REHABROBO-QUERY: Answering Natural Language Queries about Rehabilitation Robotics Ontology on the Cloud

Zeynep Dogmus and Esra Erdem and Volkan Patoglu
Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul, Turkey
E-mail: {zeynepdogmus,esraerdem,vpatoglu}@sabanciuniv.edu

Abstract We introduce a novel method to answer natural language queries about rehabilitation robotics, over the formal ontology REHABROBO-ONTO. For that, (i) we design and develop a novel controlled natural language for rehabilitation robotics, called REHABROBO-CNL; (ii) we introduce translations of queries in REHABROBO-CNL into SPARQL queries, utilizing a novel concept of query description trees and description logics concepts; (iii) we use an automated reasoner to find answers to SPARQL queries. To facilitate the use of our method by experts, we develop an intelligent, interactive query answering system, called REHABROBO-QUERY, using Semantic Web technologies, and make it available on the cloud via Amazon web services. REHABROBO-QUERY 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 REHABROBO-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.

Keywords: Ontology systems, query answering, rehabilitation robotics, intelligent user-interfaces, controlled natural languages

1. Introduction

We have earlier designed and developed the first formal rehabilitation robotics ontology, called REHABROBO-ONTO, in OWL (Web Ontology Language) [1,19], and made it available on the cloud via Amazon Web Services [8,9,11] (in particular, Amazon Elastic Compute Cloud)[1] so that every rehabilitation robotics researcher can easily add information about his/her robot to it, and modify this information. To facilitate such modifications of the rehabilitation robotics ontology REHABROBO-ONTO, we have also developed a Web-based software (called REHABROBO-QUERY)[2] with an intelligent user-interface. In this way, experts do not need to know the underlying logic-based representation languages of ontologies, like OWL, or Semantic Web technologies, for information entry and modification. REHABROBO-QUERY also utilizes Amazon Web Services for cloud computing. For further information about the design, development and maintenance of REHABROBO-ONTO, and how REHABROBO-QUERY facilitates modification of REHABROBO-ONTO, we refer the reader to our earlier article [11].

This article is concerned about reasoning about rehabilitation robotics over REHABROBO-ONTO. Note that a structured representation of information about rehabilitation robotics as the ontology REHABROBO-ONTO 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 REHABROBO-ONTO by expressing the requested information as a

[1]http://aws.amazon.com/ec2/
[2]http://hmi.sabanciuniv.edu/?page_id=781

1570-0844/0-1900/$35.00

formal query in a query language, such as SPARQL [28], and then by computing answers to these queries by using a state-of-the-art automated reasoner, such as PELLET [29]. 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. For that reason, we particularly focus on the process of query answering over REHABROBO-ONTO, and making it easier for the users to express their queries in a natural language and to obtain answers to their queries automatically.

Towards these goals, the main contributions of this article can be summarized as follows. For expressing queries about rehabilitation robotics, we design and develop a novel controlled natural language for rehabilitation robots, called REHABROBO-CNL. For automatically computing answers to natural language queries, we introduce an algorithm that transforms natural language queries in REHABROBO-CNL into formal queries in SPARQL. This transformation 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, PELLET is used to find relevant answers to the query. The software system REHABROBO-QUERY is extended with an interactive, intelligent user interface to guide the users to enter natural language queries in REHABROBO-CNL about the existing robots, and to present the answers in an understandable form, so that the users do not have to know about the logical formalism of the ontology or the formalism to represent queries, or the use of the technologies for computing answers to their questions about rehabilitation robots.

The ontology system consisting of REHABROBO-ONTO and REHABROBO-QUERY is of great value to robot designers as well as physical therapists and medical doctors. On the one hand, robot designers can benefit from the system, for instance, to identify robotic devices targeting similar therapeutic exercises or to determine systems using a particular kind of actuation-transmission pair to achieve a range of motion that exceeds some threshold. Availability of such information may help inspire new designs or may lead to a better decision making process. The ontology can also be utilized to group similar robots by quantifiable characteristics and to establish benchmarks for system comparisons. Overall, an ontology designed to specifically meet the expectations of the overall rehabilitation robotics effort has the potential to become an indispensable tool that helps in the development, testing, and certification of rehabilitation robots. On the other hand, physical therapists and medical doctors can utilize the ontology to compare rehabilitation robots and to identify the ones that serve best to cover their needs, or to evaluate the effects of various devices for targeted joint exercises on patients with specific disorders. This further paves the way for translational physical medicine (from bench-to-bed and back) and personalized physical medicine.

The rest of the article is organized as follows. We present the controlled natural language REHABROBO-CNL for expressing queries about rehabilitation robotics (Section 3), and discuss the extension of REHABROBO-QUERY to support querying in REHABROBO-CNL via an interactive, intelligent user-interface (Section 4). We present our transformation of a REHABROBO-CNL query into a SPARQL query (Section 5) whose answer can be computed using PELLET (Section 6). After we summarize the related work about software systems that support natural language queries over ontologies (Section 7), we conclude with a summary of our contributions (Section 8).

This article extends our earlier paper [10], presented at the International Conference on Knowledge Engineering and Ontology Development (KEOD 2014), by describing more details about the query answering system integrated in REHABROBO-QUERY, the query language REHABROBO-CNL, and the transformation algorithm.

2. A Brief Review of REHABROBO-ONTO

REHABROBO-ONTO is the first formal rehabilitation robotics ontology that represents knowledge about rehabilitation robotics in a structured form, and allows automated reasoning about this knowledge. It is developed in line with the efforts of European Network on Robotics for Neurorehabilitation[3], for standardizing terminology as well as assessment measures for rehabilitation robots.

REHABROBO-ONTO has been designed by considering suggestions of various rehabilitation robotics researchers and physical medicine experts, by first identifying the purpose, and then the basic concepts and their thematic classes, and their relationships. The main classes and their relationships are illustrated in Figure 1.

[3] http://www.rehabilitationrobotics.eu/

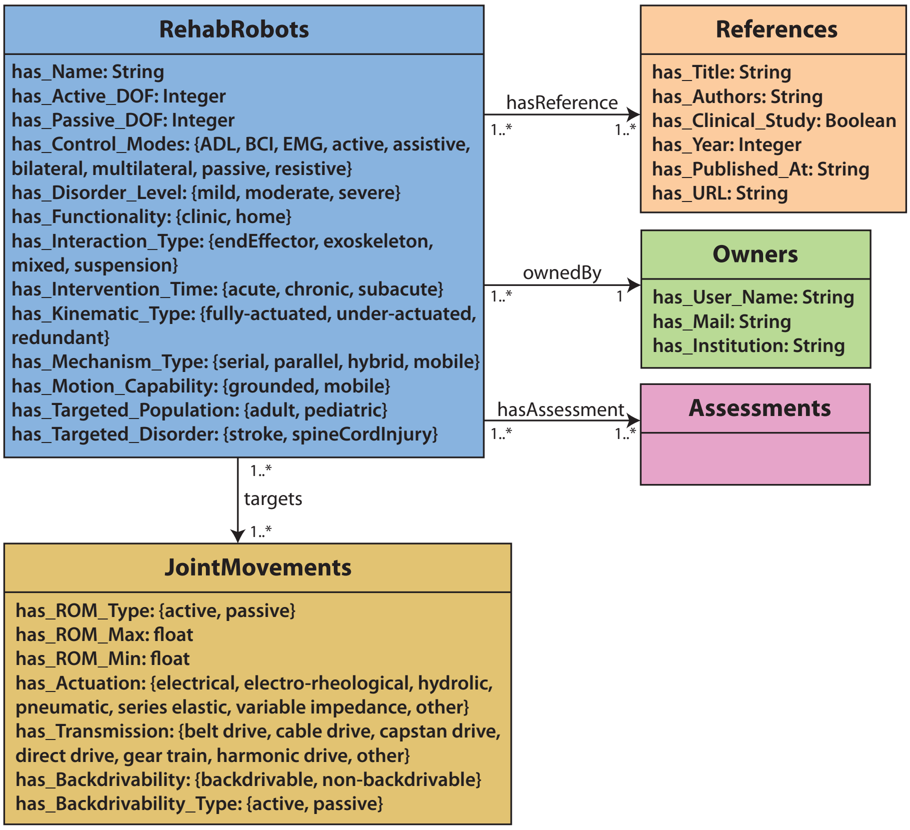


Figure 1. REHABROBO-ONTO with main classes [11, Fig. 1].

REHABROBO-ONTO has five main concepts (or thematic classes):

– `RehabRobots` (representing rehabilitation robots and their properties),
– `JointMovements` (representing targeted joint movements and their properties),
– `Owners` (representing robot designers who add/modify information in the ontology about their own robots),
– `References` (representing publications related to rehabilitation robots),
– `Assessments` (representing assessment measures for rehabilitation robots).

The main classes `RehabRobots`, `JointMovements` and `Assessments` have subclasses. Currently there are 147 classes represented in REHABROBO-ONTO.

The main concepts are related to each other by the following relations:

– a rehabilitation robot `targets` joint movements,
– a rehabilitation robot is `ownedBy` a robot designer,
– a rehabilitation robot `hasReferences` to some publications,
– a rehabilitation robot `hasAssessment` with respect to some evaluation measure.

REHABROBO-ONTO is developed using the ontology editor PROTÉGÉ [18], in the ontology language OWL 2 DL, with OWL/XML syntax. It is open-source and available on the cloud via Amazon Web Services. Its maintenance (i.e., adding, deleting, modifying information about rehabilitation robots) can be done as part of the ontology system REHABROBO-QUERY.

For further information about the design and development of REHABROBO-ONTO, and how REHABROBO-QUERY facilitates maintenance of REHABROBO-ONTO on the cloud, we refer the reader to our earlier article [11].

3. REHABROBO-CNL: A Controlled Natural Language for Queries about Rehabilitation Robotics

Reasoning over REHABROBO-ONTO can be done by means of answering questions posed by the user in a natural language. However, natural languages may have ambiguities in their vocabularies and grammars. To overcome ambiguities of a natural language, a subset of it with a restricted vocabulary and grammar can be considered for specific domains; such languages are called Controlled Natural Languages (CNLs) [22]. Essentially, with its restricted vocabulary and grammar, a CNL is a formal language. Therefore, it is easier to convert a CNL (compared to a more general natural language) 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. Due to these reasons, first we have identified the sorts of queries about rehabilitation robots, then we have designed and developed a new CNL, called REHABROBO-CNL, to express these queries in natural language.

3.1. Sorts of queries supported by REHABROBO-CNL

The variety of queries is important since the users may include not only rehabilitation robot designers and developers, but also physical therapists and medical doctors. Also, complex queries that not only require intelligent extraction of different sorts of knowledge, but their integration via conjunctions, disjunctions, negations, nested relative clauses, aggregates, and quantifications are inevitable in rehabilitation robotics, since robots, targeted movements, evaluation metrics, etc. have many features that are of interest to users. We have designed REHABROBO-CNL considering these aspects.

We have identified different sorts of complex queries useful from the perspectives of rehabilitation robotics and physical medicine. Some examples of these queries are presented in the following, with respect to their topics.

Queries about mechanical properties of rehabilitation robots These are queries to extract information about the rehabilitation robots whose information is available in REHABROBO-ONTO. Here are some examples:

- What are the robots that target shoulder movements and that have at least 210° RoM for the flexion/extension movements of the shoulder?
- What are the robots that target wrist movements and that have at least 2 active degrees of freedom?
- What are the shoulder robots that target flexion/extension movements and that do not target elevation/depression movements?
- What are the robots with active degree of freedom $\geq$ ‘3’ and that target all pelvic girdle movements with backdrivability type = ‘passive’?
- What are the ankle robots that target movements with electrical actuation and with cable drive transmission?

Queries about joint movements targeted by rehabilitation robots These are queries to extract information about joint movements targeted by rehabilitation robots in REHABROBO-ONTO. Here are some examples:

- What are the movements that are targeted by some robots with (some intervention time or with all targeted disorders)?
- What are the movements that are targeted by the robot ‘AssistOn-Finger’ and with minimum range of motion $\geq$ ‘20’?
- What are the movements that are not targeted by any robots with kinematic type = ‘redundant’ and with mechanism type $\neq$ ‘parallel’?

According to REHABROBO-ONTO, for instance, some part of the answer to the first question is as follows:

Index Finger DIPFlexion/Extension
Shoulder Scapular Elevation/Depression
Wrist Flexion/Extension

Queries about the evaluation metrics for rehabilitation robots These are queries to extract information about metrics used to evaluate rehabilitation robots. Here are some examples:

- What are the effort metrics that are evaluated by some robots with active degree of freedom ≥ 2?
- What are the movement quality metrics that are evaluated by all robots with motion capability = ‘grounded’?
- What are the kinematic aspect metrics that are evaluated by some robots that target all elbow movements?
- What are the muscle strength metrics that are evaluated by robots that target all wrist movements with transmission = ‘direct drive’?

- What are the psychomotoric aspect metrics that are evaluated by all robots with kinematic type = 'redundant' and that target all ankle movements with minimum range of motion ≥ '30'?

According to REHABROBO-ONTO, for instance, some part of the answer to the first question is as follows:

Time To Initiate Movement
Amount Of Compensation
Biomechanical Work Energy Power

Some part of the answer to the second question is as follows:

Visuomotor Coordination
Combined Task Coordination
Single Joint Coordination

Other queries These are queries to extract information about publications about rehabilitation robots, owners/institutes/laboratories of the robots. Here are some examples:

- What are the publications with clinical study and that do not reference any robots with active degree of freedom ≥ 1?
- What are the publications without clinical study or that reference some robots that do not evaluate any movement quality metrics?
- What are the publications with place of publication 'ICORR' and that reference some robots that are owned/maintained by some users with institution 'Sabanci University'?
- What are the users that own/maintain some robots that target all ankle movements?

3.2. *Grammar of* REHABROBO-CNL

The grammar of REHABROBO-CNL is shown in Table 1. 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. No further nesting of conjunctions (resp., disjunctions) in a simple disjunction (resp., conjunction) is allowed. A query can contain any number of conjunctions and disjunctions on the condition that they match to the rule above. 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 following query is in DNF:

What are the robots with no targeted disorder or (with active degree of freedom>1 and with control modes='{active,assistive}' and with no disorder level)?

The functions (in italic font) in the grammar 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 such a matching of types with verbs, it is possible to prevent semantically wrong queries like "What are the publications that target some shoulder movements?". All of the matches between types and verbs in expressions of the form "*Type*() that *Verb*()" are shown in Table 3. Note that these matchings essentially come from the structure of REHABROBO-ONTO, e.g., concept names and relation names.

Similarly, it is necessary to match verbs with types in expressions of the form "*Verb*() (some | all | any* | the) *Type*()". Table 4 lists the available types that can occur after a verb in the query (e.g., in a RELATIVECLAUSE), and demonstrates what kinds of types are extracted from the ontology according to the query. If a quantifier such as "some" is used in a relative clause, then the types which have some subclasses are extracted. Here is an example query:

What are the robots that evaluate some wrist movements?

Since wrist movements class have subclasses (e.g., wrist flexion/extension, wrist radial deviation/ulnar deviation) in REHABROBO-ONTO, this query will retrieve all robots that target at least one of these subclasses. If "the" keyword is used after the verb in a query, then the leaf classes are extracted to select one specific type. Here is an example query:

What are the robots that target the wrist radial deviation/ulnar deviation?

Wrist radial deviation/ulnar deviation is a leaf class. It is also a subclass of wrist movements. This query will retrieve the robots that target this specific wrist movement. If there is a robot that targets some other wrist

Table 1
The Grammar of REHABROBO-CNL

QUERY →	WHATQUERY QUESTIONMARK
WHATQUERY →	What are the *Type*() GENERALRELATION
GENERALRELATION →	SIMPLERELATION NESTEDRELATION*
SIMPLERELATION →	(that RELATIVECLAUSE)+
SIMPLERELATION →	that INSTANCERELATION
SIMPLERELATION →	WITHRELATION
NESTEDRELATION →	(and LP SIMPLEDISJUNCTION RP)*
NESTEDRELATION →	(or LP SIMPLECONJUNCTION RP)*
SIMPLEDISJUNCTION →	(SIMPLERELATION or)* SIMPLERELATION
SIMPLECONJUNCTION →	(SIMPLERELATION and)* SIMPLERELATION
RELATIVECLAUSE →	*Verb*() (some | all | the) *Type*()
RELATIVECLAUSE →	NEG *Verb*() any *Type*()
INSTANCERELATION →	NEG? *Verb*() the *Type*() *Instance*()
WITHRELATION →	with *Noun*() EQCHECK *Value*()+
WITHRELATION →	with QUANTIFIER *Noun*()
WITHRELATION →	(with | without) *Noun*()
EQCHECK →	$= \mid != \mid \leq \mid \geq$
QUANTIFIER →	some | all | none
NEG →	*Neg*()
LP →	(
RP →	)
QUESTIONMARK →	?

Table 2
The Ontology Functions

Type()	Returns the types that correspond to concept names. They are: Robots, movements, users, publications and metrics.
Instance()	Returns robot names for robots and user names for users.
Verb()	Returns the verbs that correspond to object properties between concepts. Returns both active and passive forms of these verbs. Active forms of these verbs are: Target, evaluate, reference, own.
Noun()	Returns the nouns that correspond to data properties. ex. targeted disorder, active degree of freedom.
Value()	Returns the suitable values according to a given noun. Corresponds to the pre-defined ranges of data properties.
Neg()	Returns a suitable negation phrase. These phases are: do not, are not.

movements but wrist radial deviation/ulnar deviation, then this robot will not be included in the answer to this query.

In REHABROBO-CNL, the instances of the concepts are represented by one of their distinctive properties. For robots, this distinctive property is its name; for users, it is the user name. To illustrate, when the user wants to query about AssistOn-Shoulder, s/he specifies the instance using the name of the robot. For movements and metrics, there is no such distinctive property

Table 3
Verbs that can occur after the nouns

Type()	that	*Verb*()
robots	→	target
robots	→	are owned/maintained
robots	→	evaluate
movements	→	are targeted by
users	→	own/maintain
publications	→	reference
effort metrics	→	are evaluated by
kinematic aspect metrics	→	are evaluated by
movement quality metrics	→	are evaluated by
muscle strength metrics	→	are evaluated by
psychomotoric aspect metrics	→	are evaluated by

Table 4
Types that can occur after the verbs

Verb()	(some | all | any* | the)	*Type*()
target	(some | all | any)	all movements except leaf classes
target	the	leaf classes of movements
evaluate	(some | all | any)	all metrics except leaf classes
evaluate	the	leaf classes of metrics
are targeted by	(some | all | any)	robots
are evaluated by	(some | all | any)	robots
reference	(some | all | any)	robots
own	(some | all | any)	robots
are owned by	(some | all | any)	users

* *any* is used after negative verbs.

Table 5
Instances that can occur after the types

Type()		*Instance*()
robot	→	name of the robot
user	→	username of the user

because the concept names are sufficient to specify a movement or metric. In fact, using RELATIVECLAUSE is sufficient to query about them. To query about robots or users, however, we use INSTANCERELATION to extract the instances. Table 5 demonstrates the relevant properties of the instances that appear when a type is selected.

In addition to types and verbs, types are matched with the relevant nouns, as shown in Table 6. For instance, control modes are matched with robots whereas actuation is matched with movements. Note that these matchings are due to properties of concepts in REHABROBO-ONTO.

Further, the values for the nouns are extracted from REHABROBO-ONTO to allow suitable entries from the users. These values are listed in Table 7. The values can be considered as ranges of the nouns, that the user can choose from.

4. User-Interface of REHABROBO-QUERY for Query Answering: Intelligent and Interactive

The user interface of REHABROBO-QUERY can guide the users to add/modify information in REHABROBO-ONTO. We have extended it further so that it can guide the users to ask questions in REHABROBO-CNL, and

Table 7
Values that can occur after the nouns

$Noun()$	*	$Value()$
active DoF**	$\rightarrow$	any integer value entered by the user
actuation	$\rightarrow$	electrical, electro-rheological, hydrolic, pneumatic, series elastic, variable impedance, other
authors	$\rightarrow$	one of the authors that are added to the ontology up to now
backdrivability	$\rightarrow$	backdrivable, non-backdrivable
backdrivability type	$\rightarrow$	active, passive
control modes	$\rightarrow$	ADL, BCI, EMG, active, assistive, bilateral, multilateral, passive, resistive
disorder level	$\rightarrow$	mild, moderate, severe
functionality	$\rightarrow$	clinic,home
institution	$\rightarrow$	one of the institutions that are added to the ontology up to now
interaction type	$\rightarrow$	exoskeleton, mixed, suspension, end effector
intervention time	$\rightarrow$	acute, chronic, subacute
kinematic type	$\rightarrow$	hybrid, parallel, serial
motion capability	$\rightarrow$	grounded, mobile
name	$\rightarrow$	one of the robot names that are added to the ontology up to now
passive DoF	$\rightarrow$	any integer value entered by the user
place of publication	$\rightarrow$	one of the places of publication that are added to the ontology up to now
maximum RoM***	$\rightarrow$	any float value entered by the user
minimum RoM	$\rightarrow$	any float value entered by the user
RoM type	$\rightarrow$	active, passive
targeted disorder	$\rightarrow$	stroke, spine cord injury
targeted population	$\rightarrow$	adult, pediatric
title	$\rightarrow$	one of the publication titles that are added to the ontology up to now
transmission	$\rightarrow$	belt drive, cable drive, capstan drive, direct drive, gear train, harmonic drive, other
url	$\rightarrow$	one of the urls that are added to the ontology up to now
username	$\rightarrow$	one of the usernames that are added to the ontology up to now
year	$\rightarrow$	any year (integer value) entered by the user

* (EQCHECK|QUANTIFIER) ** degree of freedom *** range of motion

present answers to these queries in an understandable form.

While the user constructs questions, REHABROBO-QUERY's user interface can prevent nonsensical queries. For instance, shoulder elevation/depression is not a wrist movement. The user interface takes into account such information, and guides the user intelligently to construct his/her queries so that queries like

> What are the wrist robots that target shoulder elevation/depression?

are not possible. In that sense, the user interface is intelligent.

REHABROBO-QUERY's user interface is also interactive: it shows the possible choices (for instance, the sort of movements targeted by a shoulder robot) and allows auto-completion (for instance, to obtain the full name of a robotics device). To be able to identify the choices (e.g., parameters and their values) relevant to the query, REHABROBO-QUERY's user interface utilizes automated reasoning methods online. Indeed, as the user enters his/her query, in the background the user interface considers the part of the query constructed so far and asks REHABROBO-QUERY's ontological reasoning module to compute all possible values of parameters. Once the reasoning module returns these choices, the user interface presents them, e.g.,

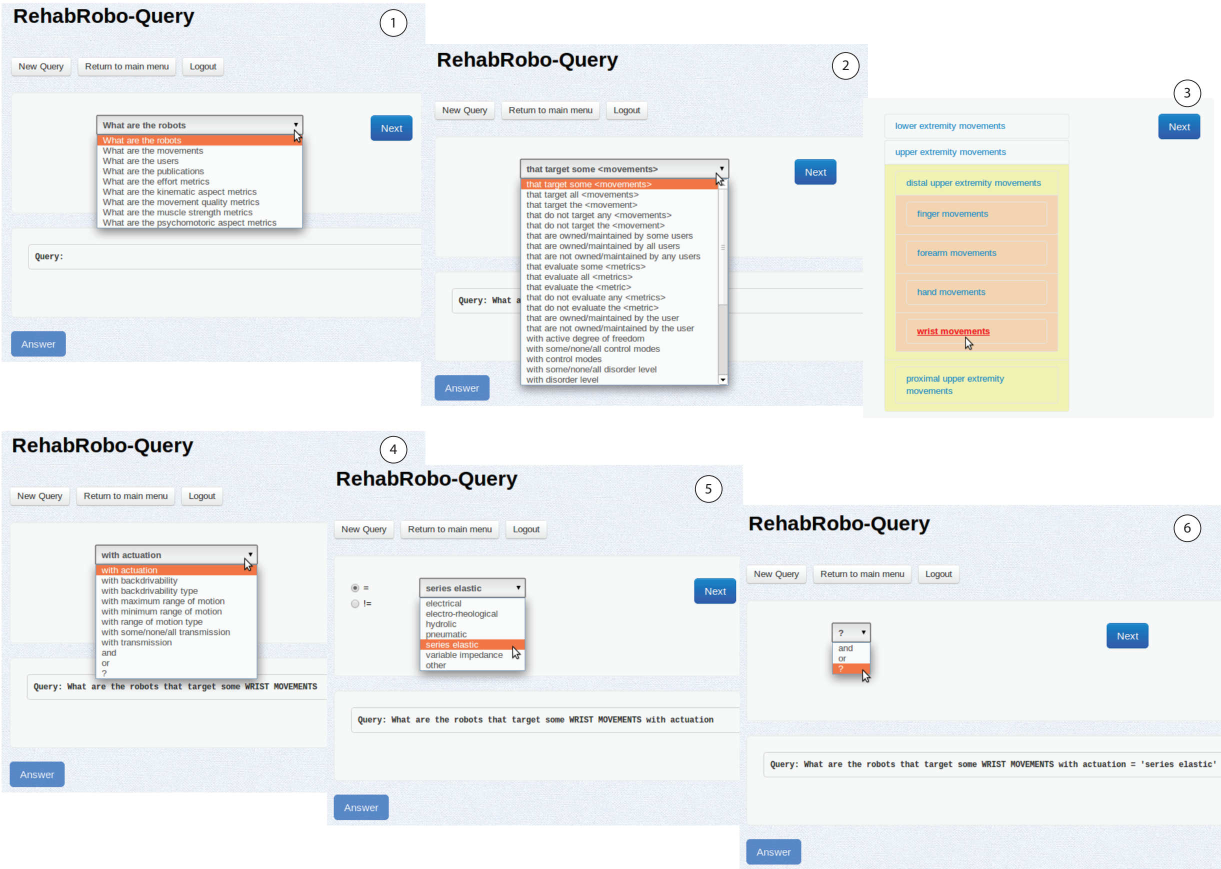


Figure 2. Construction of a query with the guidance of the intelligent interactive user interface of REHABROBO-QUERY.

within a pull-down menu. Figure 2 shows the construction of the following query with the guidance of the user interface:

> What are the robots that target some wrist movements with actuation='series elastic'?

Figure 3. An answer for the query constructed in Figure 2, presented to the user.

Furthermore, REHABROBO-QUERY 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 REHABROBO-QUERY displays an accordion view and enables the user to click on the option s/he wants. In addition, REHABROBO-QUERY 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.

Once the query is constructed, REHABROBO-QUERY asks the reasoning module to compute answers, and presents answers to queries concisely, while also including links to detailed information in case the user wants further information. The answer to the query constructed in Figure 2 is shown to the user as in Figure 3.

Note that how the results of a query is displayed to user depends on the sort of 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

Table 6
Nouns that can occur after the types

$Type()$	with	$Noun()$
robots	→	active degree of freedom
robots	→	control modes
robots	→	disorder level
robots	→	functionality
robots	→	interaction type
robots	→	intervention time
robots	→	kinematic type
robots	→	motion capability
robots	→	name
robots	→	passive degree of freedom
robots	→	targeted disorder
robots	→	targeted population
movements	→	actuation
movements	→	backdrivability
movements	→	backdrivability type
movements	→	maximum range of motion
movements	→	minimum range of motion
movements	→	range of motion type
movements	→	transmission
publications	→	authors
publications	→	clinical study
publications	→	place of publication
publications	→	title
publications	→	url
publications	→	year
users	→	institution
users	→	mail
users	→	username

or metrics, then the user sees the concept names instead of the instance URIs which would make no sense to the user.

5. Transforming a Query in REHABROBO-CNL to a SPARQL Query

The process of answering a query in REHABROBO-CNL, illustrated in Figure 4, starts with a transformation of the query into a SPARQL query. We propose a transformation with the following steps.

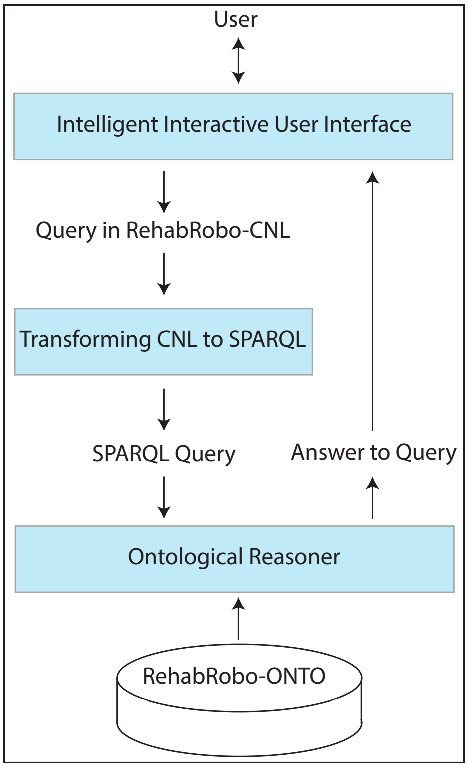


Figure 4. An overview of query construction in REHABROBO–QUERY.

1. We parse the query and form a query description tree.
2. We traverse the tree and obtain a DL concept description.
3. We transform the DL concept into a SPARQL concept.
4. We form a SPARQL query.

5.1. Query Description Trees (QDT)

We introduce a rooted, directed tree, called query description tree (QDT), to parse the REHABROBO-CNL query entered by the user. In this tree, there are 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”.
– and-node: Represents a conjunction.
– or-node: Represents a disjunction.

Every root/that/with-node characterizes a phrase and a type/instance. An and/or-node cannot be a leaf.

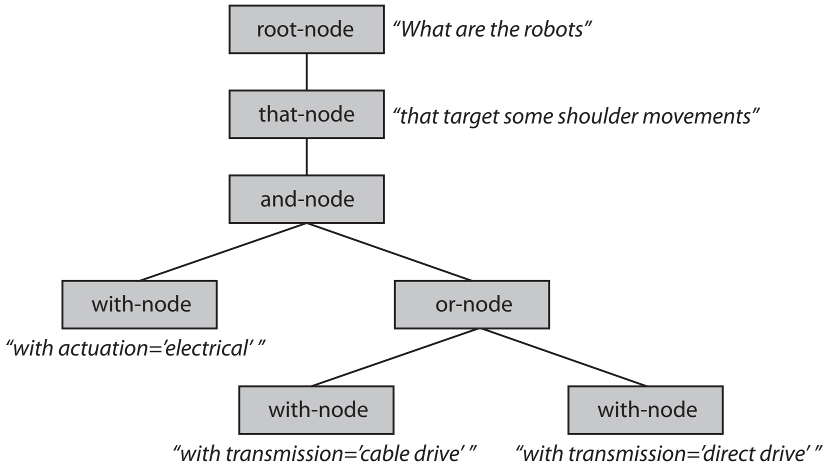


Figure 5. Tree representation of the sample query.

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 5 for the query:

> What are the robots that target some shoulder movements with actuation='electrical' and (with transmission = 'cable drive' or with transmission='direct drive')?

This tree is constructed online while the user expresses the query by the help of the user interface. 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, and 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 relative clauses, including a simple disjunction. Clauses joined with a conjunction (resp., disjunction) are characterized by an and-node (resp., or-node) as their parent. Since these relative clauses start with "with", they are with-nodes. They include values of properties instead of types.

5.2. 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.

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
// $n.class$ denote associated class of a node n
// $n.children$ denote children of a node n
$Q \leftarrow \emptyset$;
$n \leftarrow$ first (root) node in T;
if *n is a root-node* **then**
  $Q \leftarrow Q \sqcap n.class$;
  foreach *child node* $c \in n.children$ **do**
    $Q \leftarrow Q \sqcap transform(c)$;
else if *n is a that-node* **then**
  $Q \leftarrow Q \sqcap transformThatNode(n)$;
else if *n is a with-node* **then**
  $Q \leftarrow Q \sqcap transformWithNode(n)$;
else if *n is an and-node OR n is an or-node* **then**
  $tempQ \leftarrow \emptyset$;
  foreach *child node* $c \in n.children$ **do**
    if *n is an and-node* **then**
      $tempQ \leftarrow tempQ \sqcap transform(c)$;
    else
      $tempQ \leftarrow tempQ \sqcup transform(c)$;
  $Q \leftarrow Q \sqcap (tempQ)$;
return Q

By a depth-first traversal of a QDT, we represent the corresponding concept in Description Logics (DL) as

Algorithm 2: transformThatNode

Input : A that-node n
Output: A DL concept description Q that represents the concept in n
// $n.class$ denote associated class of a node n
// $n.verb$ denote associated verb of a node n
// $n.negative$ denote the negativity of a node n
// $n.quantifier$ denote the quantifier of a node n
// $n.instance$ denote the instance of a node n, if exists
// $n.child$ denote child of a node n
// $n.class.identifierNoun$ denote the noun that identifies $n.class$
$Q \leftarrow \emptyset$;
$childQ \leftarrow \emptyset$;
if $n.child$ *is not empty* **then**
  $childQ \leftarrow transform(n.child)$;
else if n *includes an instance* **then**
  $childQ \leftarrow \exists(n.class.identifierNoun).\{n.instance\}$;
if $n.quantifier = ALL$ **then**
  if $n.verb$ *is passive* **then**
    $Q \leftarrow Q \sqcap \forall(n.verb)^{-}.((n.class) \sqcap childQ)$;
  else
    $Q \leftarrow Q \sqcap \forall(n.verb).((n.class) \sqcap childQ)$;
else
  // If there is no quantifier or the quantifier is SOME
  if $n.verb$ *is passive* **then**
    $Q \leftarrow Q \sqcap \exists(n.verb)^{-}.((n.class) \sqcap childQ)$;
  else
    $Q \leftarrow Q \sqcap \exists(n.verb).((n.class) \sqcap childQ)$;
if $n.negative = True$ **then**
  $Q \leftarrow \neg Q$;
return Q

Algorithm 3: transformWithNode

Input : A with-node n
Output: A DL concept description Q that represents the concept in n
// $n.noun$ denote associated noun of a node n
// $n.values$ denote the list of values of a node n
// $n.quantifier$ denote the quantifier of a node n
// $n.aggregator$ denote the aggregator of a node n
// $n.datatype$ denote datatype of the noun in a node n
$Q \leftarrow \emptyset$;
if *n.noun is functional* **then**
  if $n.datatype = boolean$ **then**
    $Q \leftarrow Q \sqcap \exists(n.noun).\{'n.values'_0\text{^^}xsd:boolean\}$;
  else
    if $n.aggregator = '\geq'$ *or* $n.aggregator = '\leq'$ **then**
      $Q \leftarrow Q \sqcap \exists(n.noun).(n.aggregator)_{n.values_0}$;
    else if $n.aggregator = '='$ **then**
      $Q \leftarrow Q \sqcap \exists(n.noun).\{n.values_0\}$;
    else if $n.aggregator = '!='$ **then**
      foreach *value* $v \in n.values$ **do**
        $Q \leftarrow Q \sqcap \neg\exists(n.noun).\{v\}$;
else
  if $n.quantifier = NO$ **then**
    $Q \leftarrow Q \sqcap \neg\exists(n.noun).xsd:(n.datatype)$;
  else if $n.quantifier = ALL$ **then**
    $Q \leftarrow Q \sqcap \forall(n.noun).xsd:(n.datatype)$;
  else if $n.quantifier = SOME$ **then**
    $Q \leftarrow Q \sqcap \exists(n.noun).xsd:(n.datatype)$;
  else
    if $n.aggregator = '='$ **then**
      foreach *value* $v \in n.values$ **do**
        $Q \leftarrow Q \sqcap \exists(n.noun).\{v\}$;
    else if $n.aggregator = '!='$ **then**
      foreach *value* $v \in n.values$ **do**
        $Q \leftarrow Q \sqcap \neg\exists(n.noun).\{v\}$;
return Q

presented in Algorithm 1. Algorithm 2 demonstrates the transformation for a that-node and Algorithm 3 demonstrates the transformation for a with-node.

As the algorithm traverses a QDT, according to the types of nodes, it generates parts of the DL concept. For instance, let us explain Algorithm 1 by an example. Suppose that the input is the tree in Figure 5. It starts from the root node and enters the first if condition. Since the associated class of the node is "robot", our concept description starts with `Robot`. Then, the algorithm calls $transform$ recursively for each child of the root node. In the first recursive call, since the current node is a `that-node`, the algorithm calls $transformThatNode$ (Algorithm 2) and passes `that-node` as an input. Since it has a child node, the first condition of having children is satisfied. Next, the algorithm calls *transform* for its child node, `and-node`, whose children are conjoined after each transformation. The first child of `and-node` is transformed into an existential restriction by calling *transformWithNode* (Algorithm 3). The second child of `and-node` contains a simple disjunction. The algorithm transforms the information in the children of `or-node` into existential restrictions and disjoins them. Finally, the algorithm finishes recursions and returns to `that-node`, covers the transformation of its children with brackets, and transforms `that-node` into an existential restriction. The resulting DL concept that is returned from the algorithm is as follows.

$$\mathtt{Robot} \sqcap \exists \mathtt{targets}.(\mathtt{ShoulderMovements} \sqcap \\ (\exists \mathtt{actuation}.\{\mathtt{electrical}\} \sqcap \\ (\exists \mathtt{transmission}.\{\mathtt{cabledrive}\} \sqcup \\ \exists \mathtt{transmission}.\{\mathtt{directdrive}\})))$$

5.3. *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 [27] and [14]. We also introduce some novel transformations that are not covered by other work. Some transformation examples are shown in Table 8. The transformations without a citation are novel.

Novel transformations include the transformation of inverse roles. Consider the inverse role example in Table 8. DL representation of this concept corresponds to the following first-order formula:

$$\exists x.targets(x,y) \wedge Robot(x).$$

In this formula, we state 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` operator, 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, evaluation of the DL concept and the SPARQL concept will return the same result.

Consider the complement example in Table 8. 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 corresponds to an existential restriction. However, we do not want such mappings of `x`. Therefore, according to the semantics of `NOT EXISTS` in a filter expression, `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 will 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, evaluation of the DL concept and the SPARQL concept will return the same result.

Finally, consider the universal restriction example in Table 8. 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 `FILTER NOT EXISTS`. Since a universal restriction such as $\forall xA(x)$ corresponds to a negated existential restriction $\neg\exists x\neg A(x)$, each `FILTER NOT EXISTS` operator in the SPARQL query corresponds to a negation.

We first use two triples to represent a publication that references a robot. Then, we refer to that publication with its variable, `x`. We also use these triples to make sure that the query does not return an answer if there is no robot in the ontology. In such a case, even though the remaining `FILTER` expression is satisfied, the set of mappings for `x` will be an empty set because none of the publications in the ontology could reference a robot.

The first `FILTER NOT EXISTS` expression contains another `FILTER NOT EXISTS` expression, which contains a triple that represents the robots `y2` referenced by the previously described publication `x`. There is a triple in the first expression as well, that states `y2` is a robot. Since both expressions contain the same variable, the mappings should agree on the robot `y2`. In addition, the set of mappings for `y2` must be an empty set to satisfy the inner `FILTER NOT EXISTS` expression. The outer `FILTER NOT EXISTS` expression then contains the in-

Table 8
DL to SPARQL Transformation Examples

Constructor	**DL**	SPARQL
Concept [14]	`Robot`	`?x rdf:type ns:RehabRobots.`
Role [27]	`targets`	`?x ns:targets ?y.`

Complement	$\neg\exists$`name.{AssistOn}`	`FILTER NOT EXISTS {`
		`?x ns:has_Name 'AssistOn'.`
		`}`

Inverse Role	$\exists$`targets`$^{-}$`.Robot`	`?x ns:targets ?y.` `?x rdf:type ns:RehabRobots.`
Existential Restriction [27]	$\exists$`targets.ShoulderMovements`	`?x ns:targets ?y.` `?y rdf:type ns:ShoulderMovements.`
hasValue Restriction [27]	$\exists$`name.{AssistOn}`	`?x ns:has_Name 'AssistOn'.`

Universal Restriction	$\forall$`reference.Robot`	`?x rr:reference ?y.`
		`?y rdf:type rr:RehabRobots.`
		`FILTER NOT EXISTS {`
		`FILTER NOT EXISTS {`
		`?x rr:reference ?y2.}`
		`?y2 rdf:type rr:RehabRobots.}`

Intersection [14]	`Robot` $\sqcap$ $\exists$`functionality.{clinic}`	`?x rdf:type ns:RehabRobots.` `?x ns:has_Functionality 'clinic'.`

Union [14]	$\exists$`functionality.{clinic}` $\sqcup$
	$\exists$`motionCapability.{grounded}`
		`{?x ns:has_Functionality 'clinic'.}`
		`UNION`
		`{?x ns:has_Motion_Capability 'grounded'.}`

stances of the robots (mappings for `y2`) that are not referenced by the publication `x`. The set of such instances must be empty to satisfy this expression. Otherwise, the query does not return an answer. Therefore, we represent a universal restriction as a negated existential restriction in SPARQL using its operators.

Let us explain the transformation of a DL concept to a SPARQL concept by explaining it over our DL concept. The following DL concept is the concept description that we obtained from our example query.

$$\texttt{Robot} \sqcap \exists \texttt{targets}.(\texttt{ShoulderMovements} \sqcap (\exists \texttt{actuation}.\{\texttt{electrical}\} \sqcap (\exists \texttt{transmission}.\{\texttt{cabledrive}\} \sqcup \exists \texttt{transmission}.\{\texttt{directdrive}\})))$$

First, we transform the concept `Robot`. For that, we assign a variable for robot, and specify its type, `RehabRobots`:

```
?robot1 rdf:type rr:RehabRobots.
```

Second, we transform the existential restriction

$$\exists \texttt{targets}.(\texttt{ShoulderMovements}).$$

The transformation of an existential restriction results in two triples. The first triple is about the relation, and the second triple is about the type of the second variable in the relation. With the following triple, we say that our robot targets a movement of type `ShoulderMovements`.

```
?robot1 rr:targets ?movement1.
?movement1 rdf:type rr:ShoulderMovements.
```

Then, the concept description contains a hasValue restriction, which is transformed into one triple. The triple specifying that the targeted movement has an electrical actuation is as follows.

```
?movement1 rr:has_Actuation 'electrical'.
```

Finally, we transform the simple disjunction that contains two hasValue restrictions as follows. We combine the triples with UNION, a keyword that SPARQL provides for disjunctions.

```
{?movement1 rr:has_Transmission 'cable drive'.}
UNION
{?movement1 rr:has_Transmission 'direct drive'.}
```

The resulting SPARQL concept:

```
?robot1 rdf:type rr:RehabRobots.
?robot1 rr:targets ?movement1.
?movement1 rdf:type rr:ShoulderMovements.
?movement1 rr:has_Actuation 'electrical'.
{?movement1 rr:has_Transmission 'cable drive'.}
UNION
{?movement1 rr:has_Transmission 'direct drive'.}
```

Then, 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 rdf: <http://www.w3.org/...>
PREFIX rr: <http://www.semanticweb.org/...>

SELECT DISTINCT ?name
WHERE {
  ?robot1 rr:has_Name ?name.
  ?robot1 rdf:type rr:RehabRobots.
  ?robot1 rr:targets ?movement1.
  ?movement1 rdf:type rr:ShoulderMovements.
  ?movement1 rr:has_Actuation 'electrical'.
  {?movement1 rr:has_Transmission
     'cable drive'.}
  UNION
  {?movement1 rr:has_Transmission
     'direct drive'.}
}
```

5.4. Some more examples

For a better understanding of the sequence of transformations above, that takes a REHABROBO-CNL and turns it into a SPARQL query, two more examples are provided in Figures 6 and 7.

6. Answering Queries Using Pellet

We use the DL reasoner PELLET to find answers to queries, through the Jena framework. Consider, for instance, the query

> What are the robots that target some wrist movements with actuation='series elastic'?

whose SPARQL representation is as follows.

```
SELECT DISTINCT ?name
WHERE {
  ?robot1 rr:has_Name ?name.
  ?robot1 rdf:type rr:RehabRobots.
  ?robot1 rr:targets ?movement1.
  ?movement1 rdf:type rr:WristMovements.
  ?movement1 rr:has_Actuation
     'series elastic'.
}
```

After loading REHABROBO-ONTO into PELLET, we present this SPARQL query to PELLET, and get the following answer:

```
( ?name = "AssistOn-Mobile" )
```

Consider, for instance, another query:

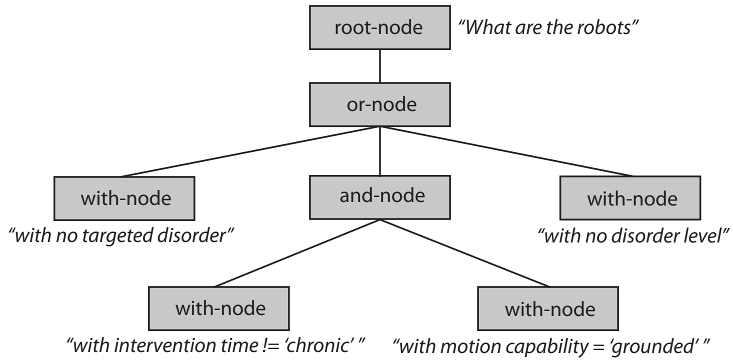
> What are the publications with place of publication 'ICORR' and that reference some robots that are owned/maintained by some users with institution 'Sabanci University' ?

SPARQL representation of this query is as follows.

```
SELECT DISTINCT ?name
WHERE {
  ?publication1 rr:has_Title ?name.
  ?publication1 rdf:type rr:References.
  ?publication1 rr:has_PublishedAt 'ICORR'.
  ?robot1 rr:hasReference ?publication1.
  ?robot1 rdf:type rr:RehabRobots.
  ?robot1 rr:ownedBy ?user1.
  ?user1 rdf:type rr:Owners.
  ?user1 rr:has_Institution
     'Sabanci University'.
}
```

Q11. What are the robots with no targeted disorder or (with intervention time!='chronic' and with motion capability='grounded') or with no disorder level?

Q11 in Query Description Tree:



Q11 in Description Logics (DL):

$\texttt{Robot} \sqcap (\neg\exists\texttt{targetedDisorder}.(\texttt{xsd:string}) \sqcup (\neg\exists\texttt{interventionTime}.\{\texttt{chronic}\} \sqcap \exists\texttt{motionCapability}.\{\texttt{grounded}\}) \sqcup \neg\exists\texttt{disorderLevel}.(\texttt{xsd:string}))$

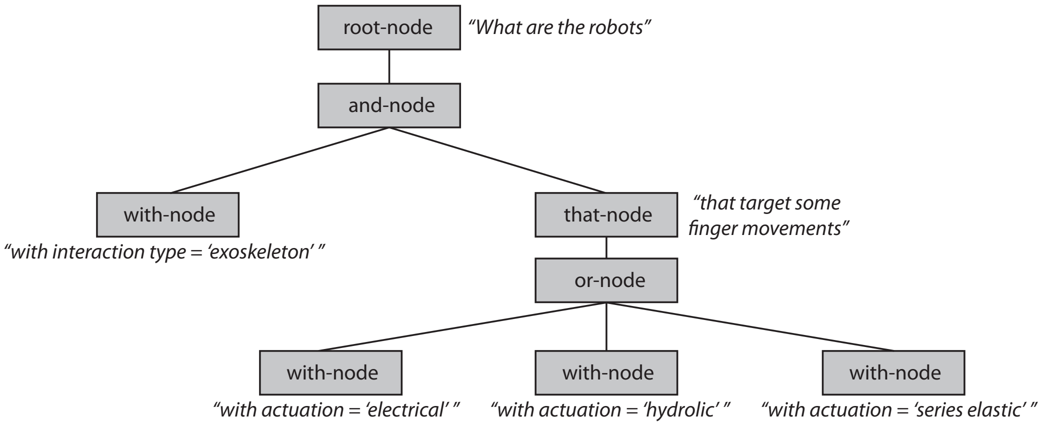
Q11 in SPARQL:

```
SELECT DISTINCT ?name
WHERE {
  ?robot1 rdf:type rr:RehabRobots.
  ?robot1 rr:has_Name ?name.
  {FILTER NOT EXISTS {
    ?robot1 rr:has_Targeted_Disorder ?val1.
  }} UNION
  {FILTER NOT EXISTS {
    ?robot1 rr:has_Intervention_Time 'chronic'.
  }
  ?robot1 rr:has_Motion_Capability 'grounded'.
  } UNION
  {FILTER NOT EXISTS {
    ?robot1 rr:has_Disorder_Level ?val2.
  }}
}
```

Figure 6. Transformation of a query, Q11, from REHABROBO-CNL to a SPARQL.

Q12. What are the robots with interaction type = 'exoskeleton' and that target some finger movements (with actuation = 'electrical' or with actuation = 'hydrolic' or with actuation = 'series elastic') ?

Q12 in Query Description Tree:



Q12 in Description Logics (DL):

$$\mathtt{Robot} \sqcap \exists \mathtt{interactionType}.\{\mathtt{exoskeleton}\} \sqcap$$
$$\exists \mathtt{target}.(\mathtt{FingerMovement} \sqcap (\exists \mathtt{actuation}.\{\mathtt{electrical}\} \sqcup \exists \mathtt{actuation}.\{\mathtt{hydrolic}\} \sqcup \exists \mathtt{actuation}.\{\mathtt{series\ elastic}\}))$$

Q12 in SPARQL:

```
SELECT DISTINCT ?name
WHERE {
  ?robot1 rdf:type rr:RehabRobots.
  ?robot1 rr:has_Name ?name.
  ?robot1 rr:has_Interaction_Type 'exoskeleton'.
  ?robot1 rr:targets ?movement1.
  ?movement1 rdf:type rr:FingerMovements.
  { ?movement1 rr:has_Actuation 'electrical'. } UNION
  { ?movement1 rr:has_Actuation 'hydrolic'. } UNION
  { ?movement1 rr:has_Actuation 'series elastic'. }
}
```

Figure 7. Transformation of a query, Q11, from REHABROBO-CNL to a SPARQL.

We get the following answers from PELLET to this query:

```
( ?name = "Brain Computer Interface
  based Robotic Rehabilitation with
  Online Modification of Task Speed" )
( ?name = "Passive Velocity Field
  Control of a Forearm-Wrist
  Rehabilitation Robot" )
( ?name = "Design of a reconfigurable
  ankle rehabilitation robot and its
  use for the estimation of the ankle
  impedance" )
```

7. Related Work

The most related work involves ontology systems and tools that support natural language queries over ontologies.

Development of natural language interfaces that provide query answering over ontologies has been subject of research for many years. For this reason, many systems [2,3,4,16,20,21,23,25,30,32,33] have been developed that propose various approaches over some common challenges, such as processing of the natural language input (balancing ambiguity and expressiveness) and support for broad or narrow domains (portability).

The most recently developed systems include BIOQUERY-ASP [12], which is a software system that answers natural language queries over biomedical databases and ontologies. It utilizes Answer Set Programming (ASP) [24] to query such knowledge resources. It allows the users to enter queries in a controlled natural language from its user interface, and then answers the queries by transforming the query in a controlled natural language into an ASP program. To enable interoperability over multiple biomedical ontologies and databases, it integrates ontologies via a rule layer in ASP. To answer queries, it utilizes ASP solvers such as CLASP [17] and CLASP-NK, and it also provides explanations to the queries.

Ferrández et al. [15] introduce QACID, which covers a movie ontology. The idea is to train the system using many queries and keep the resulting set of clusters (mostly asked questions) in a database. Then, manually, each query type is associated with a SPARQL query. Finally, the queries are answered by a query engine that is implemented in QACID and proposed as a new entailment-based engine.

FREYA [6] is developed by the creators of and as a development upon QUESTIO [30]. In order to support natural language queries, it uses Stanford Parser [7] to generate a parse tree. Then, using GATE libraries [5], it tries to find some ontology concepts that can be mapped to the query terms. Then, it generates a SPARQL query and executes the query using the inference engine in BIGOWLIM, that supports SPARQL, on the top of Sesame. It relies on clarification dialogues with users in the cases of ambiguity or in the cases where the system cannot find an answer to a query. Over time, the system learns to ask the correct questions to the users by placing correct suggestions on top of similar queries. The system is tested on one dataset, and it is stated that FREYA failed to answer some questions (e.g., queries including negation) correctly. These questions could not be mapped to a SPARQL query in spite of clarification dialogues and learning mechanism.

Lopez et al. [26] introduce PowerAqua, which is evolved from AquaLog [25]. It provides natural language querying over multiple ontologies; thus, supports high scalability and portability. It uses GATE libraries and WordNet [13] to process natural language queries. It transforms the queries to triples and answers them with its own query engine. To limit the search space, it uses filtering and ranking heuristics. Since it does not contain any linguistic knowledge in the background, it has a limited linguistic coverage. It is good at answering simple questions yet it fails on questions that contain comparisons and quantifiers.

Valencia-García et al. [31] introduce OWLPath, which gets user queries in a controlled natural language, transforms it into a SPARQL query and executes the query over an ontology via Jena framework and using the DL reasoner PELLET. The statements in its CNL start with "View any..." and follow English grammar. However, they are not full and valid English sentences. Although it is stated that OWLPath provides a Web interface through AJAX, it is not available online. For each condition in the query, OWLPath adds a FILTER statement in the SPARQL query. Therefore, the transformation of the query into SPARQL is not, in fact, a transformation to triples but a set of FILTER statements. Evaluations are done on ontologies in OWL DL.

8. Conclusion

In this article, we have introduced methods for representing queries about rehabilitation robotics in a controlled natural language, transforming them into formal queries, and computing answers to them using automated reasoners over the first formal rehabilitation robotics ontology REHABROBO-ONTO. We have also introduced an interactive intelligent user-interface as part of the software system REHABROBO-QUERY, that guides the users during this whole process in such a way that the users do not have to know about the underlying logical formalism of the ontology or the formalism to represent queries;, and they do not have to know about the use of the technologies for computing answers to their questions.

By means of answering sophisticated queries over REHABROBO-ONTO, right rehabilitation robots for a particular patient or a physical therapy can be found or designed; this further paves the way for translational physical medicine (from bench-to-bed and back) and personalized physical medicine. Also, REHABROBO-QUERY aids exchange of information across rehabilitation robots researchers over the world, and thus to improve the state-of-the-art; it allows to identify "gaps" in functionality of rehabilitation robots, that can further improve research efforts.

Having a structured formal representation of knowledge about rehabilitation robotics as an ontology, allows answering complex queries that requires integration with other knowledge resources (e.g., patient databases, disease ontologies). Along this research direction, integration of REHABROBO-ONTO with existing anatomy, disease and patient ontologies can be achieved by providing a rule layer between these ontologies and REHABROBO-ONTO, for integration of the related concepts. In addition, some extensions in the grammar of REHABROBO-CNL, the algorithms and the user interface of REHABROBO-QUERY are needed to be able to answer complex queries about therapies, diseases and anatomy. These studies concerning interoperability of REHABROBO-ONTO is part of our ongoing work.

Acknowledgment

We would like to thank Muhammed Kilinc and Sibel A. Yildirim (Hacettepe University, School of Physical Therapy and Rehabilitation) for their feedbacks on the sorts of queries from the perspectives of physical medicine, and Agis Papantoniou (Cognizone) for his help with the design of REHABROBO-QUERY. We would like to thank Ahmetcan Erdogan (Rehabilitation Institute of Chicago), Mustafa Yalcin, Murat Isik and Ata Otaran (Human-Machine Interaction Lab, Sabanci University), and Ezgi Demirel (Knowledge Represenattion and Reasoning Group, Sabanci University) for their helps with testing REHABROBO-QUERY. We also would like to thank members of European Network on Robotics for NeuroRehabilitation for useful discussions.

References

[1] G. Antoniou and F. V. Harmelen. Web ontology language: Owl. In *Handbook on Ontologies in Information Systems*, pages 67–92. Springer, 2003.

[2] A. D. L. Battista, N. Villanueva-Rosales, M. Palenychka, and M. Dumontier. SMART: A web-based, ontology-driven, semantic web query answering application. In *Semantic Web Challenge*, volume 295, 2007.

[3] A. Bernstein and E. Kaufmann. GINO - a guided input natural language ontology editor. In *Proceedings of the 5th international conference on The Semantic Web*, pages 144–157, 2006.

[4] P. Cimiano, P. Haase, J. Heizmann, M. Mantel, and R. Studer. Towards portable natural language interfaces to knowledge bases - the case of the ORAKEL system. *Data Knowl. Eng.*, 65(2):325–354, 2008.

[5] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters. *Text Processing with GATE (Version 6)*. 2011.

[6] D. Damljanovic, M. Agatonovic, and H. Cunningham. FREyA: an interactive way of querying linked data using natural language. In *Proceedings of the 8th international conference on The Semantic Web*, ESWC'11, pages 125–138, 2012.

[7] M.-C. de Marneffe, B. MacCartney, and C. D. Manning. Generating typed dependency parses from phrase structure trees. In *LREC*, 2006.

[8] Z. Dogmus, G. Gezici, V. Patoglu, and E. Erdem. Developing and maintaining an ontology for rehabilitation robotics. In *Proc. of KEOD*, pages 389–395, 2012.

[9] Z. Dogmus, A. Papantoniou, M. Kilinc, S. A. Yildirim, E. Erdem, and V. Patoglu. Rehabilitation robotics ontology on the cloud. In *Proc. of ICORR*, 2013.

[10] Z. Dogmus, V. Patoglu, and E. Erdem. Answering natural language queries about rehabilitation robotics ontology on the cloud. In *Proc. of KEOD*, pages 75–83, 2014. .

[11] Z. Dogmus, E. Erdem, and V. Patoglu. Rehabrobo-onto: Design, development and maintenance of a rehabilitation robotics ontology on the cloud. *Robotics and Computer-Integrated Manufacturing*, 33:100–109, 2015.

[12] E. Erdem, H. Erdogan, and U. Oztok. BIOQUERY-ASP: Querying biomedical ontologies using answer set programming. In *Proc. of RuleML2011@BRF Challenge*, 2011.

[13] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Language, Speech and Communication. Mit Press, 1998.

[14] D. Y. S. Fernandes. *Using Semantics to Enhance Query Reformulation in Dynamic Distributed Environments*. PhD thesis, Federal University of Pernambuco, 2009.

[15] O. Ferrández, R. Izquierdo, S. Ferrández, and J. L. Vicedo. Addressing ontology-based question answering with collections of user queries. *Information Processing and Management*, 45(2):175 – 188, 2009.

[16] A. Frank, H.-U. Krieger, F. Xu, H. Uszkoreit, B. Crysmann, B. Jörg, and U. Schäfer. Question answering from structured knowledge sources. *Journal of Applied Logic*, 5(1):20 – 48, 2007.

[17] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. clasp: A conflict-driven answer set solver. In *Logic Programming and Nonmonotonic Reasoning*, pages 260–265. Springer, 2007.

[18] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu. The evolution of Protégé: an environment for knowledge-based systems development. *Int. J. Hum.-Comput. Stud.*, 58(1):89–123, 2003.

[19] I. Horrocks, P. F. Patel-Schneider, and F. V. Harmelen. From shiq and rdf to owl: The making of a web ontology language. *Journal of Web Semantics*, 1:2003, 2003.

[20] E. Kaufmann, A. Bernstein, and R. Zumstein. Querix: A natural language interface to query ontologies based on clarification dialogs. In *5th ISWC*, pages 980–981. Springer, 2006.

[21] E. Kaufmann, A. Bernstein, and L. Fischer. NLP-Reduce: A naive but domain-independent natural language interface for querying ontologies. In *4th European Semantic Web Conference*, 2007.

[22] T. Kuhn. A survey and classification of controlled natural languages. *Comput. Linguist.*, 40(1):121–170, 2014. .

[23] Y. Lei, V. Uren, and E. Motta. SemSearch: A search engine for the semantic web. In *Proc. 5th International Conference on Knowledge Engineering and Knowledge Management Managing Knowledge in a World of Networks, Lect. Notes in Comp. Sci.*, pages 238–245. Springer-Verlag, 2006.

[24] V. Lifschitz. What is answer set programming?. In *AAAI*, volume 8, pages 1594–1597, 2008.

[25] V. Lopez, V. Uren, E. Motta, and M. Pasin. AquaLog: An ontology-driven question answering system for organizational semantic intranets. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):72–105, 2007.

[26] V. Lopez, M. Fernández, E. Motta, and N. Stieler. PowerAqua: Supporting users in querying and exploring the semantic web. *Semantic Web*, 3(3): 249–265, 2012.

[27] G. Orsi. *Context Based Querying of Dynamic and Heterogeneous Information Sources*. PhD thesis, Politecnico di Milano, 2011.

[28] E. Prud'Hommeaux, A. Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[29] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51 – 53, 2007.

[30] V. Tablan, D. Damljanovic, and K. Bontcheva. A natural language query interface to structured information. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, ESWC'08, pages 361–375, 2008.

[31] R. Valencia-García, F. García-Sánchez, D. Castellanos-Nieves, and J. Fernández-Breis. OWLPath: An OWL ontology-guided query editor. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41 (1):121–136, 2011.

[32] C. Wang, M. Xiong, Q. Zhou, and Y. Yu. PANTO: A portable natural language interface to ontologies. In *Proceedings of the 4th European conference on The Semantic Web: Research and Applications*, ESWC '07, pages 473–487, 2007.

[33] Q. Zhou, C. Wang, M. Xiong, H. Wang, and Y. Yu. SPARK: adapting keyword query to semantic search. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 694–707, 2007.