformations of models can be implemented using general-purpose languages, ATL was chosen, as it allows the definition of transformation rules more naturally.

ATL is a domain-specific language to define model transformations [48]. It was created as a proposal of OMG QVT standard for performing model transformations. ATL language allows the use of both declarative and imperative constructs. The preferred style is declarative, because declarative rules are closer to the way the developers intuitively perceive a transformation and can hide complex transformation algorithms in a simple syntax [48]. However, for complex mappings the declarative style can be insufficient. In such situations, developers can use imperative constructs.

ATL transformations can be executed in one direction. During the transformation, the target model is created by navigating a read-only source model. The target model cannot be navigated [48].

Models of transformations are serialized using XMI standard. In this work, the source model of transformation is the SBVR XMI model containing business vocabulary and question. The source model is created by SBVR SLE, which is called through API. The target model is the SPARQL XMI model. This is the first step of transforming questions to SPARQL.

In the second step, the SPARQL XMI model is used to generate the textual query. The generation is performed using Acceleo language. Acceleo is a code generator, the implementation of OMGôs Model to Text Language (MTL) standard. Acceleo takes the XMI model as input and generates the output ï SPARQL query in the text file.

ATL transformations of SBVR to SPARQL are presented in appendix 1. Appendix 2 presents the Acceleo query generation template.

5 EXPERIMENTAL EVALUATION

To evaluate the created solution, several experiments were performed. They were performed and evaluated by the author of the dissertation.

The first experiment was performed to evaluate the suitability of SBVR SLE to write specifications and support all the required SBVR constructions. In another experiment, the editorôs capabilities to write specifications in different languages and in different domains were evaluated. It is important, because this tool is used to configure NLI to ontologies and must allow its multilingualism and portability.

The created NLI to ontologies was evaluated by measuring its correctness. Standard knowledge bases and sets of questions were used to compare the solution with other NLIs. In addition, questions were translated to the Lithuanian language and the multilingualism of the solution was evaluated. In this experiment, two knowledge bases were used to ensure that the created NLI could be configured to question ontologies of different domains.

In the last experiment, questioning complex ontologies (i.e., when questions do not directly correspond to the structure of ontologies) was evaluated.

5.1 Evaluating the completeness of SBVR structured language editor

The goal of this experiment is to evaluate the completeness of the created SBVR structured language grammar and find out whether SBVR SLE is suitable for writing SBVR specifications and generating SBVR XMI schemas. The experiment consists of three parts: evaluating capabilities to specify SBVR business vocabulary, business rules, and questions.

In the first part, the capabilities to specify the following vocabulary constructions were evaluated:

- General concepts;
- Verb concept roles;
- Verb concepts;
- Property associations;
- Characteristics;
- Partitive verb concepts;
- Roles;
- Individual concepts;
- Hierarchy of concepts;
- Segmentations and categorization schemes;
- Facts;
- Definitions.

In the second part, the capabilities to specify structural, operative, and derivation rules, based on following representations of atomic formulations were evaluated:

- With both placeholders not replaced;
- With placeholders, replaced by names;
- With the second placeholder, replaced by quantity restriction;
- With placeholders, replaced by quantification representations.

In the third part, whether the grammar and the created editor is suitable to write *WH* and *find* questions, based on previously listed representations of atomic formulations and questions to find instances of certain type were evaluated.

During the experiment, SBVR XMI schemas were generated. Examples of these schemas are presented in appendix 2.

5.1.1 The investigated model of business domain

For a representative example, the domain of event organization was chosen. The conceptual model of this domain is presented in Figure 5.1. In this model, SBVR general concepts are expressed as UML classes and verb concepts as associations. For the completeness, it contains class hierarchy, generalization between associations, generalization between roles, bidirectional associations, aggregation, cardinality constraints, categorization schemes, segmentations, etc. Therefore, it is suitable to investigate, if the created grammar is complete enough and the editor is capable to use all the required constructions of SBVR.

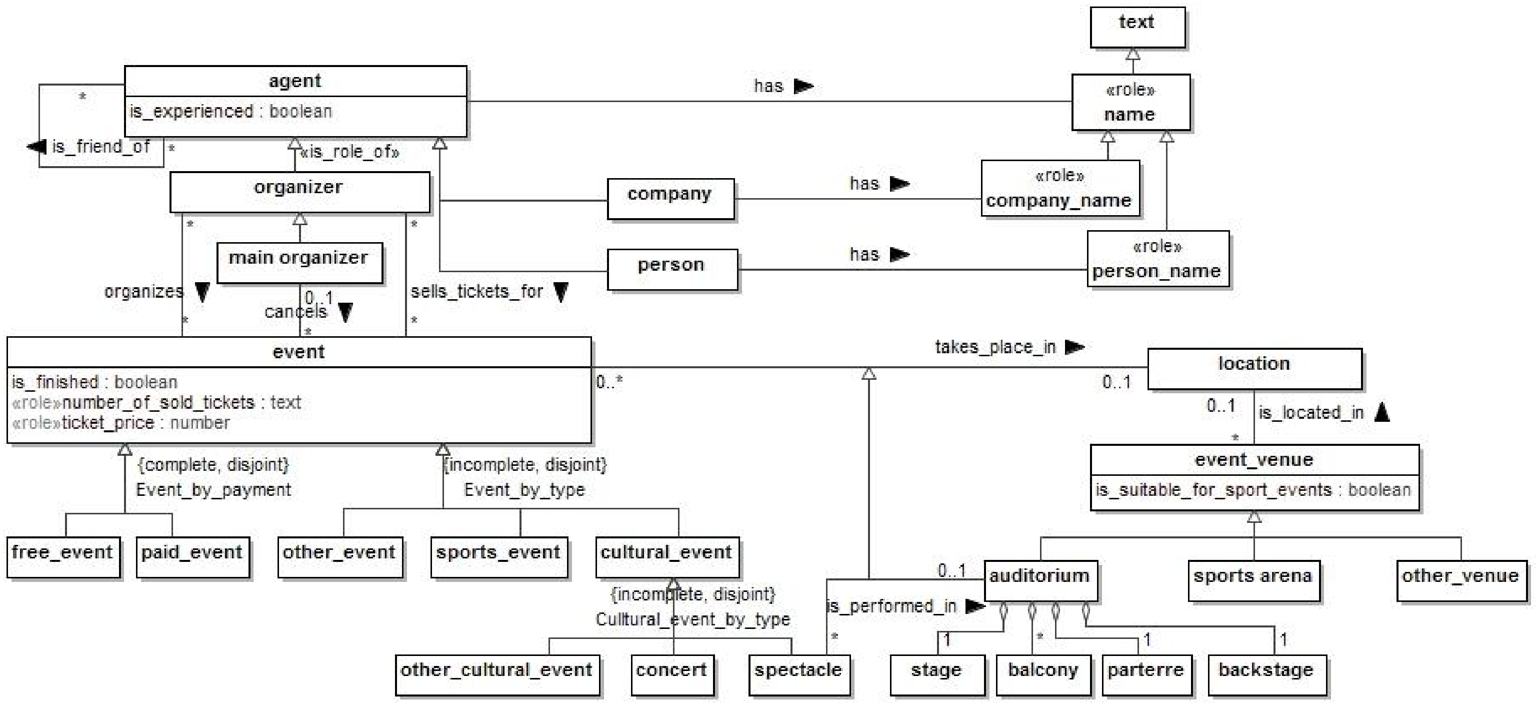


Figure 5.1. Conceptual model of event organization domain presented as UML class diagram

5.1.2 Evaluating the completeness of specifying business vocabularies

In the first part, business vocabulary of the presented domain was specified. Fragments of vocabulary specifications are presented in Table 5.1.

Table 5.1. Fragments of specification of business vocabulary

General concepts

`person`
`General concept: agent`
`Synonym: human`
`Description: "a living human"`
`Company`
`General concept: agent`
`event`

Verb concept roles

`organizer`
`Concept_type: verb concept role`
`General concept: agent`

Verb concepts
organizer *organizes* event Synonymous_form: event *is_organized_by* organizer
Property associations

`name`
`General_concept: text`
`Concept_type: role`
`ticket price`
`General_concept: number`
`Concept_type: role`
organizer *has* name
`Concept_type: property association`
event *has* ticket price
`Concept_type: property association`

Characteristics
event *is_finished* event venue *is_suitable_for_sport_events* concept 'sports arena' *incorporates* characteristic 'event venue *is_suitable_for_sport_events*'
Partitive verb concepts
auditorium *includes* stage `Concept_type: partitive verb concept` auditorium *includes* parterre `Concept_type: partitive verb concept`
Roles

`Company name`
`Concept type: role`
`General concept: text`

Individual concepts
`Jonas Grinius`

General_concept: person Events Ltd General_concept: company
Hierarchy of concepts
company General_concept: organizer company name General_concept: name event *takes_place_in* location General_concept: spectacle *is_performed_in* auditorium
Segmentations and categorization schemes
payment type Concept_type: categorization type Necessity: is_for general_concept event Events by payment Necessity: segmentation for general_concept event that subdivides event by payment type free event General_concept: event Necessity: is_included_in Events by payment paid event General_concept: event Necessity: is_included_in Events by payment
Facts
company Events Ltd *organizes* spectacle Tern
Definitions
paid event Definition: event that *has* ticket price greater_than 0 organizer Definition: person that *organizes* event and *sells_tickets_for* event

5.1.3 Evaluating the completeness of specifying business rules

In the second part, business rules of the domain were specified. Examples of business rules are presented in Table 5.2.

Table 5.2. Fragments of specification of business rules

Rules using representations with both placeholders not replaced
It is obligatory that organizer *sells_tickets_for* event if event *is* paid event. It is necessary that event *takes_place_in* location if event *takes_place_in* event venue which *is_located_in* location.
Rules using representations with placeholders replaced by names
It is obligatory that cultural events *are_organized_by* Events_Ltd. It is obligatory that Volleyball_match *takes_place_in* sports arena.
Rules using representations with the second placeholder replaced by quantity restriction

It is obligatory that organizer *cancels* event if number of sold tickets *of* event is_less_than 50. It is necessary that event *is_paid* if ticket price *of* event is_greater_than 0.
Rules using placeholders replaced by quantification representations
It is necessary that event *takes_place_in* exactly 1 location. It is necessary that organizer *is* company that *organizes* at_least 1 event. It is necessary that organizer *is_experienced* if organizer *organizes* at_least 20 events.

5.1.4 Evaluating the completeness of writing questions

In the third part, questions of the presented domain were specified. Examples of business rules are presented in Table 5.3.

Table 5.3. Examples of questions

Questions using representations with both placeholders not replaced
Find names *of* organizers that *organize* events. Find person that *organize* events that *are* free events.
Questions using representations with placeholders replaced by names
What organizer *organizes* spectacle Tern? Find events that *has_location* city Kaunas.
Questions using representations with the second placeholder replaced by quantity restriction
Find events whose number of sold tickets is_greater_than 100. What person *organize* events that *have* ticket price that is_greater_than 150?
Questions using placeholders replaced by quantification representations
Find person that *organizes* at_least 5 events. Find organizer that *cancelled* at_least 7 events.
Questions to find instances of certain type
Find persons. Find events.

5.1.5 Conclusions of evaluating the completeness of SBVR editor

The experiment showed that the created grammar of SBVR structured language is complete enough to specify all the required constructions of SBVR. It allows specifying all types of SBVR vocabulary concepts (i.e., noun concepts, verb concepts, and individuals), adding specifications for concepts, etc. The grammar allows specifying operative, structural and derivation rules, based on all types of representations of atomic formulations. All the required question types (presented in 2.1.5) are supported.

5.2 Evaluating the portability and multilingualism of SBVR structured language editor

This experiment was carried out to find out, if the editor can be used for configuring NLI to ontologies and allow its portability and multilingualism.

Evaluation of the portability of the SBVR SLE was performed writing SBVR specifications in three different domains: agents, events, and e-commerce.

To evaluate multilingualism, the editor was adjusted for the Lithuanian language by appending grammar with Lithuanian keywords and integrating a Lithuanian morphological library. Further, SBVR specifications of three domains were translated to the Lithuanian language.

The main problems of adjusting the editor to another language are related with establishing syntax based relations between compound concepts, as described in subsection 4.1.2. To avoid these problems, adjusting the cross-reference resolution mechanism for the Lithuanian language, morphological library, created in the SemantikaLT project [99] was used; as it contains a lemma finder and morphological generator functions. The lemma finder returns the lemma of word. Morphological generator returns the word in the specified morphological form. In SBVR SLE, the morphological generator is used to generate all morphological forms of word. For example, if the analysed word is a noun, generator will be used to find that noun in all cases and genders. A generated list is further passed to the Xtext cross-reference resolution mechanism. It accepts all possible forms at once and searches, to identify if at least one of them exist in the vocabulary (i.e. searches for vocabulary concept, specified in one of the forms). The drawback of such a solution is that many redundant forms must be generated.

To evaluate capabilities of establishing relations between concepts in the Lithuanian language, the number of compound individual concepts were specified. They were used to formulate facts. During the evaluation, how many of individuals in facts were successfully related with specified individual concept were counted. A short example of Lithuanian specification from domain of events with marked recognition errors is presented in Table 5.4.

Table 5.4. Example of specifying individuals and using them in facts

```
spektaklis
teatras
Priežastys_ir_pasekmės
 General_concept: spektaklis
Čipolino_nuotykiai
 General_concept: spektaklis
Snieguolė_ir_septyni_nykštukai
 General_concept: spektaklis
Kauno_Dramos_teatras
 General_concept: teatras
Kauno_Lėlių_teatras
 General_concept: teatras
Priežastis_ir_pasekmes vaidina Kauno_Dramos_teatre
Čipolino_nuotykius vaidina Kauno_Lėlių_teatre
Snieguolę_ir_septynis_nykštukus vaidina Kauno_Lėlių_teatre
```

Two different techniques were used to recognize concepts and establish relations: lemma finder alone and lemma finder with morphological generator. Recognition errors occurred if either lemmas of words were not found (i.e., morphological vocabulary errors), or lemmas or generated words were in unsuitable morphological forms (i.e., morphological errors).

Results of the experiment are presented in Table 5.5. They are calculated using precision PR_i, (i.e., the ratio of correctly recognized concepts), recall RR_i (i.e., the ratio of recognized concepts), and F-measure FR_i metrics. These metrics were calculated using parameter AT_i - the total number of individuals to recognize (i.e., individuals used in facts) and RT_i ï counted number of recognized individuals:

$$PR_i = \frac{RT_i \cap AT_i}{AT_i} \qquad RR_i = \frac{RT_i \cap AT_i}{RT_i} \qquad FR_i = 2 \times \frac{PR_i \times RR_i}{PR_i + RR_i}$$

These metrics were calculated for three cases:

1) Using lemma finder only (i=1);
2) Using lemma finder with morphological generator (i=2);
3) Using lemma finder with morphological generator, but excluding terms with lemmas not found. This shows the pure impact made by morphological generator (i=3).

The impact of morphological generator on the quality of recognizing terms in the Lithuanian language was evaluated by the increase of recall ΔRR_i and F measure ΔFR_i:

$$\Delta RR_i = RR_i - RR_1, \Delta FR_i = FR_i + FR_1, i = 2,3$$

Table 5.5. Evaluation of quality to recognize compound terms in Lithuanian language

Business domain	**Terms to recognize**	**Recognized terms**	***PR*** PR_i	RR_i	FR_i	ΔRR_i	ΔFR_i
Lemma finder							
Agents	178	131	1	0,735	0,847	ï	ï
Events	117	84	1	0,718	0,836	ï	ï
E-commerce	509	59	1	0,116	0,208	ï	ï
Lemma finder with morphological generator							
Agents	178	154	1	0,865	0,928	0,129	0,081
Events	117	114	1	0,974	0,987	0,256	0,151
E-commerce	509	458	1	0,900	0,947	0,784	0,739
Lemma finder with morphological generator excluding vocabulary errors (pure impact made by the morphological generator)							
Agents	155	154	1	0,994	0,997	0,258	0,150
Events	115	114	1	0,991	0,995	0,273	0,159
E-commerce	477	458	1	0,960	0,980	0,844	0,772

Conclusions. The results show, that the recall of recognizing concepts using only lemma finder is not good at all (0,116 ï 0,735). The worst results are in the e-commerce domain. Vocabulary of this domain has many titles of product categories, defined in plural form. As lemmas are in singular form, recognition errors occur.

The main errors in domain of agents were caused by unrecognized names or surnames (i.e., vocabulary errors) and disappeared pronominal forms in lemmas. In the events domain, errors were similar ï they were caused by unrecognized titles of spectacles, disappeared pronominal forms or the wrong form of genders in lemmas.

However, the morphological generator greatly improved the concepts recognition results (0,865 ï 0,974). Certainly, it is better when vocabulary errors were not counted (0,960 ï 0,994). Therefore, the editor can be adjusted to write SBVR specifications in different languages. For complete elimination of term recognition errors, the morphology support in the SBVR SLE can be improved. To eliminate vocabulary errors, morphological vocabulary can be improved to be able for finding lemmas even of rare words.

The experiment also showed that SBVR SLE allows portability of NLI, because it is capable of writing specifications in different domains.

5.3 Evaluating the effectiveness, multilingualism, and portability of NLI to ontologies

The goal of this experiment is to evaluate the created NLI to ontologies and find out, if SBVR can be used as a basis of NLI for questioning ontologies in multiple languages and if the solution is portable. Effectiveness was evaluated calculating correctness of answering English and Lithuanian questions. Portability was evaluated configuring NLI for different domains.

The prototype, described in subsection 4.2 was used to perform the experiment. However, it additionally uses clarification dialog to disambiguate questions. The used data sets are based on the Mooney Natural Language Learning Data created by Ray Mooney and his group from the University of Texas at Austin [75]. The original knowledge bases were created using Prolog and have been used to evaluate NLIs to databases. They were translated to OWL knowledge base and published by the Dynamic & Distributed information Systems Group from the University of Zurich [87]. This knowledge base is now often used to evaluate NLIs to ontologies. It contains three test knowledge bases (i.e., geography, restaurants, and jobs) with sets of question. The correctness is expressed by precision and recall parameters.

In this experiment, two knowledge bases were used: geography and restaurants. The first one stores geographical information about the United States of America: states, cities of states, capitals of states, borders of states, population of cities and states, rivers, highest points, etc. It contains a set of 880 questions. However, the subset of 250 questions is often used, because it semantically represents the whole set. The conceptual model of the geography knowledge base is presented as a class diagram in Figure 5.2.

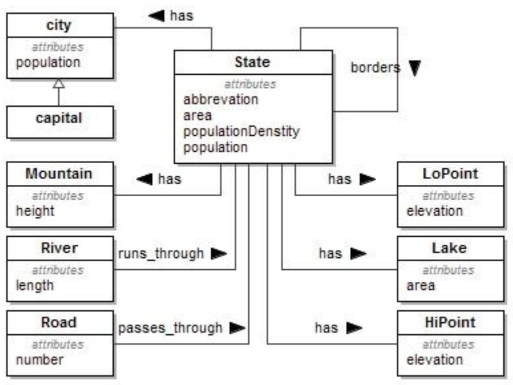


Figure 5.2. Conceptual model of geography knowledge base

The restaurant knowledge base contains information about restaurants, their ratings, locations, type of food, etc. It has 251 representative questions. The conceptual model of the restaurant knowledge base is presented in Figure 5.3.

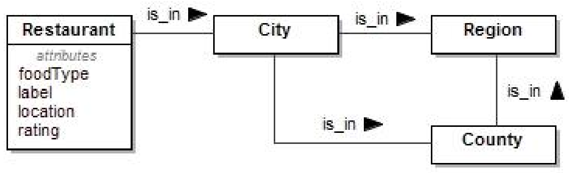


Figure 5.3. Conceptual model of restaurant knowledge base

Mooney knowledge bases contain English questions to evaluate NLIs. They were translated to the Lithuanian language to evaluate the solution, not only in English but also in the Lithuanian language. Translations of questions of the geography knowledge base are presented in appendix 3.

In the created solution, questioning is carried out using SBVR business vocabulary and business rules specifications that correspond the ontology. These specifications were created during configuration with SBVR SLE in the English and Lithuanian languages. Definitions for concepts that are derived from their properties were also specified (e.g., *large_city*, *italian_restaurant*, etc.). Fragments of English and Lithuanian SBVR specifications are presented in Table 5.6.

Table 5.6. Fragments of SBVR specifications

Geography knowledge base in English
city
population
General_concept: number
Concept_type: role
city *has* population
Concept_type: property association
It is necessary that major city *is* city that *has* population greater_than 300000.
Geography knowledge base in Lithuanian
miestas
populiacija
General_concept: number
Concept_type: role
miestas *turi* populiaciją
Concept_type: property association
Būtina, kad didelis mietas *yra* miestas kuris *turi* populiaciją didesnę_už 300000.
Restaurant knowledge base in English
restaurant
rating
General_concept: text
Concept_type: role
restaurant *has* rating

```
   Concept_type: property association
restaurant has food type
   Concept_type: property association
```

```
It is necessary that good french restaurant is restaurant that
has rating "good" and has food type "french"
```

Restaurant knowledge base in Lithuanian

```
restoranas
reitingas
   General_concept: number
   Concept_type: role
restoranas turi reitingą
   Concept_type: property association
restoranas gamina patiekalų rūšį
```

```
Būtina, kad geras prancūziškas restoranas yra restoranas kuris
turi reitingą "geras" ir gamina patiekalų rūšį "prancūziškas".
```

After creating SBVR specifications, OWL ontologies were prepared by adding labels with SBVR representations for ontology resources in order to establish the compliance between ontology resources and SBVR concepts using principles defined in [51].

During the experiment, English and Lithuanian questions were transformed to SPARQL queries using the created transformations. An example of a English question and transformed query is presented in Table 5.7.

Table 5.7. Example question and transformed query

```
What is population of Dallas?
```

```
SELECT
 ?population_i
WHERE {
 ?city_i ?city_has_population ?population_i.
 ?city_has_population rdfs:label "city has population"@en .
 ?city_i rdf:type ?city_c.
 ?city_c rdfs:label "city"@en.
 ?population_i rdf:type ?population_c.
 ?population_c rdfs:label "population"@en
 FILTER regex( ?city_i, "Dallas")
}
```

Queries were executed against OWL ontology and parameters of precision, recall, and F-measure were calculated. These parameters are adapted from the information retrieval area. The precision *PQ* is the number of questions for which the correct answer is returned (*CQ*) divided by number of questions whose answers were returned at all (*AQ*). The recall *RQ* is the number of questions for whose correct answers were returned (*CQ*) divided by the total number of questions (*TQ*) that can be answered by the knowledge base [113]. Formulas of calculating precision, recall, and *F*-measure are presented below:

$$PQ = \frac{CQ}{AQ} \qquad RQ = \frac{CQ}{TQ} \qquad FQ = 2 \times \frac{PQ \times RQ}{PQ + RQ}$$

Results of evaluating the correctness are presented in Table 5.8.

Table 5.8. Results of evaluating correctness

Knowledge base	***TQ***	***AQ***	***CQ***	***PQ***	***RQ***	***FQ***
Geography	250	English questions				
		224	205	0,9151	0,82	0,8649
		Lithuanian questions				
		232	222	0,9569	0,888	0,9212
Restaurants	251	English questions				
		247	188	0,7611	0,749	0,7550
		Lithuanian questions				
		248	187	0,754	0,745	0,7495

Conclusions. In the geography knowledge base, the created prototype was not able to answer questions with negations. Due to the incompleteness of natural language analysis algorithms, some English questions could not be answered. For example, questions of grammatical structure that differs from the structure of SBVR concepts (e.g., *Through which states does the Mississippi run?*) or questions to find minimum or maximum values according to the specified criterion (e.g., *What is the smallest state by area?*). Question transformation rules could not transform the questions with double comparisons of minimum or maximum values (e.g., *Which states have points higher than the highest point in Colorado?*).

Results of the restaurant knowledge base are worse, because prototype could not identify the type of answer correctly for many questions. For example, the question *Where is Chinese food in Bay area?* was answered incorrectly by showing a list of Chinese restaurants instead of their exact locations. It negatively affected precision.

Clarification dialog made a significant impact on improving precision. It helped to answer questions by disambiguating proper names of places. For example, the question *What is the population of Seattle Washington?* contains the composite proper name *Seattle Washington*, meaning city *Seattle* in state *Washington.*

The experiment showed that NLI is portable and can be configured to question ontologies of different domains. However, it does not guarantee, that the correctness will always be good. Each domain can have specific questions that could not be easily answered. The correctness depends on the question analysis rules, which can be improved to achieve better results.

The solution also allows questioning ontologies in different languages. It is ensured by SBVR SLE, which allows writing multilingual SBVR specifications and transformation rules. These rules transform SBVR questions to SPARQL queries using a model of the questionĝs meaning that is independent from language. Therefore, correctness of answering English and Lithuanian questions is similar.

The solution was compared with other NLIs to ontologies that were analysed in subsection 2.1.10 and evaluated using the same knowledge bases. Results of the comparison are presented in Table 5.9.

Table 5.9. Comparison of this solution to other NLIs to ontologies

	Geography			**Restaurants**		
	Precision	***Recall***	***F-measure***	***Precision***	***Recall***	***F-measure***
FREyA [19]	0,924	0,924	0,924	ï	ï	ï
PANTO [129]	0,8805	0,8586	0,8694	0,9087	0,9664	0,9367
Querix [56]	0,8608	0,8711	0,8659	ï	ï	ï
This solution	0,9151	0,82	0,8649	0,7611	0,749	0,7550

The correctness of the solution is similar to PANTO and Querix systems. FREyA is the most sophisticated of the compared systems. It has a lot of means to analyze the semantics of question, such as communicating with users writing an ambiguous question, helping to formulate correct questions, using ontology knowledge to interpret questions. Therefore, this system has the highest correctness.

In the experiment, how easily the solution can be adjusted for different domains was not evaluated, because it is very difficult to estimate and compare. The effort that is needed to configure NLIs has been rarely quantified and compared across paradigms [13].

5.4 Evaluating the capabilities of NLI to ontologies to map questions with combinations of ontology resources

The goal of this experiment is to evaluate questioning capabilities when the structure of the ontology differs from language formulations used for writing questions. The evaluation was performed using the n-ary relation case, which often occurs in Semantic Web applications. The representative example of n-ary relation was adapted from [82] and presented in Figure 5.4. It contains the relation class *purchase*, which is connected with classes *buyer*, *seller*, and *products* that are being purchased.

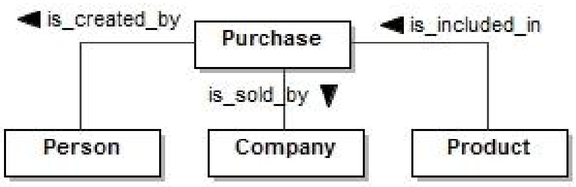


Figure 5.4. N-ary relation of purchases domain

Relations of this ontology do not express very useful information for users. For example, it is unlikely that he or she will be interested what products were included in certain purchase. Probably, users will be interested in relations that are not explicitly declared in the ontology, but can be derived. For example, *what products were bought by some person?* The solution allows describing derivations in SBVR specification and formulating questions using derived concepts. SBVR vocabulary and derivation rules of the example are presented in Table 5.10.

Table 5.10. SBVR specification for describing n-ary relations of purchases domain

```
purchase
person
product
purchase is_created_by person
product is_included_in purchase
product is_bought_by person
--------------------------------------------------------------------
It is necessary that product is_bought_by person if product
is_included_in purchase that is_created_by person.
```

The example question and transformed SPARQL query is presented in Table 5.11.

Table 5.11. SBVR specification for describing n-ary relations of purchases domain

```
What products were bought by John Smith?
--------------------------------------------------------------------
SELECT
 ?product_i
WHERE {
 ?product_i ?is_included_in ?purchase_i.
 ?is_included_in rdfs:label "product is_included_in
purchase"@en.
 ?purchase_i ?is_created_by ?person_i.
 ?is_created_by rdfs:label "purchase is_created_by person"@en.
 ?product_i rdf:type ?product_c.
 ?product_c rdfs:label "product"@en.
 ?purchase_i rdf:type ?purchase_c.
 ?purchase_c rdfs:label "purchase"@en.
 ?person_i rdf:type ?person_c.
 ?person_c rdfs:label "person"@en
 FILTER regex(?person_i, "John Smith")
}
```

Conclusions. The experiment showed, that the solutions allow questioning, when it is needed to map questions with combinations of ontology resources. Mapping of a question to ontology can be implemented using SBVR derivation rules or formal definitions.

5.5 Threats to validity and answers to research questions

The confidence of experiments may be affected by internal and external threats to validity. In this work, threats to internal validity can be caused by chosen experimental domains or by chosen external tools needed for implementing the prototype of the solution (i.e., morphological library and SBVR to OWL 2 transformation component), which can make a positive or negative influence to experiment results.

To avoid the influence of chosen domains, the following means were used:

1) The representative example, used in the experiment of specification capabilities of SBVR SLE, contains all SBVR constructs to represent;
2) Lithuanian language concepts of 3 different domains were analysed in the experiment for evaluating the portability and multilingualism of SBVR SLE;

3) In the experiment of NLI to ontologies, standard test data sets were used that are now often used to evaluate NLIs to ontologies [75], [87]. Therefore, the comparison of the solution with other existing solutions can be accepted as a trustworthy one.

The external tools could not make an unssen influence to the experimental results because:

1) Results of the experiment of portability and multilingualism of SBVR SLE depend on quality of the morphological component. More precisely, the set of recognizable words and capability to generate required morphological forms. The functionality of the morphological library of the Lithuanian language was insufficient;
2) Results of the experiment of NLI to ontologies depend on the quality of the configuration of the NLI. If SBVR specification does not fully meet the ontology (e.g., some ontology concepts do not have corresponding SBVR concepts, lack of derivations, etc.), results of correctness can be worse. To minimise this risk, SBVR to OWL 2 transformation ([51], [52], and [53]) component can be used to synchronise ontologies and SBVR specifications automatically.
3) Threats to external validity raise a question whether the research results are applicable to other languages. During experiments of multilingualism of SBVR SLE and NLI to ontologies, the English and Lithuanian languages were considered. It seems enough, because the Lithuanian language is complex and morphologically rich. This suggests that the solution can be adjusted to other grammatically similar languages.

In this work, the following answers to research questions were concluded.

Is it possible to use SBVR questions for querying ontologies and relating natural language questions with combinations of ontology resources? SBVR questions can be used for querying ontologies. Questions can be written and interpreted using SBVR specifications as lexicon. SBVR standard allows representing meaning of a question. This standard can be used as an intermediate knowledge representation model and transformed to SPARQL queries. To relate formulation of questions (i.e., the way, how users think and question data) and ontology structure, SBVR derivation rules or formal definitions can be used.

How natural language questions can be transformed to SPARQL using SBVR? Natural language questions can be transformed to SPARQL following these common steps: (1) performing analysis of natural language question and identifying SBVR concepts that questions are based on; (2) creating SBVR model of meaning of question; (3) performing model-based transformations of SBVR to SPARQL; (4) generating textual queries from SPARQL models.

Is it possible to achieve portability without compromising the correctness of NLI to ontologies using SBVR? The portability requires a domain independent tool for writing SBVR specifications. The mutual correspondence between vocabulary concepts and ontology resources must be ensured. The created solution allows achieving portability of a natural language interface to ontologies of different domains by:

- Using domain independent SBVR SLE;
- Using domain independent transformations of meaning of SBVR questions to SPARQL queries;
- Synchronizing SBVR and OWL 2 concepts manually or via mutual SBVR ï OWL 2 transformations, and, in particularly, using labels for ontology resources to relate them with SBVR concepts.

Greater efforts put into configuring NLI (e.g., specifying all available derivations) will cause better correctness. However, to ensure sufficient correctness in certain domains, improvement of question analysis algorithms may be necessary, especially when facing specific formulations of questions that were not analysed before.

Can SBVR based NLI to ontologies be adjusted to different languages and what components are language specific? SBVR based NLI to ontologies can be adjusted for investigated natural languages. For interpreting questions in a certain language, question analysis algorithms and morphological library must be adjusted.

6 CONCLUSIONS

1. Analysis of scientific publications has shown that the preferable interface for querying ontologies is a natural language interface (NLI). The most important requirements of such a NLI are adjustability to different languages, ability to deal with complex structures of ontology resources, portability and habitability. Existing NLIs to ontologies only allow questioning in English and usually in the form when the formulation of question directly corresponds to the structure of ontology. On the other hand, existing solutions are portable and can be configured for questioning in different domains. To improve habitability, NLIs use various techniques that can be divided into two groups: (1) methods intended to familiarize users with lexicon and help formulating questions; (2) methods intended to help interpreting and disambiguating questions.
2. The analysis of SBVR standard has shown that a distinguishing feature of its metamodel to separate the meaning from representation allows achieving multilingualism. It suggests that questions in the SBVR based NLI could be transformed to semantic queries using language-independent rules, because the model of a questionôs meaning is the same for all languages. The language dependent components of such a NLI should only be those that help writing and interpreting questions. In order to achieve multilingualism, these components should be replaced or adjusted for questioning in a certain language. The architecture of the implemented NLI was designed in pursuance of these ideas. Another important aspect related with deciding to use SBVR for NLI to ontologies is the derivation rules that can be used to relate simple questions with complex ontology structure.
3. The analysis of SPARQL revealed that the syntax of this query language and metamodel of SBVR questions has conforming elements, expressing information needs and restrictions of a query. It led to the assumption that

transformation of SBVR questions to SPARQL is feasible. Therefore, it was decided to describe detailed mappings between the metamodel of SBVR questions and SPARQL and create transformation rules.

4. An important prerequisite to use SBVR for NLI to ontologies is a robust SBVR editing tool for specifying vocabulary and rules (i.e., configuration of NLI) and generating XMI models of questions for further transformations. The analysis of existing tools showed that none of them meets the requirements and their further improvements are complicated. Therefore, it was decided to create a new SBVR editor.
5. The conception of SBVR editor is based on structured language grammar (in EBNF-like form) for specifying business vocabularies, rules, and writing questions. The grammar was described analysing the metamodel of SBVR representations, structured language examples from SBVR specification and practice. Grammar supports questions to retrieve; objects of a certain type, questions with modifier attachments, cardinality restrictions, numeric comparisons, and count function.
6. Xtext framework was used to describe the grammar and implement the SBVR editor. Experiments have shown that the editor allows specifying all the required SBVR constructions. It also creates the necessary preconditions for portability and a multilingual NLI to ontologies, along with being suitable for the configuration task.
7. The conception of SBVR based NLI to ontologies contains the following components (only the first two are language-dependent):
   - User interface, which allows formulating natural language questions, conforming to SBVR vocabulary;
   - Question analyser, which identifies SBVR concept (s) that the question is based on;
   - SBVR model composer, which constructs the questionŏs SBVR XMI model;
   - Component of query transformation, which transforms the questionŏs SBVR XMI model to SPARQL query.
8. Implementation of SBVR based NLI proved that it is possible to use SBVR questions for querying ontologies. Additionally, experimental investigation shows that the solution allows querying ontologies, whose structure (i.e., expressing relevant part of domain knowledge) directly does not correspond to the structure of the natural language questions. This is achieved specifying derivation rules of SBVR concepts that are used when transforming questions to SPARQL.
9. Experiments of questioning in two different domains proved that it is possible to achieve the portability and multilingualism of a NLIthat uses SBVR standard. Portability is achieved by allowing SBVR specification to be written fora certain domain and linking it with the ontology. The evaluation of effectiveness showed similar result as other NLIs (i.e. f-measure is 0,86 in the domain of geography and 0,75 in the domain of restaurants). However, the

main advantages of the created solution is multilingualism and the ability to question ontologies, whose structure does not directly correspond to the structure of natural language questions.

7 REFERENCES

[1] **Androutsopoulos, I., Ritchie, G. D., Thanisch, P.** Natural Language Interfaces to Databases ï An Introduction. *Natural Language Engineering*, *1*(1), 1995, Pages 29-81.

[2] **ANother Tool for Language Recognition (ANTLR).** Available from: http://www.antlr.org/ [Accessed: 21 Jun 2011].

[3] **Benjamin, P. C., Menzel, C. P., Mayer, R. J., Fillion, F., Futrell, M. T., deWitte, P. S., Lingineni, M.** *IDEF5 Method Report.* Knowledge Based Systems, Inc., September 21, 1994.

[4] **Berners-Lee, T., Hendler, J., Lassila, O.** The Semantic Web, *Scientific American*, May 2001, p.28ï37.

[5] **Bernotaitytă, G., Nemuraită, L., Butkienă, R., Paradauskas, B.** Developing SBVR vocabularies and business rules from OWL2 ontologies. In *Proceedings of 19th International Conference on Information and Software Technologies, ICIST 2013*, Kaunas, Lithuania, October 10-11, 2013, p.134-145.

[6] **Bernstein, A., Kaufmann, E., Kaiser, C.** Querying the semantic web with ginseng: A guided input natural language search engine. In: *Proceedings of 15th Workshop on Information Technologies and Systems*, Las Vegas, Nevada, USA, 2005, p.112ï126.

[7] **Bodenstaff, L., Ceravolo, P., Ernesto Damiani, R., Fugazza, C., Reed, K., Wombacher, A.** Representing and Validating Digital Business Processes. *Advances in Web Semantics*, January, 2007, p.219-246.

[8] **Borst, W.** *Construction of Engineering Ontologies.* PhD thesis, Institute for Telematica and Information Technology, University of Twente, Enschede, The Netherlands, 1997.

[9] **Brook, J.** SUS ï A quick and dirty usability scale. P. Jordan, B. Thomas, B. Weerdmeester, A. McClelland (eds.) *Usability Evaluation in Industry*, 1996, p.189-194.

[10] **Cabot, J., Pau, R., Raventos, R.** From UML/OCL to SBVR specifications: A challenging transformation. *Information Systems, 35*(4), 2010, p.417-440.

[11] **Chapin, D.** Implementing SBVR with a Practitionerǿs Perspective. In *Proceedings of the 2010 international conference on Semantic web rules (RuleML'10)*, Washington, DC, USA, October 21-23, p.16-19.

[12] **Cimiano, P., Haase, P., Heizmann, J., Mantel, M.** ORAKEL: *A Portable Natural Language Interface to Knowledge Bases.* Technical report, Institute AIFB, University of Karlsruhe (2007).

[13] **Cimiano, P., Minock, M.** Natural Language Interfaces: What is the Problem? - A data-driven quantitative analysis. In *Proceedings of the 14th international conference on Applications of Natural Language to Information Systems (NLDB'09)*, Saarbrucken, Germany, June 24-26, 2009, p.192-206.

[14] **Cohen, W. W., Ravikumar, P., Fienberg, S.E.** A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of the 18th International Joint*

Conference on Artificial Intelligence, Accapulco, Mexico, August 9-15, 2003, p.73-78.

[15] **Collibra.** *Business Semantics Glossary*. Available from: https://www.collibra.com/ [Accessed: 25 Mar 2012].

[16] **Ļeponienā, L., Nemuraitā, L., Vedrickas, G.** Semantic business rules in service oriented development of information systems. In *Proceedings of the 15th International Conference on Information and Software Technologies (IT 2009)*, Kaunas, Lithuania, April 23-24, 2009, p.404ï 416.

[17] **Damljanovic, D.** *Natural Language Interfaces to Conceptual Models*. Ph.D. thesis, The University of Sheffield, Language Resources and Evaluation. http://etheses.whiterose.ac.uk/1630/ (2011).

[18] **Damljanovic, D.** Towards Portable Controlled Natural Languages for Querying Ontologies. In *Proceedings of Second Workshop on Controlled Natural Language*, Marettimo Island, Italy, September 13-15, 2010.

[19] **Damljanovic, D., Agatonovic, M., Cunningham, H.** Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-based Lookup through the User Interaction. In *Proceedings of the 7th international conference on The Semantic Web: research and Applications*, Heraklion, Crete, Greece, May 30 - June 3, 2010, p.106-120.

[20] **Damljanovic, D., Agatonovic, M., Cunningham, H., Bontcheva, K.** Improving Habitability of Natural Language Interfaces for Querying Ontologies with Feedback and Clarification Dialogues. *Web Semantics: Science, Services and Agents on the World Wide Web*. Volume 19, March 2013, 1ï 21.

[21] **Damljanovic, D., Bontcheva, K.** Towards Enhanced Usability of Natural Language Interfaces to Knowledge Bases. *Web 2.0 & Semantic Web*, 2009, p.105-133.

[22] **Damljanovic, D., Tablan, V., Bontcheva, K.** A Text-based Query Interface to OWL Ontologies. In *Proceedings of 6th Language Resources and Evaluation Conference (LREC)*, Marrakech, Marocco, May 28-30, 2008, p.205-212.

[23] **De Tommasi, M., Corallo, A.** SBEAVER: A Tool for Modeling Business Vocabularies and Business Rules. In *Proceedings of 10th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KESó06)*, Bournemouth, UK, October 9-11, 2006, p.1083-1091.

[24] **Demuth, B., Liebau, H. B.** An Approach for Bridging the Gap Between Business Rules and the Semantic Web. In *Proceedings of the 2007 International Conference on Advances in Rule Interchange and Applications*, Orlando, Florida, USA, October 25-26, 2009, p.119ï 133.

[25] **Eclipse Modeling Framework (EMF).** Available from: http://eclipse.org/modeling/emf [Accessed: 10 Oct 2012].

[26] **Epstein, S. S.** Transportable Natural Language Processing through Simplicity - the PRE System. *ACM Transactions on Information Systems (TOIS)*, *3*(2), April 1985, p.107-120.

[27] **Eysholdt, M., Behrens, H.** Xtext: implement your language faster than the quick and dirty way. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion (OOPSLAó2010)*, Reno/Tahoe, Nevada, USA, October 17-21, p.307–309.

[28] **Fellbaum, C.** *WordNet - An Electronic Lexical Database*. Cambridge, MA: MIT Press, 1998.

[29] **Ford, B.** Parsing Expression Grammars: A Recognition-Based Syntactic Foundation. In *Proceedings of the 31st ACM SIGPLAN ï SIGACT symposium on Principles of programming languages*, Venice, Italy, January 14-16, 2004, p.111-122.

[30] **Garcia, V. L., Motta, E., Uren, V.** AquaLog: An ontology-driven Question Answering System to interface the Semantic Web. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, June 4-9, 2006, New York, USA, p.269-272.

[31] **Goedertier, S., Vanthienen, J.** A Vocabulary and Execution Model for Declarative Service Orchestration. In *Proceedings of the 2007 International Conference on Business Process Management*, Brisbane, Australia, 2007, p.496ï 501.

[32] **Gronback, R. C.** *Eclipse Modeling Project. A Domain-Specific Language Toolkit.* Addison-Wesley Professional, 2009, p.1–675.

[33] **Grosz, B.** TEAM: A transportable natural language interface system. In *Proceedings of the Conference on Applied Natural Language Processing*, February 1-3, Santa Monica, California, 1983, 39-45.

[34] **Gruber, T.** A translation approach to portable ontologies. *Knowledge Acquisition*, *5*(2), 1993, p.199-220.

[35] **Gruber T.** Ontology. *Encyclopedia of Database Systems*, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag US, 2009.

[36] **Guarino, N.** Formal Ontology in Information Systems. In *Proceedings of the first International Conference Formal Ontology in Information Systems (FOISô98)*, Trento, Italy, 6-8 June 1998. Amsterdam, IOS Press, p.3-15.

[37] **Guarino, N., Oberle, D., Staab, S.** What Is an Ontology. *Handbook on Ontologies*. Springer, 2nd edition, 2009, p.1-17.

[38] **Guarino, N.** Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. In *Proceedings of International Summer School on Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology (SCIE-97)*, Springer-Verlag, 1997, p.139-170.

[39] **Gudas, S., Brundzaitå, R.** Knowledge-based enterprise modelling framework. In *Proceedings of 4th International Conference, ADVIS 2006*, Izmir, Turkey, October 18-20, 2006, p.334-343.

[40] **Gudas, S., Lopata, A.** Knowledge-based refinement of business management functions. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD 2009)*, Madeira, Portugal, October 6-8, 2009, p.435-442.

[41] **Gudas, S., Lopata, A.** Workflow models based acquisition of enterprise knowledge. *Information Technology and Control*, *36*(1A), 2007, p.103-109.

[42] **Haase, P., Broekstra, J., Eberhart, A., Volz, R.** A Comparison of RDF Query Languages. In *Proceedings of Third International Semantic Web Conference*, Hiroshima, Japan, November 7-11, 2004, p.502-517.

[43] **Harris, A., Seaborne, A.** *SPARQL 1.1 Query Language.* W3C Recommendation 21 March 2013. Available from: http://www.w3.org/TR/sparql11-query/ [Accessed 06 May 2014].

[44] **Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D., Slocum, J.** Developing a Natural Language to Complex Data. *ACM Transactions on Database Systems*, 3(2), 1978, p.105-147.

[45] **Hevner, A. R., March, S. T., Park, J., Ram, S.** Design Science in Information Systems Research. *MIS Quarterly, 28* (1), 2004, p.75-105.

[46] **Huber, S., Carrez, C., Suttner, H.** Development of Innovative Services Enhancing Interoperability in Cross-Organizational Business Processes. In *Proceedings of Third International IFIP Working Conference (IWEI 2011)*, Stockholm, Sweden, March 23-24, 2011, p.75-88.

[47] **Hyvonen, E., Makela, E.** Semantic Autocompletion. In *Proceedings of the first Asia Semantic Web Conference (ASWC 2006)*, Beijing, China, September 3-7, 2006, p.4-9.

[48] **Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.** ATL: A model transformation tool. *Science of Computer Programming, 72*(1-2), 2008, p.31-39.

[49] **Jurkeviļius, D., Vasilecas, O.** Ontology Creation by Using a Formal Concepts Approach. In *Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing (CompSysTechô10)*, Sofia, Bulgaria, June 17-18, 2010, Vol. 471, p.64ï 70.

[50] **Kamada, A., Mendes, M.** Business Rules in a Service Development and Execution Environment. In *Proceedings of Eleventh International IEEE EDOC Conference Workshop (EDOCW'07)*, October 17-19, 2007, p.1366ï 1371.

[51] **Karpoviļ, J., Kriģiĩnienā, G., Ablonskis, L., Nemuraitā, L.** The comprehensive mapping of semantics of business vocabulary and business rules (SBVR) to OWL 2 ontologies. *Information Technology and Control, 43*(3), 2014, p.289-302.

[52] **Karpoviļ, J, Nemuraitā, L.** Transforming SBVR business semantics into Web ontology language OWL2: main concepts. In *Proceedings of 17th International Conference on Information and Software Technologies (IT 2011)*, Kaunas, Lithuania, April 27ï 29, 2011, p.231ï 238.

[53] **Karpoviļ, J., Nemuraitā, L., Stankeviļienā, M.** Requirements for semantic business vocabularies and rules for transforming them into consistent OWL2 ontologies. In *Proceedings of 18th International Conference on Information and Software Technologies (ICIST 2012)*, Kaunas, Lithuania, September 13-14, 2012, p.420-435.

[54] **Kaufmann, E., Bernstein, A.** Evaluating the Usability of Natural Language Query Languages and Interfaces to Semantic Web Knowledge Bases. *Web Semantics: Science, Services and Agents on the World Wide Web, 8*(4), 2010, p.377–393.

[55] **Kaufmann, E., Bernstein, A., Fischer, L.** NLP-Reduce: A naive but domain independent natural language interface for querying ontologies. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, Busan, Korea, November 11-15, 2007, p.281-294.

[56] **Kaufmann, E., Bernstein, A., Zumstein, R.** Querix: A Natural Language Interface to Query Ontologies Based on Clarification Dialogs. In *Proceedings of 5th International The Semantic Web Conference (ISWC 2006)*, Athens, Georgia, USA, November 5-9, 2006, p.980ï 981.

[57] **Kjernsmo, K., Passant, A.** *SPARQL New Features and Rationale.* Available from: http://www.w3.org/TR/2009/WD-sparql-features-20090702/ [Accessed 06 July 2012].

[58] **Klein, D., Manning, C.D.** Accurate Unlexicalized Parsing. In *Proceedings of the 41th Annual Meeting on Association for Computational Linguistics*, Sapporo, Japan, July 7-12, 2003, Vol. 1, p.423-430.

[59] **Kriechhammer, M.** *Querying Systems for business models.* Available from: http://www.kriechhammer.com /?English_Portfolio:my_Documents:Finals, 2006 [Accessed: 25 June 2010].

[60] **Krifka, M.** Quantifiers in Questions. *Korean Journal of English Language and Linguistics 3*, 2003, p.499-526.

[61] **Krigľiĩnienă, G., Nemuraită, L., Butkienă, R., Paradauskas, B.** Rules for transforming OWL 2 ontology into SBVR. In *Proceedings of the 6th international conference on knowledge engineering and ontology development (KEOD 2014)*, Rome, Italy, October 21-24, 2014, p.256-263.

[62] **Lemmens, I., Nijssen, M., Nijssen, S.** A NIAM2007 Conceptual Analysis of the ISO and OMG MOF Four Layer Metadata Architectures. In *Proceedings of On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*, Vilamoura, Portugal, November 25-30, 2007, p.613–623.

[63] **Linehan, M. H.** Ontologies and Rules in Business Models. In *Proceedings of 11th International IEEE EDOC Conference Workshop*, Annapolis, Maryland, USA, October 15-19, 2007, p.149ï 156.

[64] **Linehan, M. H.** SBVR Use Cases. In *Proceedings of 2008 International Symphosium on Rule Representation, Interchange and Reasoning on the Web (RuleMLâ08)*, Orlando, Florida, USA, October 30-31, 2008, p.182–196.

[65] **Linehan, M. H.** Semantics in Model-Driven Business Design. In *Proceedings of 2nd International Semantic Web Policy Workshop (SWPW'06)*, Athens, Georgia, November 5-9, 2006, p.86-93.

[66] **Lopata, A., AmbraziĨnas, M., Gudas, S.** Knowledge-based MDA requirements specification and validation technique. *Transformations in Business & Economics*, 2012, 11(1(25)), p.248-260.

[67] **Lopata, A., AmbraziĨnas, M., Gudas, S., Butleris, R.** The main principles of knowledge-based information systems engineering. *Electronics and Electrical Engineering*, 2012, 4(120) p.99-102.

[68] **Lopata, A., AmbraziĨnas, M., Veitaită, I., Masteika, S., Butleris, R.** SysML and UML models usage in knowledge based MDA process. *Electronics and electrical engineering.* 2015, 21(2), p.50-57.

[69] **Marinos, A., Gazzard, P., Krause, P.** An SBVR Editor with Highlighting and Auto-completion. In Proceedings of 5th International Rule Challenge, Vol. 799, 2011, p.111–118.

[70] **Marinos, A., Krause, P.** An SBVR Framework for RESTful Web Applications. In *Proceedings of the 2009 International Symposium on Rule Interchange and Applications*, Las Vegas, Nevada, USA, November 5-7, 2009, p.144ï 158.

[71] **Mill , J. S.** *A System of Logic*. University Press of the Pacific, Honolulu, 2002.

[72] **Minock, M.** A STEP Towards Realizing Coddâs Vision of Rendezvous with the Casual User. In *Proceedings of 33rd International Conference on Very Large Data Bases*, Vienna, Austria, September 23-27, 2007, p.1358-1361.

[73] **Minock, M., Olofsson, P., N¨aslund, A.** Towards Building Robust Natural Language Interfaces to Databases. In *Proceedings of the 13th international conference on*

Natural Language and Information Systems: Applications of Natural Language to Information Systems, London, UK, June 24-27, 2008, p.187 - 198.

[74] **Nadeem, T.** Automated Translation of SBVR to SQL Queries. *International Journal of Emerging Sciences, 4*(1), 2014.

[75] **Natural Language Learning Data.** Available from: http://www.cs.utexas.edu/users/ml/nldata.html [Accessed: 05 Apr 2015].

[76] **Nemuraite, L., Skersys, T., Ġukys, A., Ġinkeviļius, E., Ablonskis, L.** VETIS tool for editing and transforming SBVR business vocabularies and business rules into UML&OCL models. In *Proceedings of the 16th International Conference on Information and Software Technologies (IT 2010)*, Kaunas, Lithuania, April 21-23, 2010, p.377–384.

[77] **Nenortaitā, J., Butleris, R.** Improving Business Rules Management through the Aplication of Adaptive Business Intelligence Technique. *Information Technology and Control*, *38*(1), 2009, p.21–28.

[78] **Nijssen, S.** SBVR: Semantics for Business. *Business Rules Journal*, *8*(10), 2007. Available from: http://www.BR Community.com/a2007/b367.html. [Accessed: 20 Jan 2011].

[79] **Nihalani, N., Silakari, S., Motwani, M**. Natural Language Interface for Database: A Brief Review. *IJCSI International Journal of Computer Science Issues, 8*(2), 2011.

[80] **Normantas, K., Vasilecas, O.** A systematic review of methods for business knowledge extraction from existing software Systems. *Baltic Journal of Modern Computing (BJMC)*, Vol. 1, no 1-2, 2013, p.29-51.

[81] **Normantas, K., Vasilecas, O.** Business rules discovery from existing software Systems. *International Journal of Scientific & Engineering Research, 3*(10), 2012, p.1-7.

[82] **Noy, N., Rector, A.** *Defining N-ary Relations on the Semantic Web*. W3C Working Group Note, 12 April 2006. Available from: http://www.w3.org/TR/swbp-n-aryRelations/ [Accessed 14 Junuary 2016].

[83] **Number of Internet Users by Language.** *Internet World Stats*, Miniwats Marketing Group, 30 November 2015. Available from: http://www.internetworldstats.com/stats7.htm. [Accesed: 15 Mar 2016].

[84] **Ogden, W., Bernick, P.** Using Natural Language Interfaces. *Handbook of Human-Computer Interaction*. Elsevier Science Publishers B.V. (North-Holland), 1996.

[85] **ONTORULE project.** Available from: http://ontorule-project.eu [Accessed: 18 Jan 2012].

[86] **Ontotext GraphDB.** Available from: http://ontotext.com/products/graphdb/ [Accessed: 31 Mar 2016].

[87] **OWL Test Data.** Available from: https://files.ifi.uzh.ch/ddis/oldweb/ddis/research/talking-to-the-semantic-web/owl-test-data/ [Accessed: 05 Apr 2015].

[88] **Paradauskas, B., Laurikaitis, A.** Business Knowlegde extraction using program understanding and data analysis techniques. In *Proceedings of the 15th International Conference on Information and Software Technologies (IT 2009)*, Kaunas, Lithuania, April 23-24, 2009, p.337–354.

[89] **Patel-Schneider, P. F., Motik, B.** *OWL 2 Web Ontology Language Mapping to RDF Graphs* (Second Edition). W3C Recommendation 27 October 2009. Available From: https://www.w3.org/TR/owl2-mapping-to-rdf/ [Accessed: 15 Mar 2016].

[90] **Perrault, C. R., Grosz, B. J.** Natural-language interfaces. *Exploring artificial intelligence*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA 1988, p.133-172.

[91] **Popov, B., Kiryakov, A., Kirilov, A., Manov, D., Ognyano, D., Goranov, M.** KIM - Semantic Annotation Platform. In *Proceedings of the 2nd International Semantic Web Conference (ISWC2003)*, Sanibel Island, Florida, USA, October 20-23, Berlin, 2003, p.484-499.

[92] **Pruddhommeaux, E., Seaborne, A.** *SPARQL Query Language for RDF*. W3C Recommendation. 15 January 2008. Available from: http://www.w3.org/TR/rdf-sparql-query/ [Accessed 06 May 2011].

[93] **Resnik, P.** Access to Multiple Underlying Systems in JANUS. BBN report 7142. Bolt Beranek and Newman Inc., Cambridge, Massachusetts, September 1989.

[94] **Ross, R. G.** *The RuleSpeak® Business Rule Notation.* Available from: http://www.brcommunity.com/b282.php [Accessed 18 June 2013].

[95] **RuleXpress: The business tool for expressing and communicating business rules.** Available from: http://www.rulexpress.com/ [Accessed: 18 Feb 2014].

[96] **Ruppenhofer, J., Ellsworth, M., Petruck, M. R. L., Johnson, C. R., Scheffczyk, J.** FrameNet II: Extended Theory and Practice. Technical Report, ICSI, 2005.

[97] **SBVR Lab 2.0.** Available from: http://www.sbvr.co/ [Accessed: 20 Feb 2014].

[98] **Semantics of Business Vocabulary and Business Rules (SBVR), Version 1.3.** OMG Document Number: formal/2015-05-07, 2015.

[99] **SemantikaLT.** Syntactic-semantic analysis and search system for Lithuanian Internet, corpus and public sector applications (2012-2014). Contracting authority: Information Society Development Committee (IVPK). Supported by Structural funds of EU, No VP2-3.1-IVPK-12-K, 2014.

[100] **Slator, B., Anderson, M., Conley, W.** Pygmalion at the interface. *Communications of the ACM CACM Homepage archive, 29*(7), July 1986, p.599-604.

[101] **Smith, M. K., Welty, C., McGuiness, D. L.** *OWL Web Ontology Language Guide.* W3C Recommendation 10 Feb 2004. Available from: https://www.w3.org/TR/2004/REC-owl-guide-20040210/ [Accessed: 08 Apr 2016].

[102] **Smith, B., Welty, C.** Ontology: Towards a New Synthesis. In *Proceedings of the international conference on Formal Ontology in Information Systems (FOIS '01)*, Ogunquit, Maine, USA, October 17-19, 2001, p.3-9.

[103] **Son, J., Jeong, D., Baik, D.** Practical Approach: Independently Using SPARQL-to-SQL Translation Algorithms on Storage. In *Proceedings of the 4th International Conference on Networked Computing and Advanced Information Management*, Gyeongju, Korea, September 2-4, 2008, p.598-603.

[104] **Spreeuwenberg, S., Anderson, H. K.** SBVR's approach to controlled natural languages. In *Proceedings of the 2009 conference on Controlled natural language*, Marettimo Island, Italy, June 8-10, 2009, p.155-169.

[105] **Spreeuwenberg, S., Gerrits, R.** Business Rules in the Semantic Web, are there any or are they different? In: *Reasoning Web*. Springer Berlin/Heidelberg, Germany, LNCS, Vol. 4126, 2006, p.152–163.

[106] **Stojanovic, N.** On the Query Refinement in the Ontology-Based Searching for Information. In *Information Systems – Special issue: The 15th international conference on advanced information systems engineering (CAiSE 2003), 30*(7), Novermber 2005, p.543-563.

[107] **Ġukys, A., Ablonskis, L., Nemuraitǟ, L., Paradauskas, B.** A Grammar for Advanced SBVR Editor. In *Information Technology and Control, 45*(1), IT&C 2015, p.27-41.

[108] **Ġukys, A., Nemuraitǟ, L.** SemantiniÔ uǵklausÔ vykdymas saugant ontologijŃ reliacinǟje duomenÔ bazǟje. 15-osios tarpuniversitetinǟs magistrantÔ ir doktorantÔ konferencijos "Informacinǟ visuomenǟ ir universitetinǟs studijos" (IVUS 2010) medǵiaga, 2010 m. geguǵǟs 13 d, Kaunas, Lietuva. Kaunas, Vytauto Didǵiojo universitetas. 2011, 15, pp. p.145-151.

[109] **Ġukys, A., Nemuraitǟ, L., Paradauskas, B.** Representing and transforming SBVR question patterns into SPARQL. In *Proceedings of 18th International Conference on Information and Software Technologies (ICIST 2012)*, Kaunas, Lithuania, September 13-14, 2012, p.436-451.

[110] **Ġukys, A., Nemuraitǟ, L, Paradauskas, B., Ġinkeviļius, E.** SBVR based representation of SPARQL queries and SWRL rules for analyzing semantic relations. In *Proceedings of the First International Conference on Business Intelligence and Technology (Bustech 2011)*, September 25-30, Rome, Italy, p.1-6.

[111] **Ġukys, A., Nemuraitǟ, L., Paradauskas, B., Ġinkeviļius, E.** Transformation framework for SBVR based semantic queries in business information systems. In *Proceedings of the second International Conference on Business Intelligence and Technology (Bustech 2012)*, July 22-27, 2012, Nice, France, p.19-24.

[112] **Ġukys, A., Nemuraitǟ, L., Ġinkeviļius, E., Paradauskas, B.** Querying ontologies on the base of semantics of business vocabularies and business rules. In *Proceedings of the 17th international conference on Information and Software Technologies (IT 2011)*, Kaunas, Lithuania, April 27-29, 2011, p.247-254.

[113] **Tang, L. R., Mooney, R. J.** Using Multiple Clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*, Freiburg, Germany, September 2001, p.466-477.

[114] **Tekutov, J., Gudas, S., Denisovas, V.** The Refinement of Study Program Content Based on a Problem Domain Model. *Transformations in Business & Economics, 11*(1(25)), 2012, p.199–212.

[115] **Templeton, M., Burger, J.** Problems in natural-language interface to DBMS with examples from EUFID. In *Proceedings of the first conference on Applied natural language processing (ANLC '83)*, Santa Monica, California, USA, February 1-3, p.3-16.

[116] **Thompson, B. H., Thompson, F. B.** ASK is Transportable in Half a Dozen Ways. *ACM Transactions on Office Information Systems, 3*(2), April 1985, p.185–203.

[117] **Thompson, B. H., Thompson, F. B.** Introducing ASK, A Simple Knowledgeable System. In *Proceedings of the first conference on Applied natural language processing (ANLC '83)*, Santa Monica, California, USA, February 1-3, p.17-24.

[118] **Thompson, C. W., Pazandak, P., Tennant, H. R.** Talk To Your Semantic Web. *IEEE Internet Computing, 9*(6), November 2005, 75-78.

[119] **TrinkÎnas J., Vasilecas O.** A Graph Oriented Model For Ontology Transformation Into Conceptual Data Model. *Information Technology and Control, 36*(1A), 2007, p.126ï132.

[120] **Vasilecas, O., Kalibatienā, D., Guizzardi, G.** Towards a Formal Method for the Transformation of Ontology Axioms to Application Domain Rules. *Information Technology and Control*, *38*(4), 2009, p.271ï282.

[121] **Vasilecas, O., Normantas, K.** Deriving business rules from the models of existing information systems. In *Proceedings of the 12th International Conference on Computer Systems and Technologies (CompSysTech 2011)*, Vienna, Austria, June 16-17, 2011, p.95-100.

[122] **VEPSEM.** *Integration of Business processes and business rules on the base of Business Semantics.* Research project VP131V008F, supported by Lithuanian Education and Science ministry (2013-2015).

[123] **VeTIS.** *Business Rules Solutions for Information Systems Development.* Research project No K B-04/2008, supported by Lithuanian State Science and Studies Foundation (2008-2009).

[124] **Vileinigkis, T.; Ġukys, A.; Butkienā, R.** An approach for semantic search over Lithuanian news website corpus. In *Proceedings of the 7th international joint conference on knowledge discovery, knowledge engineering and knowledge management (IC3K 2015)*, Lisbon, Portugal, November 12-14, 2015, p.57-66.

[125] **Vyġniauskas, E., Nemuraitā, L., Butleris, R., Paradauskas, B.** Reversible lossless transformation from OWL 2 ontologies into relational databases. *Information Technology and Control*, *40*(4), 2011, p.293ï306.

[126] **Vyġniauskas, E., Nemuraitā, L., Paradauskas, B.** Preserving semantics of OWL 2 ontologies in relational databases using hybrid approach. *Information Technology and Control, 41*(2), 2012, p.103-115.

[127] **Vyġniauskas, E., Nemuraitā, L., Ġukys, A.** A hybrid approach for relating OWL 2 ontologies and relational databases. In *Proceedings of the 9th international conference (BIR 2010)*, Rostock, Germany, September 29 – October 1, 2010, p.86–101.

[128] **Waltz, D.** Natural Language Access to a Large Database: an Engineering Approach. In *Proceedings of the 4th international joint conference on Artificial intelligence (IJCAI'75)*, 1975, p.868-872.

[129] **Wang, C., Xiong, M., Zhou, Q., Yu, Y.** Panto: A portable natural language interface to ontologies. In: *Proceedings of the 4th European conference on The Semantic Web: Research and Applications*, Innsbruck, Austria, June 3-7, 2007, p.473-487.

[130] **Warren, D. H. D., Pereira, F. C. N.** An Efficient Easily Adaptable System for Interpreting Natural Language Queries. *Computational Linguistics, 8*(3-4), July-December 1982, p.110-122.

[131] **Warth, A.** Experimenting with Programming Languages. Ph.D. thesis, University of California, Los Angeles, 2009, p.1–122.

[132] **Watt, W. C.** Habitability. *American Documentation*, 19(3), July 1968, p.338ï351.

[133] **Woods, W. A, Kaplan, R. M., Webber, B. L.** The Lunar Science Natural Language Information System: Final Report. Report 2378. Bolt, Beranek, and Newman, Inc. Cambridge, Mass., 1972.

[134] **Xtext domain specific language development framework.** Available from: http://www.eclipse.org/Xtext/ [Accessed: 10 Nov 2011].

[135] **Yue, J.** Transition from EBNF to Xtext. In: MODELS-PSRC 2014, Poster Session and ACM SRC of MODELS, Valencia, Spain, 2014, p.75–80.

[136] **Zoltan-Ford, E.** Reducing Variability in Natural-Language Interactions with Computers. In *Proceeding of the Human Factors Society 28th Annual Meeting*, Santa Monica California, USA, 1984, p.768-772.

8 LIST OF AUTHORóS PUBLICATIONS ON DISSERTATION THEME

Articles in Journals referred in Master Journal List of the Thomson Scientific Information Institute (ISI) with impact factor

1. Ġukys, Algirdas ; Ablonskis, Linas; Nemuraitā, Lina; Paradauskas, Bronius. A Grammar for Advanced SBVR Editor // Information Technology and Control, 45(1), IT&C 2015. ISSN: 1392-124X, p. 27-41.
2. Ġukys, Algirdas; Nemuraitā, Lina; Butkienā, Rita. SBVR based Natural Language Interface to Ontologies // Information Technology and Control, 46(1), IT&C 2017. ISSN: 1392-124X, p. 118-137.

Articles referred in other publications of the Thomson Scientific Information Institute (ISI) (proceedings)

1. Ġukys, Algirdas; Nemuraitā, Lina; Paradauskas, Bronius. Representing and transforming SBVR question patterns into SPARQL // Information and software technologies : 18th International Conference, ICIST 2012, Kaunas, Lithuania, September 13-14, 2012 : proceedings / [edited by] Tomas Skersys, Rimantas Butleris, Rita Butkiene. Berlin, Heidelberg : Springer, 2012. (Communications in computer and information science, Vol. 319, ISSN 1865-0929), ISBN 9783642333071. p. 436-451. DOI: 10.1007/978-3-642-33308-8. [Conference Proceedings Citation Index]. [0,333]
2. Ġukys, Algirdas; Nemuraitā, Lina; Ġinkeviļ ius, Edvinas; Paradauskas, Bronius. Querying ontologies on the base of semantics of business vocabularies and business rules // Information Technologies' 2011 : proceedings of the 17th international conference on Information and Software Technologies, IT 2011, Kaunas, Lithuania, April 27-29, 2011 / Edited by R. Butleris, R. Butkiene ; Kaunas University of Technology. Kaunas : Technologija. ISSN 2029-0020. 2011, p. 247-254. [Conference Proceedings Citation Index]. [0,250]

Articles published in the other reviewed scientific publications (proceedings)

1. Ġukys, Algirdas; Nemuraitā, Lina; Paradauskas, Bronius; Ġinkeviļ ius, Edvinas. SBVR based representation of SPARQL queries and SWRL rules for analyzing semantic relations // Bustech 2011 [elektroninis iġteklius] : the First International Conference on Business Intelligence and Technology, September 25-30, Rome, Italy. [S.l.] : IARIA, 2011. ISBN 9781612081601. p. 1-6. [0,250].;

2. Ġukys, Algirdas; Nemuraitã, Lina. SemantiniÔ uġklausÔ vykdymas saugant ontologijŃ reliacinãje duomenÔ bazãje // Informacinãs technologijos = Proceedings of Master and PhD students conference on informatic : 15-osios tarpuniversitetinãs magistrantÔ ir doktorantÔ konferencijos "Informacinã visuomenã ir universitetinãs studijos" (IVUS 2010) medġiaga, 2010 m. geguġãs 13 d, Kaunas, Lietuva / [Vytauto Didġiojo universitetas, Kauno technologijos universitetas, Vilniaus universitetas]. Kaunas : Vytauto Didġiojo universitetas. ISSN 2029-249X. 2011, nr. 15, p. 145-151. [0,500].;
3. Ġukys, Algirdas; Nemuraitã, Lina; Paradauskas, Bronius; Ġinkeviļ ius, Edvinas. Transformation framework for SBVR based semantic queries in business information systems // Bustech 2012 [elektroninis iġteklius] : the second International Conference on Business Intelligence and Technology, July 22-27, 2012, Nice, France. [S.l.] : IARIA, 2012. ISBN 9781612082233. p. [1-6]. [0,250].;
4. Vileiniġkis, Tomas; Ġukys, Algirdas; Butkienã, Rita. An approach for semantic search over Lithuanian news website corpus // IC3K 2015 : proceedings of the 7th international joint conference on knowledge discovery, knowledge engineering and knowledge management, Vol. 1: KDIR, Lisbon, Portugal, November 12-14, 2015 Setúbal: Science and technology publications, ISBN 9789897581588. p. 57-66. [Indãlis: 0,333]

[Indãlis grupãje: 1,333]

SL344. 2017-05-09, 13,25 leidyb. apsk. l. Tiraġas 16 egz. Uġsakymas 161.
Iġleido Kauno technologijos universitetas, K. Donelaiļ io g. 73, 44249 Kaunas
Spausdino leidyklos ĂTechnologijañ spaustuvã, StudentÔg. 54, 51424 Kaunas