Answering Natural Language Queries about Rehabilitation Robotics Ontology on the Cloud

Zeynep Dogmus, Volkan Patoglu and Esra Erdem
Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey

Keywords: Rehabilitation Robotics, Query Answering, Natural Language, Ontologies.

Abstract: We introduce a novel method to answer natural language queries about rehabilitation robotics, over the formal ontology REHARBO-ONTO. For that, 1) we design and develop a novel controlled natural language for rehabilitation robotics, called REHARBO-CNL; 2) we introduce translations of queries in REHARBO-CNL into SPARQL queries, utilizing a novel concept of query description trees and description logics concepts; 3) we use an automated reasoner to find an answer to the SPARQL query. To facilitate the use of our method by experts, we develop an intelligent, interactive query answering system, using Semantic Web technologies, and make it available on the cloud via Amazon web services. This interface guides the users to express their queries in natural language and displays the answers to queries in a readable format, possibly with links to detailed information. Easy access to information on REHARBO-ONTO through complex queries in natural language may help engineers inspire new rehabilitation robot designs, while also guiding practitioners to make more informed decisions on technology based rehabilitation.

1 INTRODUCTION

Recently, the first formal ontology about rehabilitation robotics, called REHARBO-ONTO, has been designed and developed in OWL (Web Ontology Language) (Horrocks et al., 2003; Antoniou and van Harmelen, 2004), and made available on the cloud (Dogmus et al., 2013; Dogmus et al., 2012), to facilitate access to various kinds of information about the existing robots. Also this effort of having a structured representation of information about rehabilitation robotics is inline with the standardization efforts of by European Network on Robotics for Neurorehabilitation, and IEEE-RAS Ontologies for Robotics and Automation Working Group.

Such a formal ontology allows rehabilitation robotics researchers to learn various properties of the existing robots and access to the related publications to further improve the state-of-the-art. Physical medicine experts also can find information about rehabilitation robots and related publications to better identify the right robot for a particular therapy or patient population. Such requested information can be obtained from REHARBO-ONTO by expressing the requested information as a formal query in a query language, such as SPARQL (PrudHommeaux et al., 2008), and then by computing answers to these queries by using a state-of-the-art automated reasoner, such as PELLET (Sirin et al., 2007). However, expressing the requested information as a formal query by means of formulas is challenging for many users, including robot designers and physical medicine experts.

This paper is concerned about the process of query answering over REHARBO-ONTO, and making it easier for the users to express their queries in a natural language and to obtain answers to their queries automatically. By this way, the users are not required to be familiar with the underlying formal query language, Semantic Web technologies, or automated reasoners. Easy access to information on REHARBO-ONTO through complex queries may help engineers inspire new rehabilitation robot designs, while also guiding practitioners to make more informed decisions on technology based rehabilitation.

Our contributions can be summarized as follows. First, we design and develop a novel controlled natural language for rehabilitation robotics, called REHARBO-CNL, to express queries. We introduce a method of translating natural language queries in REHARBO-CNL into formal SPARQL queries.

1http://www.rehabilitationrobotics.eu/
2http://www.ieee-ras.org/industrial/standards.html

DOI: 10.5220/0005081400750083
In Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD-2014), pages 75-83
Copyright © 2014 SCITEPRESS (Science and Technology Publications, Lda.)
This translation utilizes two intermediate representations: a novel tree structure, called a Query Description Tree (QDT), and Description Logics (DL) concepts. Once the natural language query is transformed into a formal query, we use Pellet to find answers to the query. To guide the users throughout the whole process of expressing queries in REHABROBO-CNl, and to show the computed answers to them in an understandable way, we also design and develop an interactive, intelligent user interface. The overall system is made available on the cloud via Amazon Elastic Compute Cloud (Amazon EC2)—a web service that provides resizable compute capacity in the cloud, and makes web-scale computing easier for developers.

2 REHABROBO-CNl: A CONTROLLED NATURAL LANGUAGE FOR REHABILITATION ROBOTICS

Reasoning over REHABROBO-ONTO is done by means of answering questions posed by the user in natural language. To overcome the ambiguities in the vocabulary and grammar of natural languages, we introduce a Controlled Natural Language (CNL), a subset of a natural language with a restricted vocabulary and grammar. A CNL is essentially a formal language, and thus it is not difficult to convert a CNL to a logic-based formalism. In that sense, a CNL facilitates the use of automated reasoners to find answers to queries expressed in a CNL.

In order to express queries about rehabilitation robots, we designed and developed a new CNL, called REHABROBO-CNl. Although we designed REHABROBO-CNl considering REHABROBO-ONTO, it is possible to expand it to support queries about integrated knowledge resources (e.g., patients, diseases, genetic information). Some example queries in REHABROBO-CNl are listed below:

- What are the robots that target some wrist movements with actuation='series elastic'?
- What are the effort metrics that are evaluated by some robots with active degree of freedom ≥ 2?
- What are the publications with clinical study and that reference some robots with active degree of freedom ≥ 2?

With REHABROBO-CNl, it is possible to construct queries that contain nested relative clauses, disjunctions, conjunctions, negations, and quantifications; such as some, all, any, none.

To eliminate the ambiguities in nesting of conjunctions and disjunctions, REHABROBO-CNl provides two ways of constructing a query: A query in REHABROBO-CNl should either be in Conjunctive Normal Form (CNF), or in Disjunctive Normal Form (DNF). In other words, REHABROBO-CNl supports conjunctions of simple disjunctions, and disjunctions of simple conjunctions. An example of a query in CNF is as follows.

What are the robots with mechanism type='hybrid' and (with motion capability = 'grounded' or with functionality='clinic') and that target some wrist movements?

The part of the grammar of REHABROBO-CNl that describes this query is shown in Table 1. The italic functions in the grammar are used to extract relevant information from REHABROBO-ONTO. These ontology functions are described in Table 2.

The information extracted with the ontology functions are coupled by their relevance. For instance, only the verb “reference” can appear after the type Publications. By matching types with verbs, it is possible to prevent semantically wrong queries like “What are the publications that target some shoulder movements?”. Similarly, it is necessary to match verbs with types.

In addition to types and verbs, types are matched with the relevant nouns. For instance, control modes are matched with robots whereas actuation is matched with movements. Further, the values for the nouns are extracted to allow suitable entries from the users. The values can be considered as ranges of the nouns, that the user can choose from.

3 TRANSLATING QUERIES FROM REHABROBO-CNl INTO SPARQL

To answer a query in REHABROBO-CNl using automated reasoners, we transform the query into the formal query language SPARQL with the following steps.

1. We parse the query as a tree.
2. We traverse the tree and obtain a description logics (DL) concept description.
3. We transform the DL concept into a SPARQL concept.
4. We form a SPARQL query.
Parsing a Query. To parse a REHABROBO-CNL query, we introduce the concept of a Query Description Tree (QDT). A QDT is a rooted, directed tree that consists of five types of nodes:

- **root-node**: Represents the sort of the query.
- **that-node**: Represents a relative clause beginning with “that”.
- **with-node**: Represents a relative clause beginning with “with”.
- **or-node**: Represents a conjunction.
- **and-node**: Represents a disjunction.

Every root/that/with-node characterizes a phrase and a type/instance. An and/or-node cannot be a leaf. For each path from the root node to a leaf node, there can be at most one and-node and one or-node. With-nodes are leaves only. That-node has one child only. Consider, for instance, the QDT in Figure 1 for the query: “What are the robots that target some shoulder movements with actuation=electrical and (with transmission=cable drive or with transmissions=direct drive)?” The root denotes the beginning of the query “What are the robots...”. According to the root, the answer to the query will contain robot names only. Since the query is about robots, the type contained in the root is “robot”.

The relative clause about these robots is the child of the root. Since this relative clause starts with “that”, it is a that-node; the type contained in this node is “shoulder movement”.

The query continues with a conjunction of two relative clauses. Clauses joined with a conjunction (resp., disjunction) are characterized by an and-node (resp., or-node) as their parent.

One of conjoined the relative clauses starts with “with”, so it is a with-node. The other relative clause is a simple disjunction, so it is an or-node. It disjoins two clauses, each starting with “with”; so it has two children that are with-nodes. The with-nodes include values of properties instead of types.

From QDT to a DL Concept. The tree representing the query, in fact, represents a concept. While creating a query, we define a new concept and search for its instances. Retrieved instances that fit our description are the answers to our query.

By a depth-first traversal of a QDT, we represent the corresponding concept in Description Logics (DL) as described in Algorithm 1. For instance, for the QDT in Figure 1, the algorithm returns the following concept:

```
[Robot ⊓ \∃ targets.ShoulderMovements
	∃ actuation.{electrical}
	∩ (∃ transmission.{cabledrive} ∪
	∃ transmission.{directdrive})).
```

It starts from the root node. Since the associated class of the node is “robot”, our concept description starts with Robot. The child of the root is an and-node, so the algorithm calls transform recursively for each grandchild of the root node and conjoins the results by \(\sqcap\). For the that-node, the algorithm calls transformThatNode (Algorithm 2). The that-node has no child, it involves the quantifier “some” over its associated class “shoulder movements”. Therefore, transformThatNode returns the
From a DL Concept to a SPARQL Concept. To obtain a SPARQL concept from a DL concept, we utilize some of the existing translations in related publications, such as (Orsi, 2011) and (Fernandes, 2009). We also introduce some novel transformations. Some transformation examples are shown in Table 3. The transformations without a citation are the novel transformations.

Consider the inverse role transformation in Table 3. DL representation of this concept corresponds to the first-order formula, \( \exists x. \text{targets}(x, y) \land \text{Robot}(x) \), which expresses that “there exists a robot \( x \) that targets a movement \( y \)”. Our transformation to SPARQL includes two triples, having a common variable \( x \). The variable \( x \) should satisfy two conditions: it must be a robot and it must target a movement \( y \). According to the semantics of AND (Perez et al., 2006), the result contains the mappings of \( x \) and \( y \) to the nodes in the ontology, that agree on the nodes that correspond to \( x \). This corresponds to the existential restriction in the first-order formula, that should satisfy two conditions combined with a conjunction. Therefore, evaluations of the DL concept and the SPARQL concept return the same result.

Consider the complement transformation in Table 3. DL representation of this concept contains a negated existential quantifier. SPARQL transformation contains a triple covered with FILTER NOT EXISTS. The triple searches for a mapping of variable \( x \) to a node, that is related to another node AssistOn with an edge that characterizes has_Name relation. This

![Figure 1: Tree representation of the sample query.](image-url)

Algorithm 1: transform.

Input: A tree \( T \) representing the concept that the user described
Output: A DL concept description \( Q \) that represents the concept in \( T \)

\[ Q \leftarrow 0; \]
\[ n \leftarrow \text{first (root) node in } T; \]
\[ \text{if } n \text{ is a root-node then} \]
\[ Q \leftarrow Q \sqcap n.\text{class}; \]
\[ \text{foreach child node } c \in n.\text{children do} \]
\[ Q \leftarrow Q \sqcap \text{transform}(c); \]
\[ \text{else if } n \text{ is a that-node then} \]
\[ Q \leftarrow Q \sqcap \text{transformThatNode}(n); \]
\[ \text{else if } n \text{ is a with-node then} \]
\[ Q \leftarrow Q \sqcap \text{transformWithNode}(n); \]
\[ \text{else if } n \text{ is an and-node OR } n \text{ is an or-node then} \]
\[ \text{tempQ} \leftarrow 0; \]
\[ \text{foreach child node } c \in n.\text{children do} \]
\[ \text{if } n \text{ is an and-node then} \]
\[ \text{tempQ} \leftarrow \text{tempQ} \sqcap \text{transform}(c); \]
\[ \text{else} \]
\[ \text{tempQ} \leftarrow \text{tempQ} \sqcup \text{transform}(c); \]
\[ Q \leftarrow Q \sqcap (\text{tempQ}); \]
return \( Q \).
Table 3: DL to SPARQL Transformation Examples.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL</th>
<th>SPARQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role (Orsi, 2011)</td>
<td>targets</td>
<td>?x ns:targets ?y.</td>
</tr>
<tr>
<td>Complement</td>
<td>¬∃name. {AssistOn}</td>
<td>FILTER NOT EXISTS {?x ns:has_Name 'AssistOn'.}</td>
</tr>
<tr>
<td>Inverse Role</td>
<td>?targets • Robot</td>
<td>?x ns:targets ?y.</td>
</tr>
<tr>
<td></td>
<td>?x rdf:type ns:RehabRobots.</td>
<td></td>
</tr>
<tr>
<td>hasValue Restriction (Orsi, 2011)</td>
<td>∃name. {AssistOn}</td>
<td>?x ns:has_Name 'AssistOn'.</td>
</tr>
<tr>
<td></td>
<td>?y rdf:type rr:RehabRobots.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FILTER NOT EXISTS {?x rr:reference ?y2.}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>?y2 rdf:type rr:RehabRobots.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>?x ns:has_Functionality 'clinic'.</td>
<td></td>
</tr>
<tr>
<td>Union (Fernandes, 2009)</td>
<td>⊓∃actuation. {electrical}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>⊓∃transmission. {cabledrive}.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>⊓∃transmission. {directdrive}.</td>
<td></td>
</tr>
</tbody>
</table>

corresponds to an existential restriction. However, we do not want such mappings of \( x \). According to the semantics of \( \text{NOT EXISTS} \) in a filter expression (Pérez et al., 2006), \( \text{FILTER NOT EXISTS} \{ C \} \) is satisfied if the mapping of \( C \) is an empty set. Therefore, there should not be any mapping of the variables in \( C \) to a node in the ontology. The result that is returned from our SPARQL concept does not contain any node that satisfies the condition in the triple, and that corresponds to a negated existential restriction: all \( x \) must not have name AssistOn. Therefore, evaluations of the DL concept and the SPARQL concept return the same result.

Finally, consider the universal restriction example in Table 3. DL description of this concept represents the publications that reference all robots, and for that, it contains a universal quantifier. To represent this concept with SPARQL we need to describe such publications by making sure that there is no robot in the ontology that is not referenced by that publication. To describe such publications in SPARQL we use an expression constructed with two \( \text{FILTER NOT EXISTS} \). Since a universal restriction such as \( \forall x A(x) \) corresponds to a negated existential restriction \( \neg \exists x \neg A(x) \), each \( \text{FILTER NOT EXISTS} \) operator in the SPARQL query corresponds to a negation.

By applying these transformations, the DL concept that is obtained from the QDT in Figure 1

\[
\text{Robot} \sqcap \exists \text{targets.} \text{ShoulderMovements} \sqcap \exists \text{actuation.} \{\text{electrical}\} \sqcap \{\exists \text{transmission.} \{\text{cabledrive}\} \sqcup \exists \text{transmission.} \{\text{directdrive}\}\}
\]

is transformed into the following SPARQL concept:

\[
\text{?robot1 rdf:type rr:RehabRobots.} \\
\text{?robot1 rr:targets ?movement1.} \\
\text{?movement1 rr:has_Actuation 'electrical'.} \\
\text{(?movement1 rr:has_Transmission 'cable drive'.)} \\
\text{UNION} \\
\text{(?movement1 rr:has_Transmission 'direct drive'.)}
\]

Note that these transformations are necessitated by some queries supported by our approach, that involve negation (e.g., publications that do not reference any robots with motion capability = ‘grounded’), passive verbs (e.g., movements that are targeted by robots), and universal quantifiers (e.g., robots that target all foot movements).

Obtaining a SPARQL Query. After we transform a DL concept into a SPARQL concept, we can construct
a SPARQL query as follows. We start with a PREFIX part and we declare the namespace (the location of an ontology on the Web) of REHABROBO-ONTO. Next, we continue with a SELECT clause. The instances of type Robot, by themselves, are not meaningful to the users. Thus, we want to display the names of the instances to the users. We specify it with an additional triple in the beginning of the WHERE clause, and continue the clause with the transformed SPARQL concept:

```
PREFIX rr: <http://.../RehabOnto.owl#>
PREFIX rdf: <http://.../22-rdf-syntax-ns#>

PREFIX dd: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rr: <http://www.semanticweb.org/ontologies/2004/1/OntologyDescription#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?name WHERE {
  ?robot rr:has_Name ?name.
  ...
}
```

Once we obtain a SPARQL query, we can use the DL reasoner PELLET to find answers to queries, through the Jena framework.
4 INTELLIGENT USER INTERFACE FOR QUERY ANSWERING

We have designed an interactive, intelligent user-interface to guide users to express their natural language queries about rehabilitation robots, and to present the answers to their queries with links to detailed information.

The main user interface for querying includes a drop-down list, showing the possible ways to begin a query. Then, according to the user’s choices, it provides different types of features. It provides auto-completion to help users enter values for nouns that correspond to data properties of type string. If the user should choose a concept among a hierarchy, then it displays an accordion view and enables the user to click on the option s/he wants. In addition, it allows multiple selection of values for relational properties. For functional properties, user is able to select multiple items for inequality. User can choose a number of options among “less than or equal”, “more than or equal”, “equal” and “not equal” while entering values for a data property of type integer or float. Figure 2 illustrates some parts of constructing the query “What are the robots that target some wrist movements with actuation='series elastic'?” with this interface.

How the results of a query is displayed to user depends on the query. For instance, if the query is about robots, then the user sees the names of the retrieved robots. If the query is about movements or metrics, then the user sees the concept names instead of the instance URIs which would make no sense to the user.

Note that, since the transformation of a query is designed to construct the query over REHABROBO-ONTO, the SPARQL query resulting from the transformation process that is displayed to the user is always valid in the context of REHABROBO-ONTO. Also the transformation of a query in REHABROBO-CNL to SPARQL is deterministic, and does not involve any ambiguities. Hence, the accuracy of the constructed SPARQL query is always 100%.

Also note that the transformation of a query in REHABROBO-CNL into a SPARQL query is asymptotically linear in time, in the size of the query: parsing a REHABROBO-CNL query into a QDT is done in linear time thanks to the intelligent-user interface; transformation of a QDT into a DL concept requires traversing the QDT once and thus done in linear time; transformation of a DL concept into a SPARQL query requires going over the DL concept once and thus done in linear time. Indeed, the transformation is quite efficient in terms of computation time: it is observed that the transformation a query in REHABROBO-CNL into a SPARQL query usually takes less than a few seconds of CPU time.

5 RELATED WORK

Development of natural language interfaces that provide query answering over ontologies has been subject of research for many years (Bernstein and Kauf-
6 DISCUSSION AND CONCLUSION

We have introduced a novel method to answer natural language queries about rehabilitation robotics, over the first formal rehabilitation robotics ontology \textsc{RehabRobo-Onto}. We have developed an intelligent, interactive query answering system, using Semantic Web technologies, and deployed it on the cloud via Amazon web services.\footnote{\url{http://ec2-54-228-52-230.eu-west-1.compute.amazonaws.com/rehabrobo/}} Guiding the users to express their complex queries in a natural language and displaying the answers to queries in a readable format, this user interface may help engineers inspire new rehabilitation robot designs and practitioners to make more informed decisions on technology based rehabilitation.

In our approach, queries are specified in a controlled natural language, called \textsc{RehabRobo-CNl}, whose vocabulary and grammar are designed and developed specifically for \textsc{RehabRobo-Onto}. This limits the expressiveness of questions to some extent. On the other hand, it also has a strong advantage not only by addressing the ambiguity and habitability problems of natural languages, but also by allowing more general forms of complex queries (some of which cannot be handled by the ontology systems with a full natural language interface).

Our approach transforms a CNL query into a SPARQL query, by means of a DL concept rather than directly. This provides us a flexibility of being able to utilize DL reasoners for query answering. It also provides us a logical formalism to study the formal properties of the specification language and reasoning methods in the future.

Our ongoing work involves evaluation of this system by both rehabilitation engineers and physical medicine experts. Our future work includes extension of \textsc{RehabRobo-CNl} to enable complex queries over other related ontologies, such as anatomy and disease ontologies. It also includes investigation of methods to partially answer queries, which do not have any answer with respect to the underlying ontology.

ACKNOWLEDGEMENTS

Thanks to anonymous reviewers for useful comments and suggestions. This work has been partially supported by Sabanci University IRP Grant and TUBITAK Grant 111M186.
REFERENCES


