Romanian2SPARQL: A Grammatical Framework approach for querying Linked Data in Romanian

Anca Marginean, Adrian Groza, Radu Razvan Slavescu, Ioan Alfred Letia
Computer Science Department
Technical University of Cluj Napoca
Cluj-Napoca, Romania
anca.marginean@cs.utcluj.ro, adrian.groza@cs.utcluj.ro, radu.razvan.slavescu@cs.utcluj.ro, letia@cs.utcluj.ro

***Abstract*—SPARQL queries have become the current state of the art for querying linked open data repositories. Given the difficulties to create such queries by non-expert users, interfaces that accept questions in natural language and convert them to SPARQL are one solution to this problem. We developed a translator for Romanian natural language to SPARQL. The system exploits the conceptual power of the Grammatical Framework library and the Haskell programming language. The system is validated in the Romanian cultural domain against various Linked Data Sources, including DBpedia.**

Keywords—Natural Languages. Query Languages. Ontologies.

I. INTRODUCTION

With increasing amount of data available as Linked Data, the need of flexibility and easiness in accessing them increases too. Lately, natural language has gained more ground in becoming a viable solution for querying large set of data, especially when it comes to English language, and less in Romanian language case. Situated between natural languages – that are expressive but sometimes ambiguous, and formal languages – that are semantically limited but precise, Controlled Natural Languages (CNL) proved valuable for automatic translation and interfaces with human agents [1]. We envision that ontologies, being explicit specifications of shared conceptualizations, can substantiate CNLs such that the user is not aware of using CNL instead of natural language when stating common utterances. Therefore, our aim is to introduce a CNL, for Romanian, derived from DBpedia ontology, to access existing Linked Data. The CNL is oriented towards querying information about personalities from Romanian culture as they appear in DBpedia. The user is assisted in building queries through an incremental selection over a set of terms derived from DBpedia ontology. The resulting CNL is limited, yet expressive enough.

Section II presents the difficulties related to querying linked data in Romanian language. Section III describes the concrete and abstract grammars developed for SPARQL and Romanian. We discuss the contributions of our work in section IV, while section V concludes the paper.

II. PROBLEM STATEMENT

Following a Linked Data approach, DBpedia structures the knowledge and make it accessible through SPARQL. The information imported from Wikipedia exists in the form of wiki markup and consists of infobox templates, categorization information, images, geo-coordinates, links to external pages, redirects between pages [2], and links across different language editions of Wikipedia. The main component of DBpedia project is DBpedia ontology, consisting of 320 classes with a maximum depth of 5 and 1650 properties. The ontology supports the mapping process and it is continuously extended by the community in the DBpedia Mappings Wiki. Fig. 1 presents part of the subsumption relation for the concepts *Work* and *Person*. It can be observed that the semantic mapping between types of Persons and types of Works is not met: a *book* is not an *artwork*, even though a *writer* is an *artist*. Table 1 describes some properties with their domain and range restrictions.

Being mainly a community effort, DBpedia does not feature an absolute coherence. The search for works done by a person, either artist or scientist, can be done in more ways but none guarantees results. For example, none of the musical works of the Romanian composer George Enescu are related to him through the property *author*. Instead, there is a category *Compositions_by_George_Enescu* that describes some of his main compositions, as it can be seen in the enumerated statements in Table 2. This is different for italian composer Giuseppe Verdi, where the category still exists, but there are also statements relating the composer to its compositions through the property *author*.

Work	*Person*
ArtWork	*Architect*
Painting	*Artist*
Sculpture	*Painter*
WrittenWork	*Sculptor*
Book	*Writer*
Novel	*MusicalArtist*
MultiVolumePublication	*Monarch*
MusicalWork	*Philosopher*
Opera	*Scientist*

Fig 1. Hierarchy for DBpedia concepts WORK and PERSON

978-1-4799-5094-2/14/$31.00 ©2014 IEEE

TABLE 1.PROPERTIES FROM DBPEDIA ONTOLOGY

Property	Domain	Range
author	Work	Person
writer	Work	Person
composer		Person
firstPublicationDate	WrittenWork	Date
literaryGenre	WrittenWork	-
notableWork	Writer	Book
artist	MusicalWork	Person
academicAdvisor	Scientist	Person
notableStudent	Scientist	Person
knownFor	Person	-
birthDate	Person	Date
birthPlace	Person	Place
influenced	Person	Person
movement	Artist	-

TABLE 2 EXAMPLES OF RDF STATEMENTS FROM DBPEDIA

George_Enescu *dbo:birthPlace* dbr:Liveni
dbr:George_Enescu *dbo:birthDate* "19 August 1881"
dbr:Bengal_Nights_(novel) *dbo:author* dbr:Mircea_Eliade
dbr:Bengal_Nights_(novel) *dbo:genre* dbr:Autobiographical_novel
dbr:Bengal_Nights_(novel) *dbo:publication_date* "1933"
dbr:Romanian_Rhapsodies_(Enescu) *dc:subject*
dbr:Category:Compositions_by_George_Enescu
dbr:Category:Operas_by_George_Enescu *skos:broader*
dbr:Category:Compositions_by_George_Enescu
dbr:Oedipe_(opera) *dc:subject* dbr:Category:Operas_by_George_Enescu
dbr:Emil_Cioran *dbo:philosophicalSchool* dbr:Existentialism
dbr:Emil_Cioran *dbo:mainInterest* dbr:Ethics
dbr:Emil_Cioran *dbo:mainInterest* dbr:Literature
dbr:Ion_Heliade_RÄƒdulescu *dbo:movement* dbr:Romanticism
dbr:Mihai_eminescu *dbo:genre* dbr:Romanticism

The lack of uniformity in building statements occurs also in case of movement describing the activity of an artist, either literary or intellectual. There are statements built with *dbo:movement* property, as is the case for Romanian poet and essayist Heliade, but also statements with *dbo:genre* as in case of the poet Mihai Eminescu. Consequently, the complexity of querying DBpedia has two main sources: (i) the dimension of the interrogated knowledge especially in terms of terminology, and (ii) the lack of coherence in making statements. For the first issue, we propose a controlled language described in Grammatical Framework for Romanian language. For the second the system explores more SPARQL alternatives.

Grammatical Framework (GF) is a special purpose programming language for writing grammars [3] based on typed functional programming language. GF grammars are split in two components: abstract and concrete grammars. An abstract grammar defines categories and functions, where each category stands for a set of trees, while functions produce trees of certain category. The linearization types and functions are defined in concrete grammars. For each category, a linearization type is needed and for each function a linearization function. GF includes resource libraries for 29 language, including Romanian. Each library defines lexical categories and rules, respectively phrasal categories and rules.

III. System Description

The proposed system is composed of three modules: GUI module, GF module, and Lexicon Generator.

The core of the system consists of the GF module that includes the abstract grammar and the Romanian and SPARQL concrete grammars. The GF module provides REST services with NLP common functionality applied on the grammars given in Portable Grammar Format (PGF). The GUI module has three main functionalities: (i) it supports an incremental building of the query by suggesting the correct next words; (ii) it gives the translation of the query expressed in Romanian into SPARQL; (iii) it provides access to a SPARQL endpoint of DBpedia where the generated SPARQL query is executed.

The *Lexicon Generator module* populates the lexicon for both Romanian and SPARQL with names and URIs associated to resources from DBpedia. The relevant resources for Romanian cultural domain are mainly persons and their work. GF requires that any token that is going to appear in phrases should be known at the compilation time, therefore the names and URIs must be included in the grammars.

The translation from Romanian to SPARQL is a two steps process: first a parsing from Romanian to an abstract grammar, followed by a linearization from abstract grammar to SPARQL. Fig 2. gives the main modules of the grammars and the relations between them and DBpedia ontology.

A. Abstract Grammar

For domain specific applications, the abstract grammar states the main semantic categories and trees of the language. In the cultural domain, the main topics are: (i) persons doing something, (ii) outcome of a person's actions, and (iii) qualities associated to persons or outcome of their activities. Therefore, the categories are centered around agents, activities and objects on which the activities are performed (Table 3).

The *Agent* category stands for entities that can do actions, while Object are innanimate entities on which activities are performed. *VerbPhrases* stand for activities, either one-place or expecting an object on which they apply. *Place* is the category of named places, e.g., *Suceava*, or of places determined by an agent and his activity, as *Museum* where the paintings of Nicolae Tonitza are exhibited. The categories marked with a * are dependent types [3], inherited from the constructive type theory. A dependent type is a type that takes an argument from other types and can be used for stating stronger conditions of well-formedness than ordinary types. Kind is the other type from which our dependent types take parameters in order to associate one activity to the right object.

1) Functions for Question: The set of functions are defined on previously introduced categories in Table 3.

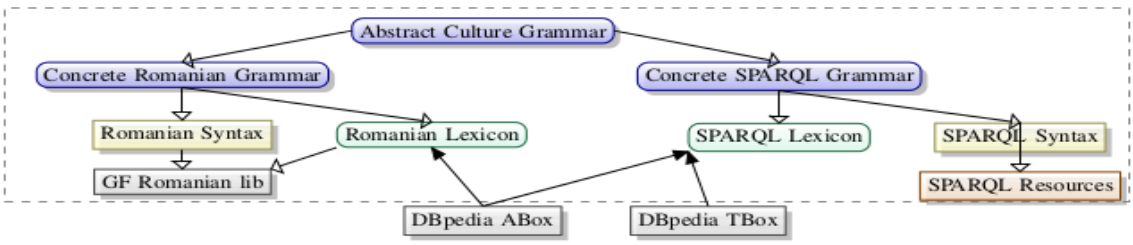

Fig 2.The main components of the grammars

TABLE 3 MAIN CATEGORIES IN ABSTRACT GRAMMAR

Category	Explanation	Example
Agent	Agent able to do something	*Dimitrie Cantemir*
Place	Named place or determined by a pair <agent, action>	*Suceava*
VerbPhras1	Activity associated to one place verb	*study*, *be born*
VerbPhrase 2*	Activity associated to two place verb	*write* a book, *influence* someone
VerbPhrase	any kind of activity	
Object *	object that someone does an action on	*sculpture*
ObjectQuality	the quality of being a result of some activity	*written*
ObjectQualityBy*	the quality of being a result of some activity of a certain agent	*written by someone*
Kind	types derived from concepts in DBpedia ontology, used for associating a certain type of object to the proper activity: write a book vs sculpt a book	*Book*, *Person* *Writing*

. The first function from Fig 3. takes a *Place* as an argument and it corresponds to questions like *Where is Suceava?*, where the expected answer is a geolocation. The second and the third question ask the location, respectively the time of an activity done by an agent, as in *Where was Dimitrie Cantemir born?* or *When did Herta Muller publish The Land of Green Plums?*. Questions 4 and 5 ask for persons related to an agent by an activity, as *Who did Emil Cioran influence?* or *Who influenced Emil Cioran?*. The agent is the one doing the activity in function *WhoVP*, while in function *WhoVPonAgent* the agent is the one on which the action was done. The questions built by functions 6, 7 and 8 ask for object done by agents. Function 6 stands for questions like *What books did Mihail Sadoveanu write?* requesting objects on that the agent did some some activity. Function 7 asks for the same issue, but in a different way: *What are the books written by Mihail Sadoveanu?*. The phrase written by Mihail Sadoveanu is a structure of type *ObjectQualityBy*. The function *WhatObjBy* asks for anything that an agent did, as *What are the works of Dimitrie Cantemir?*. As opposite to the other two functions, the activity that has the object as a result is not mentioned, allowing for answers which include for example both sculptures and novels. The function 9 stands for questions like *Who is Traian Lalescu?*.

```
1.WherePlace2 : Place → Question;
2.WhereAction: Agent→VerbPhrase→Question;
3.WhenAction: Agent→VerbPhrase→Question;
4.WhoVP: Agent→VerbPhrase2 Person→Question;
5.WhoVPonAgent: Agent→VerbPhrase2 Person→Question;
6.WhichObjActedonBy : (k:Kind) →Object k →Agent →VerbPhrase2 k
→ Question;
7.WhatObjActedonBy : (k:Kind) →ObjectQualityBy k→Object k →
Question;
8.WhatObjBy : (k:Kind) →Agent → Object k →Question;
9.WhoAgent :Agent →Ouestion;
```

Fig 3. The main functions for questions

2) Intermediate Functions: The two types of Verbal Phrases are joined in VerbalPhrase category with Action, respectively ActionOn functions. The first one has as a parameter a one place Verbal Phrase, while the second one takes a two place verbal phrases together with an object it applies on. *ActedOnBy* builds a tree corresponding to a VerbalPhrase2 for two-place verbs and the doer agent. For example, *[ActedOnBy Writing VWrite Lucian Blaga]* stands for *written by Lucian Blaga* and it applies only on objects of type *Writing*.

```
Action : VerbPhrase1 → VerbPhrase;
ActionOn : (k:Kind) → VerbPhrase2 k → Object k → VerbPhrase;
ActedOnBy : (k:Kind) → VerbPhrase2 k → Agent → ObjectQualityBy k ;
```

3) Constant functions: Constant functions do not take any argument and are used to define agents, object, verbal phrases, kinds. For dependent type, the category takes one parameter, e.g., *VWrite* is a *Verbal Phrase* of type *Writing*.

```
VWrite                          : VerbPhrase2 Writing;
VCompose, VCreate, Vsculpt : VerbPhrase2 Art;
Book                            : Object Writing;
Sculpture                       : Object Art;
```

B. Romanian Concrete Grammar

Both the lexicon and the syntax of Romanian concrete grammar rely on morphological and syntactical categories and functions defined in GF resource library for Romanian.

1) Romanian lexicon: We declared Romanian verbs corresponding to properties in DBpedia ontology, as *DieV = v_besch129 "a muri", StudyV = v_besch10 "a studia" | v_besch45 "a învăța"*. The functions *v_besch* are taken from GF Romanian Resource library and they give all the inflections of a verb for different persons, number, mode or tense. Reflexive verbs, as the translation *a se naşte* for *to be born*, are linearizated as two place verbs:

VBeBorn= reflexiveVP (P.dirV2 (v_besch106 "naşte")).

An important component of Romanian Lexicon is the set of statements about agents. For each agent, a Noun Phrase is built from a Proper Noun built from the person's name.

ag55 = mkNP (P.mkPN "Ion Barbu" (P.mkPN "Ion").g);

Different to English, in Romanian language the agreement between phrase components takes into consideration gender also and by default, GF considers feminine all proper nouns ending in *a*. The rule is applied on the given name. Exception of this rule are also specified, for example *Luca*.

```
Book=mkCN (P.mkN "carte" P.feminine);
Work=mkCN (P.mkN2 (P.mkN "opera") by8means_Prep)
                (mkNP (mkCN (P.mkN "artă")))
    | mkCN & (P.mkN "lucrare") ;
WhenAction agent act =  mkQS pastTense simultaneousAnt positivePol
                                (mkQCl when_IAdv (mkCl agent act));
```

Fig 4 Romanian linearization for *Book*, *Work*, and *WhenAction*.

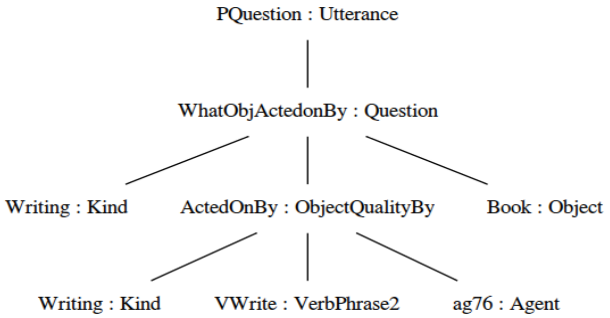

Fig 5: Abstract syntax tree for: ***Care sunt cărțile scrise de către Lucian Blaga***

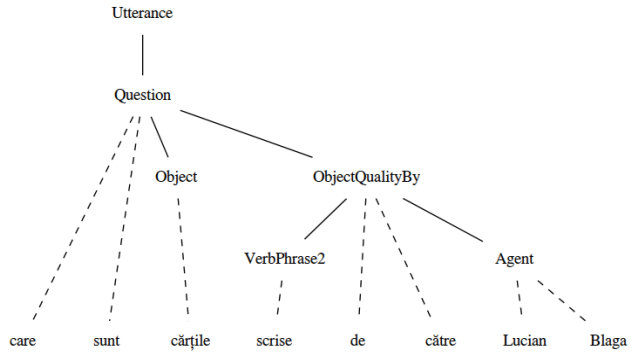

Fig 6: Parse tree for *Care sunt cărțile scrise de către Lucian Blaga*.

*2)Romanian Syntax: F*or each category from the abstract grammar, a linearization type is given, for example *VerbPhrase1* is a verb *V*, *VerbPhrase* is *VP*, *Object* is *CN*, while *ObjectQualityBy* is *AP*. Categories as *CN, PN, V, VP, or AP* are defined by the GF resource library for each language, including Romanian. For each function from abstract grammar a linearization is given, consistent with the declared linearization type. Book is linearized to romanian term *carte* that is feminine, while *Work* is linearized to *operă de artă* or *lucrare* (see Fig 4). The former is built from a relational noun accompanied by the translation *de* of preposition *by*.

For questions about the time of an activity, the linearization of function *WhenAction* creates a Question Sentence from a Question Clause that relates the agent to the activity (Fig. 4). The tense is past simple and the polarity is positive. According to this linearization, the sentence *Când s-a născut Dimitrie Cantemir* is parsed to *WhenAction ag138 VbeBorn,* where *ag138* is the id for the resource Dimitrie Cantemir.

For function *WherePlace2* we give two alternative linearizations: (i) based on existing GF Question Clause for *where* interrogative adverb, (ii) a new way of building *where* Question Clauses around the verb *a se afla*. Thus, the asbstrat syntax tree *[WherePlace2 [NamedPlace Suceava]]* is linearized to *unde este Suceava* or *unde se afla Suceava*. A new interrogative determiner *ce_IDet* was introduced for function *WhichObjActedonBy,* similar to existing *which_IDet*. The reason is to allow both interrogative forms *ce carti* and *care sunt cărțile* for *which books*, and *which are the books*.

Fig 5.and Fig 6 show the abstract syntax tree and the parse tree for a question of type *WhatObjActedonBy*. The subtree *ActedOnBy* corresponds to an adjectival phrase built from the participle of the verb associated to an activity together with the linearization of the agent doing the activity.

C. SPARQL concrete grammar

GF provides a library for the Romanian, but not for SPARQL. Therefore we defined operators for working with complete or incomplete triplets.

1)Resource module for SPARQL: Each activity can be characterized according to three dimensions: *where*, *when*, *what*. The *Context* parameter enumerates these three values. *DBTCategory* is an enumerated type with values corresponding to concepts from which DBpedia *Categories by Name* can be created, such as *Compositions_By_George_Enescu, Books_By_Neagu_Djuvara*.

```
Context = Where | When | What;    Position = Before | After;
DBTCategory = Composition | Book | Work | Sculpture ...
```

The main types proposed in Resource module are *Noun*, and *VPTriplet*. The former linearizes to a string, while the later to a record including two components of type string: one called *p* for the property and one named *s* for complete or incomplete triplet derived from the property *p*.

```
Noun:Type = {s:Str};    VPTriplet:Type = {p:Str; s:Str};
```

The most important operators are defined for building internal *Nouns, Resources* and *VPTriplets*. *VPTriplets* can be built from (i) a string representing the property, (ii) two *VPTriplets* in case of *VerbPhrase1*, one *VPTriplet* for each dimensions *when* and *where*, (iii) three *VPTriplets* for *VerbPhrase2*, one for each value of *Context*, and a *DBTCategory* indicating the type of the *DBpedia Category* associated to the property, and (iv) a VPTriplet and a variable.

2) SPARQL lexicon: For relevant properties from DBpedia ontology a VPTriplet is built, while for each agent an internal Resource. The properties are manually introduced in order to ensure consistency between natural language verb phrases and DBpedia properties. The set of resources associated to agents are automatically generated by *Generator module.*

```
LucianBlaga = mkResource „Lucian_Blaga”;
author_P = mkVPTriplet „dbpedia-owl:author”;\
birthDate_P = mkVPTriplet „dbpedia-owl:birthDate”;
birthPlace_P = mkVPTriplet „dbpedia-owl:birthPlace”;
```

3) SPARQL Syntax: The linearization types for the main categories are records with different components:

```
Agent={db:Noun; categs:DBTCategory=>Str; s:Str};
Object, Place={v:Str; body:Str};
VerbPhrase1, ObjectQualityBy = {v:Str; body: Context=>Str; it:Position};
VerbPhrase2={v:Str; p:Context=>Str; categ: DBTCategory};
```

For each constant function of type *Object* a linearization is defined to a record with two components: a variable name kept in *v*, respectively an incomplete triplet in *body* (see Fig 7). For each function with type *VerbPhrase1* or *VerbPhrase2* the functions *mkVP*, respectively *mkDBVP* defined in Resource module are used to build the corresponding structure. The

difference between linearization of *VerbPhrase1* and *VerbPhrase2* comes from the fact that for the later an alternative based on DBpedia *Categories_By_Name* is built besides the main structure. The last parameter of function *mkVP* states the position of the unknown component relative to the incomplete triplet. For example, the URIs for the agent will be concatenated in front of *dbpedia-owl:birthPlace ?birth* in order to query for agent's *birthPlace*.

The linearization for *WhichObjActedonBy* (Fig. 7) is a SPARQL query with more alternatives. For instance, when searching for compositions of George Enescu, the WHERE clause of the query is a UNION between (i) results obtained with *?v author George_Enescu*, (ii) *?v dcterms:subject Compositions_By_George_Enescu, (iii) ?cat skos:broader Compositions_By_George_Enescu. ?v dcterms:subject ?cat*. Skos Vocabulary is used for relations between Wiki Categories.

```
Book ={v="?book"; body="rdf:type" ++ Book_R.s};

VBeBorn=mkVP birthDate_P birthPlace_P "?birth" Before;

VStudy =mkVP (mkVPTriplet "") almaMater_P "?study" Before;

VCompose =mkDBVP completionDate_P (mkVPTriplet "") composer_P "?compose" Composition;

VWrite =mkDBVP pubdate_P country_P author_P  "?write" Book;


WhichObjActedonBy _ o agN vp2 =

 let ss= (mkObj (vp2.p!What) vp2.v After).s;

                    ag= agN.db;  categ=vp2.categ;

                    cs&=agN.categs;

in „SELECT"++vp2.v++"where{" ++insertEntity ag.s ss (vp2.v++ o.body) After After ++"}"++

     „UNION {"++mkSubjectTriplet vp2.v (cs!categ) ++(mkAbstractTriplet vp2.v) ++ "}"++

     „UNION {"++(mkSkosBroaderTriplet vp2.v (cs!categ))++"}}";
```

Fig 7. SPARQL linearization for *Book, VbeBorn, VStudy, VCompose, VWrite, and WhichObjActedonBy*

D. Generator module

It populates the lexicons with structures associated to resources from Dbpedia. GF requires all tokens to be known at the compilation phase, therefore it is not possible to build tokens as Books_By_Lucian_Blaga at running time from *Books* and writer's name. For this reason, the *Generator* queries DBpedia for the targeted persons and works and builds different structures in abstract and concrete grammars.

For retrieving the names of the persons the module runs a query built on Wiki Categories. There are 6 dimensions: *literature*, *artists*, *philosophers*, *scientists*, *composers*, and *royalty* with their coresponding DBpedia categories. For each category the search explored the hierarchy for 2 levels deep. For example, the hierarchy of *Romanian royalty* category includes *Rulers of Transylvania* and *Romanian monarchs*; the first one includes at its turn subcategory *Voivodes of Transylvania*, and the second *Kings of Romania*. The final query is built as a UNION off all six dimensions and three unions for each dimension's hierarchical search. We query all 6 dimensions in one single query in order to take a person only once, aware that persons may activate along more cultural areas. The module extracts romanian personalities, but also italian and british (Table 4).

E. Results

The natural language interface is shown in Fig 8 and the system is available online at *http://193.226.5.115/~isg/Romanian2Sparql*. Known questions are of type *care, ce, cine, când, unde* meaning *which, what, who, when, where.* The query is incrementally built through iterative selection from the choices given by PGF services or it can be entered as text. The generated SPARQL query is executed in the DBpedia endpoint. In Table 5 we give some examples of questions expressed in our CNL with the corresponding abstract tree.

TABLE 4. QUANTITATIVE COMPARISON

	Romanian	Italian	British
Literature	281	827	3602
Artists	90	1026	3130
Philosophers	29	87	251
Scientists	135	183	3627
Composers	27	288	747
Royalty	129	492	813
Total	*690*	*2902*	*12169*
Distinct	*588*	*2609*	*11349*

TABLE 4. KNOWN QUESTIONS

1. Unde este Muzeul Satului?
PQuest (WherePlace2 (NamedPlace (Lmuseum VillageMuseum)))
2. Unde este castelul in care a trăit Ferdinand I de Romania?
PQuest (WherePlace2 (Pplace Castle ag579 (Action VLive)))
3. Când s- a născut Traian Lalescu?
PQuest (WhenAction ag384 VBeBorn)
4. Pe cine a precedat Stephen III of Moldavia?
PQuest (WhoVP ag473 VPreceed)
5. Cine a influențat- o pe Herta Muller ?
PQuest (WhoVPonAgent ag174 VInfluence)
6. Cine l- a influențat pe Mircea Eliade?
PQuest (WhoVPonAgent ag135 VInfluence)
7. Ce regiune a condus Neagoe Basarab?
PQuest (WhichObjActedonBy Administrative Region ag208 VLead)
8. Care sunt sculpturile sculptate de către Constantin Brâncuşi?
PQuest (WhatObjActedonBy Art (ActedOnBy Art VSculpt ConstantinBrancusi) Sculpture))
9. Cine este Spiru Haret?
PQuest (WhoAgent ag396)

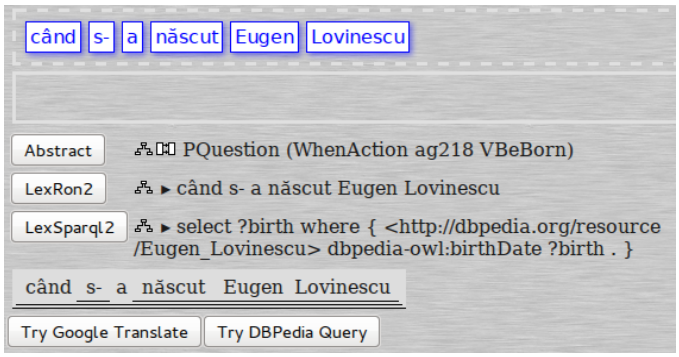

Fig 8: Natural language interface for Romanian.

IV. DISCUSSION AND RELATED WORK

From our knowledge, the only existing approach for translating queries expressed in Romanian language to SPARQL comes from Multilingual Online Translation (MOLTO) project, consisting of a framework for integrating and accessing museum linked data [4, 5]. Both Museum component of MOLTO and our system define grammars in GF for translating a query expressed in natural language to SPARQL with different domain and source language. In [4] the focus is on paintings and their features, while our focus is on persons and their activities, aiming to recognize expressions dealing with dynamic aspects, as the time someone published a book, or the place where something happened. We focus also on various activities of the agents: as paint, write, study, rule, die and their corresponding outcomes.

Our system for translating Romanian language queries into SPARQL syntax fills, in our view, an important gap among the existing linguistic resources for Romanian language. In the survey [6], the available technologies are categorized from different levels: sub-syntactic, syntactic, semantic. For the Romanian language several large annotated corpora do exist (George Orwell's novel 1984, Plato's Republic, ROCO), lexicons (WEB-DEX, CONCEDE, EUROVOC) [6] with the corresponding tools for exploiting these dictionaries (http://dexonline.ro/unelte). None of these resources deal with translation between a natural language and a formal language. The Romanian grammar in [7] includes 866 grammatical rules and 320 affixes, which have been used for the development of a morphological vocabulary of 30,000 words. For the natural language part of our work we based on the GF resource library for Romanian developed in [8]. Our morphological vocabulary was generated only for the cultural domain.

Semantic Query and Update High-Level Language (SQUALL) [9] allows querying an RDF dataset in CNL syntax. The language abstracts over the relational algebra constructs of SPARQL without sacrificing its expressivity. SQUALL relies on Montague grammars described only for English language. SQUALL and Romanian2SPARQL share the approach based on CNL with associated grammars. Our system exploits the multilingual facility of GF to handle numerous difficulties of handling Romanian language. The combination between Attempto Controlled English (ACE) and GF from [10]. Approaches for verbalization based on ontology is introduced in [11] for English and Greek languages. A CNL for editing ontology is presented in [12] based on ACE. Besides DBpedia, YAGO2 [13] focuses on automatically extracting and publishing structured knowledge from Wikipedia. While the DBpedia taxonomy is manually developed and maintained, YAGO integrates the WordNet taxonomy, which leads to a higher number of classes. Expanding our system to manage this richer taxonomy is current work in our case.

V. CONCLUSION

We introduced a system for translating from Romanian language to SPARQL. We defined grammars for Romanian language based on resource library of GF. A resource library for SPARQL was introduced and applied to concrete grammars for querying DBpedia. The Romanian and SPARQL lexicons were automatically populated with instances from DBpedia

ACKNOWLEDGMENT

Part of this work was supported by the Romania-Moldova Bilateral Contract entitled "ASDEC - Structural Argumentation for Decision Support with Normative Constraints" and by the intern research project at Technical University of Cluj-Napoca, Romania: "GREEN-VANETS: Improving transportation using Car-2-X communication and multi agent systems".

REFERENCES

[1] T. Kuhn, "A survey and classification of controlled natural languages," Computational Linguistics, vol. 40, no. 1, pp. 121–170, 2014

[2] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia," Semantic Web Journal, 2014.

[3] A. Ranta, "The GF Resource Grammar Library," Linguistic Issues in Language Technology, vol. 2, no. 2, 2009

[4] D. Dannells, A. Ranta, R. Enache, M. Damova, and M. Mateva", "Multilingual access to cultural heritage content of the semantic web," in Proc. of Language Technologies for Cultural Heritage @ACL'2013.

[5] D. Dannells, M. Damova, R. Enache, and M. Chechev, "A framework for improved access to museum databases in the semantic web," in RANLPDigHum, 2011, pp. 3–10.

[6] D. Cristea and C. Forascu, "Linguistic resources and technologies for Romanian language," Com. S..J. of Moldova, vol. 14, no. 1, p. 40, 2006.

[7] E. Boian, C. Ciubotaru, S. Cojocaru, A. Colesnicov, V. Demidova, and L. Malahova, "Lexical resources for romanian-a project overview," in Symposium on Intelligent Systems and Applications, 2003, pp. 19–20.

[8] R. Enache, A. Ranta, and K. Angelov, "An open-source computational grammar for Romanian," in Comp. Linguistics and Intell. Text Proc., A. Gelbukh, LNCS, Springer Berlin, 2010, vol. 6008, pp. 163–174.

[9] S. Ferr, "Squall: A controlled natural language as expressive as sparql 1.1," in Natural Language Processing and Information Systems, sLNCS, E. Mtais et all., Springer Berlin , 2013, vol. 7934, pp. 114–125.

[10] K. Kaljurand and T. Kuhn, "A multilingual semantic wiki based on Attempto Controlled English and GF," CoRR, vol. abs/1303.4293, 2013

[11] K. Kaljurand, "ACE View — an ontology and rule editor based on Attempto Controlled English," in 5th OWL Experiences and Directions Workshop (OWLED 2008), Karlsruhe, Germany, 26–27 October 2008.

[12] I. Androutsopoulos, G. Lampouras, and D. Galanis, "Generating natural language descriptions from owl ontologies: the naturalowl system," Journal of Artificial Intelligence Research, vol. 48, pp. 671–715, 2013.

[13] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum, "YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia," Artificial Intelligence, vol. 194, pp. 28–61, 2013