Semantic Web 0 (0) 1
IOS Press

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

Neurological injuries, such as stroke, are the leading cause of of permanent disability in developed countries [1]. In particular, among 15 million people that suffer from stoke, 5 million are left permanently disabled each year. These disabilities place a high burden on individual welfare of the patients, while negatively impacting national economies [2]. Despite the recent medical developments, the number of stroke incidents continues to increase due to the ageing population in many developed countries.

Physical rehabilitation therapy is indispensable for treating neurological disabilities. Therapies have been shown to be more effective when they are task specific [5], repetitive [7], intense [28], long term [42], and allow for active involvement of patients [34]. However, repetitive and high intensity therapies place high physical burden on the therapist, significantly increasing the cost of such treatments.

With recent advancements at electro-mechanical systems, robot-assisted rehabilitation devices have become ubiquitous, since these devices can bear the physical burden of rehabilitation exercises, while therapists are employed as decision makers. In particular, the use of robotic devices in repetitive and physically involved rehabilitation eliminate the physical burden of movement therapy for the therapists, enable safe and versatile training with increased intensity, increase the reliability, accuracy, and effectiveness of traditional physical rehabilitation therapies. Robot assisted rehabilitation devices can be applied to patients with all levels of impairment, can quantitatively measure patient progress, allow for easy tuning of duration and intensity of therapies, and enable realization of customized, interactive, innovative treatment protocols. Clinical trials with robot assisted rehabilitation indicate that this form of therapy is effective for motor recovery and possesses high potential for improving functional independence of patients [12,29,35,38,39].

As more and more rehabilitation robots are deployed, the information about them also increases, but commonly in unstructured forms, such as text in relevant publications. Consequently, accessing the requested knowledge and thus automatically reasoning about it become a challenge. For instance, accessing the flexion/extension range of motion (RoM) of ASSISTON-SE [46], and determining the rehabilitation robots that target shoulder movements and also have at least 210° RoM for the flexion/extension movements of the shoulder are challenging tasks that require one to go through and study unstructured text from several different resources.

Furthermore, given the interdisciplinary nature of rehabilitation robotics, in many cases, the requested knowledge requires integration of further knowledge from related disciplines, such as physical medicine. For instance, determination of rehabilitation robots that can treat patients with rotator cuff lesions, require one to know that rotator cuffs are muscle units employed to move the shoulder, and that, for patients with rotator cuff lesions, abduction and flexion movements of the shoulder should not have more than 90° RoM. Relevant rehabilitation robots can only be found after one acquires these crucial information.

Additionally, given the growing number of research groups contributing to the field, different approaches display large variability and the field lacks a standardized terminology. Several efforts have been initiated for standardizing terminology, as well as assessment measures, for rehabilitation robots, e.g., by European Network on Robotics for Neurorehabilitation[1]. The development of such a standardization is likely be critical step for the field, helping robotic rehabilitation technology become widely understood and accepted as a useful tool.

Motivated by these challenges and efforts, we have earlier designed and developed the first formal rehabilitation robotics ontology, called REHABROBO-ONTO, in OWL (Web Ontology Language) [3,24], and made it available on the cloud via Amazon Web Services [13,14,16] (in particular, Amazon Elastic Compute Cloud)[2] so that every rehabilitation robotics researcher can easily use it in order to publish/represent information about his/her robot, 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)[3] 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 [16].

This article is concerned with 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 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 formal query in a query language, such as SPARQL [40], and then by computing answers to these queries by using a state-of-the-art automated reasoner, such as PELLET [41]. 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 lan-

[1] http://www.cost.eu/COST_Actions/bmbs/TD1006
[2] http://aws.amazon.com/ec2/
[3] http://hmi.sabanciuniv.edu/?page_id=781

guage 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 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). We evaluate the underlying methods empirically over a variety of queries, and the user-interface by a survey analysis with participants of different backgrounds (Section 7). After we summarize the related work about software systems that support natural language queries over ontologies (Section 8), we conclude with a summary of our contributions (Section 9).

This article significantly extends our earlier paper [15], presented at the International Conference on Knowledge Engineering and Ontology Development (KEOD 2014), by providing more details about the query answering system integrated in REHABROBO-QUERY, the sorts of queries supported by REHABROBO-QUERY, the complete description of the query language REHABROBO-CNL, and the transformation algorithms and rules, and by providing evaluations of the underlying methods and the interactive user-interface of REHABROBO-QUERY.

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[4], 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.

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 [23], 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 [16].

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

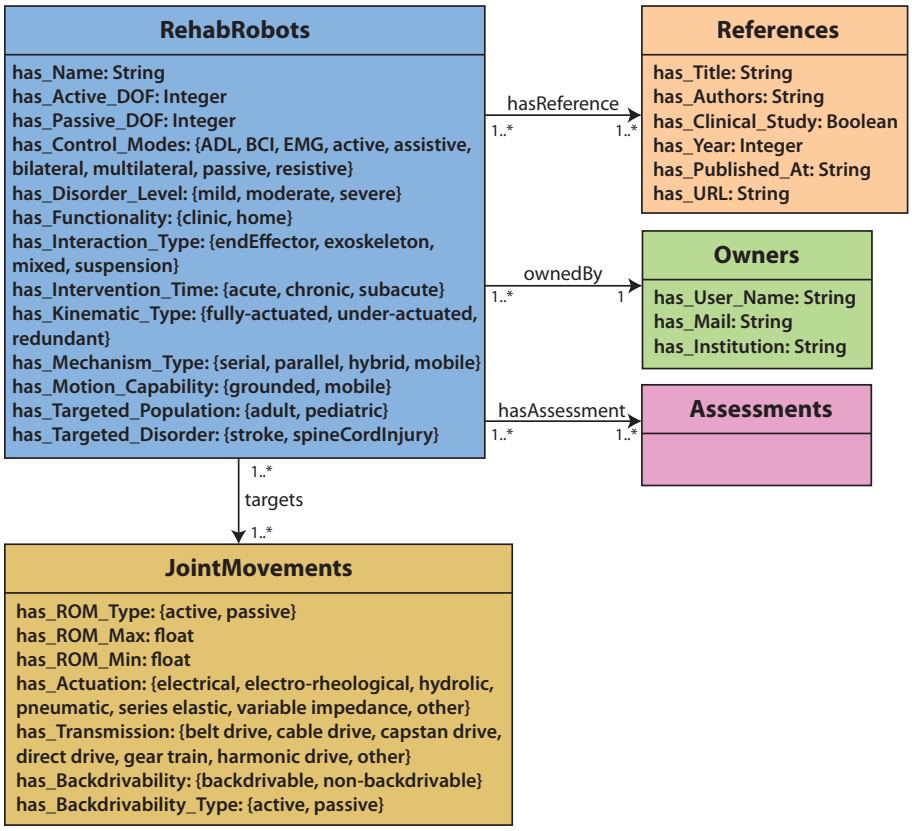


Figure 1. REHABROBO-ONTO with main classes [16, Fig. 1].

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) [27]. 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 $\geq$ '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

Once we identify different sorts of complex queries about rehabilitation robotics, we design the controlled natural language REHABROBO-CNL for expressing them in an unambiguous way and utilizing domain-specific vocabulary. The grammar of REHABROBO-CNL, with relevant vocabulary, is shown in Tables 1–7.

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.

Table 1 gives an overall summary of REHABROBO-CNL, providing information about what sorts of queries are allowed in REHABROBO-CNL. 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)?

Now we have an overall understanding of the sorts of queries supported by REHABROBO-CNL, let us provide some more details to relate it to rehabilitation robotics. The functions presented in italic font in Table 1 refine these queries by embedding relevant information from REHABROBO-ONTO. These ontology functions are described in Table 2.

The information represented 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), relative to the ontology. If a quantifier such as "some" is used in a relative clause, then the types which have some subclasses are considered in queries. 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 considered 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 movements but wrist radial deviation/ulnar deviation, then this robot will not be included in the answer to this query.

Table 5
Instances that can occur after the types

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

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 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-

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.

ONTO. We have extended it further so that it can guide the users to ask questions in REHABROBO-CNL, and 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., 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'?

Table 7
Values that can occur after the nouns

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

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

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

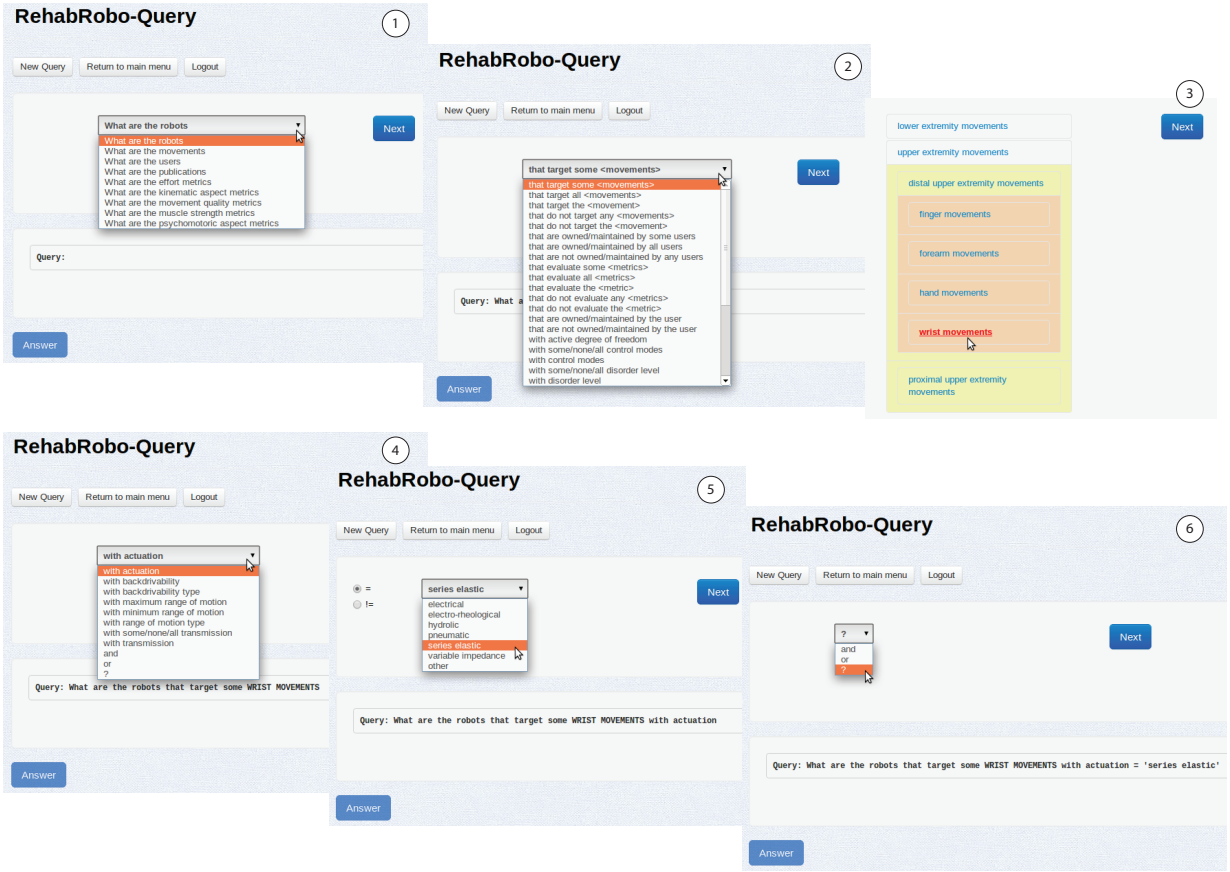


Figure 2. Construction of a query with the guidance of the intelligent interactive user interface of REHABROBO-QUERY.

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 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.

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.

The first step of this transformation relies on REHABROBO-CNL, and thus it is domain-specific. The rest of the transformation, from query description trees to SPARQL queries, is domain-independent.

Note that the second and the third steps of the transformations can be combined into a single step, as implemented in REHABROBO-QUERY. However, for the purpose of more concise descriptions of the algorithms and better understandability, we present the transformation from trees to SPARQL queries in two steps.

5.1. *Query Description Trees (QDT)*

We introduce a rooted, directed tree, called query description tree (QDT), to parse the REHABROBO-

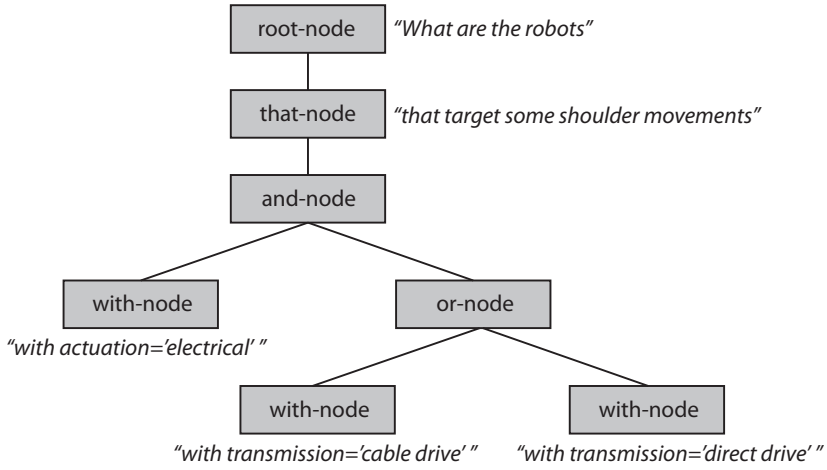


Figure 5. Tree representation of the sample query.

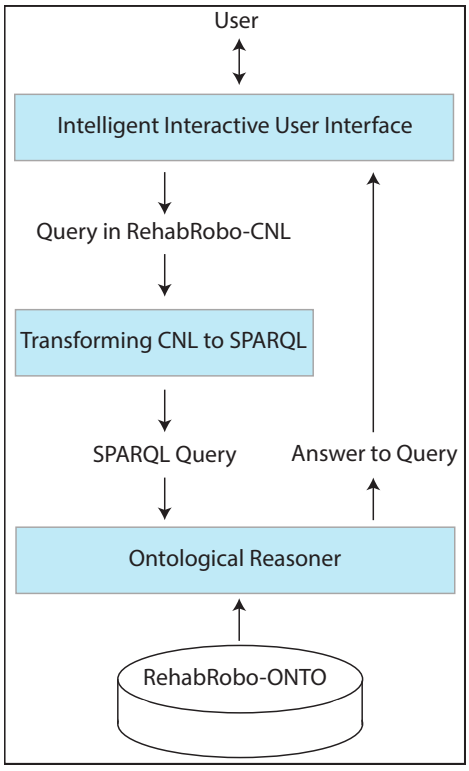


Figure 4. An overview of query construction in REHABROBO–QUERY.

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. 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.

Every QDT is constructed online by a single pass over the input REHABROBO-CNL query guided by an interactive user-interface relative to the REHABROBO-CNL grammar. Therefore, the construction of a QDT from a REHABROBO-CNL query takes linear time in the size of the query.

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".

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

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 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

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

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.

$$\begin{aligned}&\texttt{Robot} \sqcap \exists \texttt{targets}.(\texttt{ShoulderMovements} \sqcap \\ &\quad (\exists \texttt{actuation}.\{\texttt{electrical}\} \sqcap \\ &\quad (\exists \texttt{transmission}.\{\texttt{cabledrive}\} \sqcup \\ &\quad \exists \texttt{transmission}.\{\texttt{directdrive}\})))\end{aligned} \tag{1}$$

Since the transformation is done by a depth-first traversal of a QDT and thus every node is visited once, it takes linear time in the size of the QDT (i.e, number of nodes).

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 translation rules in related publications, such as [36] and [19], and introduce some novel translation rules that are not covered by other work. These transformations are depicted in Table 8; transformations without a citation are novel. Some transformation examples are shown in Table 9.

Novel transformations include the translation rules for complement, inverse roles, and universal restriction. Let us describe them by examples.

Consider the inverse role example $\exists \texttt{targets}^{-}.\texttt{Robot}$ in Table 9. DL representation of this concept corresponds to the first-order formula $\exists x.targets(x,y) \wedge Robot(x)$. This formula 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` satisfies 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 9. 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 9. 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 $\forall x A(x)$ 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. The inner `FILTER NOT EXISTS` expression is satisfied if the robot `y2` is not referenced by the publication `x`. The outer `FILTER NOT EXISTS` expression contains the instances of the robots (mappings for `y2`) that are not referenced by the publication `x`. Thus, the outer expression is satisfied if the set of such instances is empty; otherwise, the query does not return an answer. Note that the first two triples before nested `FILTER NOT EXISTS` expressions represent a publication that references a robot. 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.

In REHABROBO-QUERY, we apply the translation rules to a DL concept as we build it from a QDT in Algorithm 1. Consider, for instance, the DL concept (1) obtained from our example query (Figure 5).

First, as the concept `Robot` is constructed from the root-node of the QDT, we translate it to the SPARQL concept with respect to the relevant translation rule in Table 8 by assigning a variable for robot and specifying its type `RehabRobots`:

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

Then, as the concept $\exists \texttt{targets}.(\texttt{ShoulderMovements})$ is being constructed from that-node of the QDT, two

Table 8
DL to SPARQL Transformation Rules

Constructor	**DL**	SPARQL
Concept [19]	C	?x rdf:type C
Role [36]	R	?x R ?y
Complement	¬R	FILTER NOT EXISTS { ?x R ?y }
Inverse Role	R^{-}.C	?x R ?y . ?x rdf:type C
Existential Restriction [36]	∃R.C	?x R ?y . ?y rdf:type C
hasValue Restriction [36]	∃R.{a}	?x R a
Universal Restriction	∀R.C	?x R ?y. ?y rdf:type C. FILTER NOT EXISTS { ?z rdf:type C. FILTER NOT EXISTS {?x R ?z.}}
Intersection [19]	C⊓ D	?x rdf:type C . ?x rdf:type D
Union [19]	C⊔D	{?x rdf:type C} UNION {?x rdf:type D}

Table 9
DL to SPARQL Transformation Examples

Constructor	**DL**	SPARQL
Concept [19]	Robot	?x rdf:type ns:RehabRobots.
Role [36]	targets	?x ns:targets ?y.
Complement	¬∃name.{AssistOn}	FILTER NOT EXISTS { ?x ns:has_Name 'AssistOn'. }
Inverse Role	∃$targets^{-}$.Robot	?x ns:targets ?y. ?x rdf:type ns:RehabRobots.
Existential Restriction [36]	∃targets.ShoulderMovements	?x ns:targets ?y. ?y rdf:type ns:ShoulderMovements.
hasValue Restriction [36]	∃name.{AssistOn}	?x ns:has_Name 'AssistOn'.
Universal Restriction	∀reference.Robot	?x rr:reference ?y. ?y rdf:type rr:RehabRobots. FILTER NOT EXISTS { ?y2 rdf:type rr:RehabRobots. FILTER NOT EXISTS { ?x rr:reference ?y2.}}
Intersection [19]	Robot⊓ ∃functionality.{clinic}	?x rdf:type ns:RehabRobots. ?x ns:has_Functionality 'clinic'.
Union [19]	∃functionality.{clinic}⊔ ∃motionCapability.{grounded}	{?x ns:has_Functionality 'clinic'.} UNION {?x ns:has_Motion_Capability 'grounded'.}

triples are generated with respect to the existential restriction translation rule in Table 8: The first triple is about the relation, and the second triple is about the type of the second variable in the relation.

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

Then, as the concept ∃actuation.{electrical} is being constructed from the with-node of the QDT, the following triple is generated in SPARQL with respect to the hasValue restriction translation rule in Table 8:

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

Similarly, the concepts ∃transmission.{cabledrive} and ∃transmission.{directdrive} are translated

into two triples. Afterwards, as these two hasValue restrictions are combined into a simple disjunction at the or-node of the QDT, the corresponding triples are combined with UNION, a keyword that SPARQL provides for disjunctions, according to the translation rules for disjunction.

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

Finally, as the DL concept (1) is constructed by a simple conjunction, the following SPARQL concept is built by applying the translation rules for conjunction:

```
?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'.}
```

After a SPARQL concept is obtained, a SPARQL query is constructed 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:

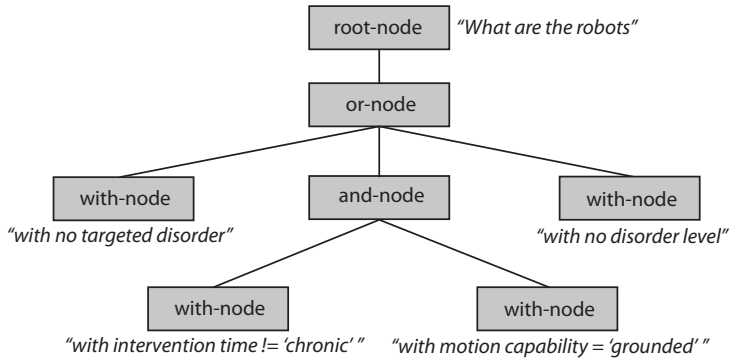
> What are the publications with place of publication 'ICORR' and that reference some robots that are owned/maintained by some users with institution 'Sabanci University' ?

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'.
}
```

Q19. What are the robots with no targeted disorder or (with intervention time!='chronic' and with motion capability='grounded') or with no disorder level?

Q19 in Query Description Tree:



Q19 in Description Logics (DL):

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

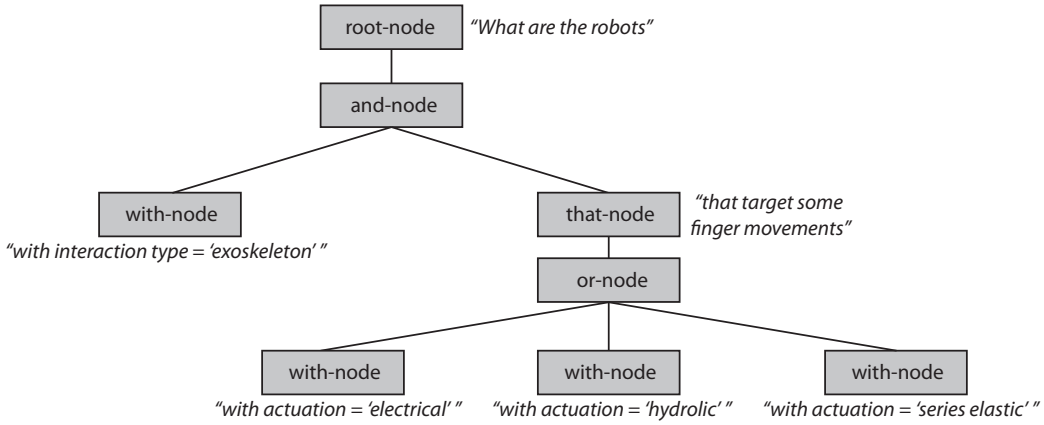
Q19 in SPARQL:

```
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, Q19, from REHABROBO-CNL to a SPARQL.

Q20. What are the robots with interaction type = 'exoskeleton' and that target some finger movements (with actuation = 'electrical' or with actuation = 'hydraulic' or with actuation = 'series elastic') ?

Q20 in Query Description Tree:



Q20 in Description Logics (DL):

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

Q20 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 'hydraulic'. } UNION
  { ?movement1 rr:has_Actuation 'series elastic'. }
}
```

Figure 7. Transformation of a query, Q19, 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. Evaluations of REHABROBO-QUERY

We have evaluated the underlying methods of the system REHABROBO-QUERY empirically over a variety of queries, and its user-interface by means of a survey with the help of participants of different expertise.

7.1. Experimental evaluations of methods

We have experimentally evaluated the computational performance of REHABROBO-CNL to SPARQL transformation and query answering. Along these lines, we have identified 20 different queries as listed in Table 10. All computations are performed using a T1 Micro Instance of Amazon EC2. A T1 Micro Instance operates with up to 600 MB memory and two EC2 compute units, each unit providing an equivalent CPU capacity of a 1.0–1.2 GHz 2007 Xeon processor. For query answering, we use PELLET version 2.3.0.

Experimental results in Table 10 indicate that the transformation from REHABROBO-CNL to SPARQL can be completed within milliseconds, while query answering can take up to 2 seconds. Please recall that REHABROBO-CNL to SPARQL transformation is linear time in the size of the query and thus note that it is independent of the population level of REHABROBO-ONTO while SPARQL query answering is a harder problem [37] and it is likely to be adversely affected as REHABROBO-ONTO is further populated. However, REHABROBO-QUERY has been implemented in Amazon EC2 to ensure scalability of the system even after high population of REHABROBO-ONTO. In particular, larger EC2 instances with more memory and CPU units can be recruited as further resources are necessitated by a larger ontology size.

7.2. User evaluations

We have evaluated the REHABROBO-CNL and the query interface of REHABROBO-QUERY with real users to demonstrate how the proposed approach fits the needs of the people working in the interdisciplinary domain of rehabilitation robotics. Along these lines, we have conducted a survey with 15 experts working in this domain. These experts include clinicians from the Rehabilitation Institute of Chicago and Acıbadem Hospital at İstanbul, rehabilitation robotics engineers from Northwestern University and Sabancı University, and experts from an SME specializing in commercialization of rehabilitation robots.

Each of participant have been given a brief overview of REHABROBO-QUERY and direct access to the interface. Participants have been asked to explore the system by forming several queries about mechanical aspects of robots, joint movements, evaluation metrics, and relevant publications. After exploring REHABROBO-QUERY with at least 20 different queries, participants have been asked to fill in a questionnaire. The questions that have been presented to the participants and the analysis of responses to this questionnaire are presented in Table 11.

The statistical analysis of responses revealed that the factor of occupation was not statistically significant at the 0.05 level for any of the survey questions; hence, all responses are aggregated for reporting. The Cronbach's α values have been calculated for the whole survey, and the α value is evaluated to be greater than 0.7, indicating high reliability of the survey.

The survey includes 3 questions: q1 is aimed at evaluating the usefulness of several query types, q2 is for assessing its potential use, and q3 is for determination of target population of REHABROBO-QUERY. For all questions, the five-point Likert scale, ranging from "1" *strongly disagree* to "5" *strongly agree* is used to measure agreement level of the participants.

The main results of the survey can be summarized as follows:

- Responses to q1 indicate that participants *agree* with the usefulness of REHABROBO-QUERY while querying about mechanical properties of robots, joint movements targeted by rehabilitation robots, evaluation metrics for rehabilitation robots, and publications about rehabilitation robots.

Table 10
Experimental Evaluation of Transformation and Query Answering

	Query	REHABROBO-CNL to SPARQL [sec]	Query Answering [sec]
Q1	What are the robots that target shoulder movements and that have at least 210° RoM for the flexion/extension movements of the shoulder?	0.0002	0.2277
Q2	What are the robots that target wrist movements and that have at least 2 active degrees of freedom?	0.0002	0.2083
Q3	What are the shoulder robots that target flexion/extension movements and that do not target elevation/depression movements?	0.0002	0.6580
Q4	What are the robots with active degree of freedom $\geq$ '3' and that target all pelvic girdle movements with backdrivability type = 'passive'?	0.2658	0.3479
Q5	What are the ankle robots that target movements with electrical actuation and with cable drive transmission?	0.0003	0.1955
Q6	What are the movements that are targeted by some robots with (some intervention time or with all targeted disorders)?	0.0002	0.5368
Q7	What are the movements that are targeted by the robot 'AssistOn-Finger' and with minimum range of motion $\geq$ '20'?	0.0002	1.8593
Q8	What are the movements that are not targeted by any robots with kinematic type = 'redundant' and with mechanism type $\neq$ 'parallel'?	0.0002	0.3790
Q9	What are the effort metrics that are evaluated by some robots with active degree of freedom $\geq$ 2?	0.0002	0.2765
Q10	What are the movement quality metrics that are evaluated by all robots with motion capability = 'grounded'?	0.0003	0.2687
Q11	What are the kinematic aspect metrics that are evaluated by some robots that target all elbow movements?	0.1569	0.4238
Q12	What are the muscle strength metrics that are evaluated by robots that target all wrist movements with transmission = 'direct drive'?	0.1570	0.3465
Q13	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 $\geq$ '30'?	0.1646	0.2351
Q14	What are the publications with clinical study and that do not reference any robots with active degree of freedom $\geq$ 1?	0.0003	0.1345
Q15	What are the publications without clinical study or that reference some robots that do not evaluate any movement quality metrics?	0.0002	0.1474
Q16	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'?	0.0003	0.1495
Q17	What are the users that own/maintain some robots that target all ankle movements?	0.1392	0.1148
Q18	What are the robots that target some wrist movements with actuation='series elastic'?	0.0004	0.1563
Q19	What are the robots with no targeted disorder or (with intervention time!='chronic' and with motion capability='grounded') or with no disorder level?	0.0003	0.1432
Q20	What are the robots with interaction type = 'exoskeleton' and that target some finger movements (with actuation = 'electrical' or with actuation = 'hydraulic' or with actuation = 'series elastic') ?	0.0003	0.1725

Table 11
Survey Questions and Summary Statistics

q1: Please rate the usefulness of the following aspects of REHABROBO-QUERY while querying about		
	Mean	σ^2
	4.08	0.70
Mechanical properties of robots	4.07	0.80
Joint movements targeted by rehabilitation robots	4.00	0.65
Evaluation metrics for rehabilitation robots	3.93	0.70
Publications about rehabilitation robots	4.33	0.62
q2: For what purpose would you use REHABROBO-QUERY in your research?		
	Mean	σ^2
	4.18	0.81
Identifying relevant rehabilitation robots suitable for a therapy	4.40	0.83
Identifying researchers/labs for relevant rehabilitation robots	4.00	0.65
Identifying outcomes of clinical studies for relevant rehabilitation robots	4.13	0.91
q3: Overall how would you rate the usefulness of REHABROBO-QUERY for the following user groups?		
	Mean	σ^2
	4.33	0.66
Rehabilitation robotics engineers	4.67	0.49
Physical medicine experts	4.14	0.66
Hospitals	3.92	0.76
Students/researchers	4.53	0.52

- From q2, we can infer that participants find REHABROBO-QUERY useful for identifying relevant rehabilitation robots suitable for a therapy, identifying researchers/labs for relevant rehabilitation robots, and identifying outcomes of clinical studies for relevant rehabilitation robots.
- Responses to q3 indicate that participants evaluated REHABROBO-QUERY to be highly useful for rehabilitation robotics engineers, students/reseracher and physical medicine experts, while also finding it useful for hospitals.

8. 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 [4,6,8,21,25,26,30,32,43,45,47] 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).

One of the most recently developed systems is BIOQUERY-ASP [17], which is a software system that answers natural language queries over biomedical databases and ontologies related to drug discovery. It utilizes Answer Set Programming (ASP) [31] 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 [22] and CLASP-NK, and it also provides explanations to the queries. BIOQUERY-ASP covers sophisticated queries with nested relative clauses, aggregates, superlatives, nonmonotonic negation, etc. as necessitated by its domain of drug discovery; and to represent and answer these queries, ASP provides a better framework. REHABROBO-QUERY covers simpler queries due to the requirements of its domain of rehabilitation robotics, and thus sophisticated aspects of ASP are not

needed. W3C recommended Semantic Web technologies provide a sufficiently good framework to represent and answer these queries. Both systems provide an interactive intelligent user-interface to construct queries with respect to their own CNLs, avoiding ambiguities.

Ferrández et al. [20] 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. Although there is no formal description of what sorts of queries are supported by QACID, all of the examples given in the paper are simple queries that do not involve disjunctions, negations, (nested) relative clauses, and quantifiers. In that sense, REHABROBO-QUERY supports a more variety of queries. In terms of user interface, since QACID allows the users to enter queries as free text, there may be ambiguities. REHABROBO-QUERY provides an interactive user-interface to enter queries, and thus prevents ambiguities in the query.

FREYA [10] is developed by the creators of and as a development upon QUESTIO [43]. In order to support natural language queries, it uses Stanford Parser [11] to generate a parse tree. Then, using GATE libraries [9], 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. REHABROBO-QUERY supports queries with negation, and can guide the user to construct queries according to its CNL without any ambiguities.

Lopez et al. [33] introduce PowerAqua, which is evolved from AquaLog [32]. It provides natural language querying over multiple ontologies; thus, supports high scalability and portability. It uses GATE libraries and WordNet [18] 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. REHABROBO-QUERY supports queries with quantifiers. In terms of user interface, PowerAqua queries are entered as free text, while some examples are illustrated to the user if requested. In that sense, there may be ambiguities of queries. Due to its CNL-based interactive interface, REHABROBO-QUERY prevents ambiguous queries.

Valencia-García et al. [44] 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. OWLPath provides suggestions for queries, relative to the terminology of its CNL; in that sense, REHABROBO-QUERY and OWLPath have some similarities. 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. In that sense, the underlying transformation of REHABROBO-QUERY is more systematic in covering other varieties of queries (e.g., with negation).

9. 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 would like to thank members of European Network on Robotics for NeuroRehabilitation for useful discussions. We are also very grateful for the participants of our survey about REHABROBO-QUERY.

References

[1] Global burden of stroke. `http://www.who.int/cardiovascular_diseases/en/cvd_atlas_15_burden_stroke.pdf` .

[2] Economic cost of stroke. `http://www.who.int/cardiovascular_diseases/en/cvd_atlas_17_economics.pdf` .

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

[4] 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.

[5] N. Bayona, K. Bitensky J.and Salter, and R. Teasell. The role of task-specific training in rehabilitation therapies. *Top Stroke Rehab.*, 12: 58–65, 2005.

[6] A. Bernstein and E. Kaufmann. GINO - a guided input natural language ontology editor. In *Proc. of ISWC*, pages 144–157, 2006.

[7] H. D. P. Butefisch, C.and Hummelsheim and K. Mauritz. Repetitive training of isolated movements improves the outcome of motor rehabilitation of the centrally paretic hand. *J. Neurol. Sci.*, 130:59–68, 1995.

[8] 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.

[9] 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.

[10] D. Damljanovic, M. Agatonovic, and H. Cunningham. FREyA: an interactive way of querying linked data using natural language. In *Proc. of ESWC*, pages 125–138, 2012.

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

[12] V. der Lee, S. J. H., B. I. A., L. H., R. C. G. J., Wagenaar, and L. M. Bouter. Exercise therapy for arm function in stroke patients: a systematic review of randomized controlled trials. *Clinical Rehab.*, 5:20–31, 2001.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

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

[20] 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.

[21] 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.

[22] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. clasp: A conflict-driven answer set solver. In *Proc. of LPNMR*, pages 260–265, 2007.

[23] 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.

[24] 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.

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

[26] E. Kaufmann, A. Bernstein, and L. Fischer. NLP-Reduce: A naive but domain-independent natural language interface for querying ontologies. In *Proc. of ESWC*, 2007.

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

[28] G. Kwakkel, R. Wagenaar, J. Twisk, G. Langkhorst, and J. Koetsier. Intensity of leg and arm training after primary middle-cerebral artery stroke: A randomized trial. *Lancet*, 35: 191–196, 1999.

[29] G. Kwakkel, B. J. Kollen, and H. I. Krebs. Effects of Robot-Assisted Therapy on Upper Limb Recovery After Stroke: A Systematic Review. *Neurorehabilitation and Neural Repair*, 22:111–121, 2008.

[30] 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, 2006.

[31] V. Lifschitz. What is answer set programming?. In *Proc. of AAAI*, pages 1594–1597, 2008.

[32] 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.

[33] 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.

[34] J. M. McDowd, D. L. Filion, P. S. Pohl, L. G. Richards, and W. Stiers. Attentional abilities and functional outcomes following stroke. *The Journals of Gerontology Series B: Psychological Sciences and Social Sciences*, 58(1):45–53, 2003.

[35] K. Nykanen. The effectiveness of robot-aided upper limb therapy in stroke rehabilitation: A systematic review of randomized controlled studies. Master's thesis, Jyvaskyla University, 2010.

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

[37] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and Complexity of SPARQL. In *Proc. of ISWC*, pages 30–43, 2006.

[38] T. Platz. Evidence-based arm rehabilitation – A systematic review of the literature. *Nervenartz*, 74(10):841–849, 2003.

[39] G. B. Prange, M. J. Jannink, C. G. Groothuis-Oudshoorn, H. J. Hermens, and M. J. Ijzerman. Systematic review of the effect of robot-aided therapy on recovery of the hemiparetic arm after stroke. *J. of Rehab. Research & Development*, 43: 171–184, 2006.

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

[41] 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.

[42] A. Sunderland, D. Tinson, E. Bradley, D. Fletcher, H. Langton, and D. Wade. Enhanced physical therapy improves recovery of arm function after stroke: A randomised clinical trial. *J. Neurol. Neursurg. Psychiatr.*, 55:530–535, 1992.

[43] V. Tablan, D. Damljanovic, and K. Bontcheva. A natural language query interface to structured information. In *Proc. of ESWC*, pages 361–375, 2008.

[44] 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.

[45] C. Wang, M. Xiong, Q. Zhou, and Y. Yu. PANTO: A portable natural language interface to ontologies. In *Proc. of ESWC*, pages 473–487, 2007.

[46] M. Yalcin and V. Patoglu. Kinematics and design of AssistOn-SE: A self-adjusting shoulder-elbow exoskeleton. In *Proc. of IEEE BioRob*, pages 1579–1585, 2012.

[47] Q. Zhou, C. Wang, M. Xiong, H. Wang, and Y. Yu. SPARK: adapting keyword query to semantic search. In *Proc. of ISWC/ASWC*, pages 694–707, 2007.