Querying Ontologies: A Controlled English Interface for End-users

Abraham Bernstein, Esther Kaufmann, Anne Göhring, Christoph Kiefer

University of Zurich, Department of Informatics, Winterthurerstrasse 190,
8057 Zurich, Switzerland
{bernstein, kaufmann, goehring, kiefer}@ifi.unizh.ch,
http://www.ifi.unizh.ch/ddis

Abstract. The semantic web presents the vision of a distributed, dynamically growing knowledge base founded on formal logic. Common users, however, seem to have problems even with the simplest Boolean expressions. As queries from web search engines show, the great majority of users simply do not use Boolean expressions. So how can we help users to query a web of logic that they do not seem to understand? We address this problem by presenting a natural language interface to semantic web querying. The interface allows formulating queries in Attempto Controlled English (ACE), a subset of natural English. Each ACE query is translated into a discourse representation structure – a variant of the language of first-order logic – that is then translated into an N3-based semantic web querying language using an ontology-based rewriting framework. As the validation shows, our approach offers great potential for bridging the gap between the logic-based semantic web and its real-world users, since it allows users to query the semantic web without having to learn an unfamiliar formal language. Furthermore, we found that users liked our approach and designed good queries resulting in a very good retrieval performance (100% precision and 90% recall).

1 Introduction

The semantic web presents the vision of a dynamically growing knowledge base that should allow users to draw on and combine distributed information sources specified in languages based on formal logic. Common users, however, were shown to have problems even with the simplest Boolean expressions; the use of the description logic formalism underlying the semantic web is beyond their understanding. Experience in information retrieval, for example, demonstrates that users are better at understanding graphical query interfaces than simple Boolean queries [1]. As queries from web search engines reveal, the great majority of users simply do not use Boolean expressions. Bowen and colleagues even show that people (CS students) who are trained in formulating queries in a logic-based formalism (SQL in their case) are usually inept in composing correct queries in realistically-sized databases rather than the small toy examples used in database classes [2]. *So how can we bridge the gap between the* *(description) logic-based semantic web and real-world users, who are at least ill at ease and, oftentimes, unable to use formal logic concepts?*

We address this problem by *presenting a natural language interface to the semantic web*. In its current form the interface provides users with a controlled natural language interface to formulate queries. The controlled natural language used, Attempto Controlled English (ACE) [3, 4], is an unambiguous subset of English, which is translated *automatically* into the N3-style[1] triple-based semantic web query language PQL [5] (which can easily be mapped to query languages such as SquishQL [6]). It provides the users with an almost natural language interface to the semantic web. As experience with controlled languages has shown, they are much easier to learn by end-users than formal languages like logic and are sufficient for querying knowledge bases [7]. We, therefore, believe that the approach presented here has great potential in bridging the gap between the semantic web and its end-users as well as becoming a major enabler for the growth of the semantic web.

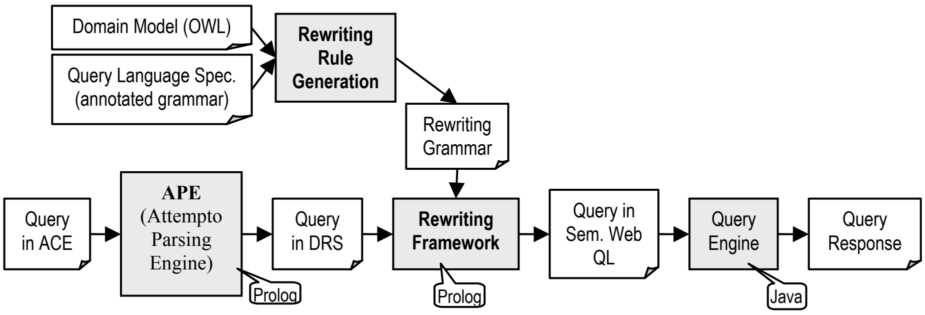


Fig. 1. Overall data flow of the controlled English query interface

The rest of this paper closely follows the data flow of the query interface (Fig. 1). Section 2 introduces Attempto Controlled English (ACE) and the Attempto Parsing Engine (APE). APE translates ACE texts into a discourse representation structure (DRS), a variant of the language of first-order logic introduced by Kamp and Reyle [8]. Section 3 describes the rewriting framework that translates the DRS to the semantic web query language. The translation is based on a rewriting grammar, which was generated using both an OWL-based domain model and a query language specification. The queries are evaluated by a standard query engine not discussed in this paper. Note that we used APE as a black-box component, which uses extended DRSs as internal representations. This allows us to exchange it with another NLP parser should the need arise. Therefore, we did not attempt a direct translation from ACE to N3. In section 4 we provide three evaluations of the approach. We close with a discussion of the current limitations as well as related and future work.

---

[1] More information about N3 can be found at http://www.w3.org/DesignIssues/Notation3

2 Attempto Controlled English as a Query Language

Our query interface automatically processes queries expressed in Attempto Controlled English (ACE), a controlled natural language originally designed for requirements specifications and knowledge representation [3, 4]. ACE is a subset of English meaning that each ACE sentence is correct English, but not vice-versa. ACE's grammar is specified by a small set of construction and interpretation rules. The construction rules allow users to build simple sentences (e.g., “John sells books.”), composite sentences (e.g., “If John sells books and John's business does not fail then he is content.”), and queries (e.g., “Which books does John sell?”). The interpretation rules eliminate syntactic and semantic ambiguities, for which natural languages are highly notorious, hereby also reducing the computational complexity of processing ACE sentences. As such, ACE avoids the major disadvantages of full natural language processing, while maintaining the ease of use for end-users and allowing the translation of all ACE sentences to first-order logic.

Though ACE appears completely natural, it is in fact a formal language and its small set of construction and interpretation rules must be learned. As an example, consider the sentence “A man sees a girl with a telescope.” In full English this sentence is ambiguous since the prepositional phrase “with a telescope” can either modify the verb phrase “sees”, leading to the interpretation that the man has the telescope, or the noun phrase “a girl”, meaning that the girl has the telescope. In ACE, however, the sentence is unambiguous since an interpretation rule limits the meaning to the first alternative “sees with a telescope”.

DRS	First-order Logic
A B	$\exists A\ B : \mathrm{customer}(A) \wedge \mathrm{book}(B) \wedge \mathrm{buy}(A, B)$

customer(A)
book(B)
buy(A, B)	

Fig. 2. DRS and first-order logic representation of “A customer buys a book.”

The Attempto Parsing Engine (APE) – implemented in Prolog as a Definite Clause Grammar – translates a possibly multi-sentence ACE text into a *discourse representation structure* (DRS) that logically represents the information of the text [8]. DRSs are a powerful means to adequately capture linguistic phenomena, for instance anaphoric references. A DRS consists of discourse referents, i.e., quantified variables representing the objects of a discourse, and of conditions for the discourse referents. The conditions can be logical atoms or complex conditions built from other DRSs and logical connectors (negation, disjunction, and implication). As an example, the translation of the sentence “A customer buys a book.” is shown in its typical box-styled DRS representation in Fig. 2 on the left. The two discourse referents, A and B, are shown at the top and the three conditions derived from the sentence are listed below. Fig. 2 shows on the right the first-order logic formula equivalent to the DRS.[2]

---

[2] To emphasize the principle of the translation we radically simplified the DRSs in all examples. Real DRSs are much more complex to adequately represent a wide range of linguistic phenomena.

3 The Rewriting Framework: From DRSs to Queries

The next (and central) step in our natural language semantic web interface is the rewriting of the APE generated DRSs into a semantic web query language (an extension and modification of [9]). To that end we generated a DRS-to-QL rewriting grammar using an ontology-based domain-model (in OWL) and a query language specification (cf. Fig. 1). This section will first succinctly introduce the exemplary domain ontology – the MIT Process Handbook [10] – which will provide the underlying examples throughout the text. Then, it will introduce the rewriting rule generation and the rewriting framework, which has both ontology specific as well as general vocabulary rules.

3.1 An Example Ontology: MIT Process Handbook

As an example ontology we chose the *MIT Process Handbook* [10] which describes organizational processes. The Process Handbook treats a real-world domain that everybody can relate to, has a large number of instances (>5000), and has been used in a number of semantic web projects. Each process (object) of the ontology enters a variety of relationships to attributes, sub-processes, exceptions, etc., and has a detailed textual description (cf. Fig. 3**).**

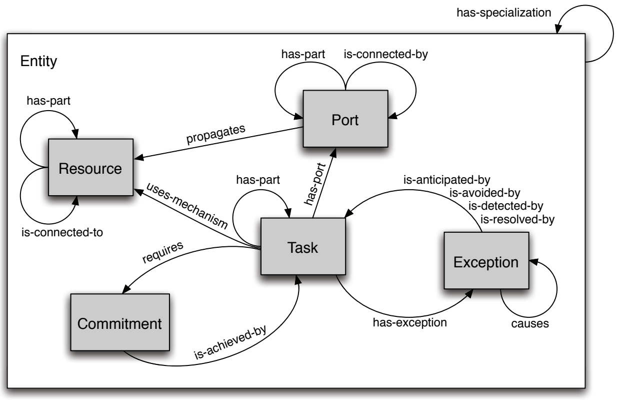


Fig. 3. The Process Handbook Meta-model

The *process query language* (PQL) presented in [5] allows to pose queries, which are then evaluated against the process ontology. PQL essentially allows the composition of (process) ontology fragments that result in a query-by-example style specification of the sought-after processes. It can be mapped straightforwardly to any triple-based semantic web query language such as SquishQL [6]. Consequently, none of our findings are limited to the Process Handbook and PQL.

PQL supports two major statement types: the first one queries for the subject and/or the predicate of a given property; the second doesn't make any assumptions about the property but does (mostly) assume that the object is a literal.[3] Fig. 4 shows an example full-text query and its corresponding triple-based query.

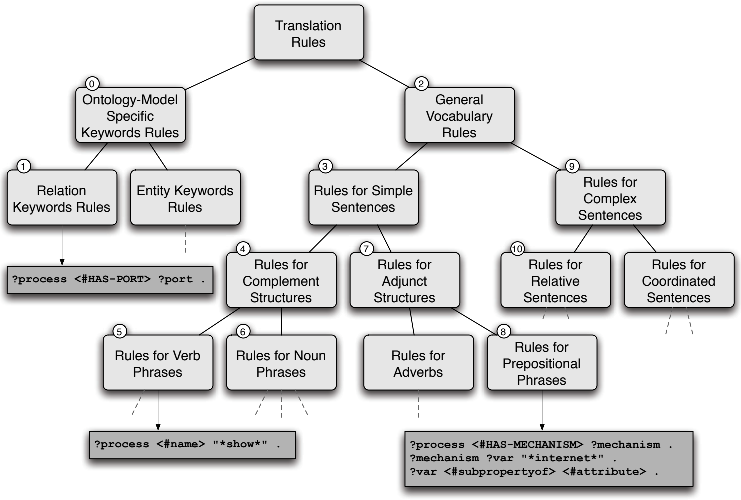
Full-text and Keywords	N3-style PQL-Query[4]
"Find all processes that sell books over the internet." *Keywords:* "sell book internet"	

```
?process   <#name>   “*sell*” ,
                     “*book*” ;
           <#has-mechanism> ?mechanism .
?mechanism ?var “*internet*” .
?var       <#subpropertyof> <#attribute> .
```

Fig. 4. An example full-text query with its corresponding keywords and derived N3-style PQL query

3.2 The Rewriting Rule Generation and Framework

In order to translate the DRSs generated by APE into triples and N3-style PQL queries, we developed rewriting rules for the DRS structures. Each linguistic structure is first matched against a set of *ontology-model specific keyword rules* that – when they apply – result in a constraint between objects (i.e., a query statement with a fixed property).


Translation Rules
0
Ontology-Model Specific Keywords Rules
2
General Vocabulary Rules
1
Relation Keywords Rules
Entity Keywords Rules
3
Rules for Simple Sentences
9
Rules for Complex Sentences
4
Rules for Complement Structures
7
Rules for Adjunct Structures
10
Rules for Relative Sentences
Rules for Coordinated Sentences
5
Rules for Verb Phrases
6
Rules for Noun Phrases
Rules for Adverbs
8
Rules for Prepositional Phrases

```
?process <#HAS-PORT> ?port .
```

```
?process <#name> "*show*" .
```

```
?process <#HAS-MECHANISM> ?mechanism .
?mechanism ?var "*internet*" .
?var <#subpropertyof> <#attribute> .
```

Fig. 5. The translation rules grammar (numbers are referred to in the text)

If none of these rules applies, then a set of *general vocabulary rules* is tried, typically resulting in the comparison with a literal value (i.e., a query statement without a known property). This structure is reflected in Fig 5., which provides a graphical

---

[3] Note that we follow the *subject–property–object* designation of triples throughout the paper.

[4] For the syntax of the triple queries we slightly extended the N3-syntax to allow for substring matching. The literal "book" matches any other literal "book." The literal "*book*" matches any other literal, which contains the substring "book."

overview of the most important rules of the translation grammar. To further explain this approach we will now discuss each of these two rule types referring to the rules by their numbers in Fig. 5.

3.2.1 Ontology-model Specific Keyword Rules

The *ontology-model specific keyword rules* apply if one of the keywords of the ontology – including its morphological or syntactic variants – appears in the DRS to be translated. For example, the expression “has a port” in the query “Which process has a port?” is identified as the ontology-model property HAS-PORT and, hence, translated into the triple-based PQL query in Ex. 1 (firing rules ⓪ and ① in Fig. 5).

ACE: Which process has a port?	
DRS	**N3-style PQL Query**
A B C	`?process  <#has-port>  ?port .`

query(A, which)
object(A, process, object)
object(B, port, object)
predicate(C, state, have, A, B)	

Ex. 1. Transformation of “Which process has a port?”

A limitation of this approach is the choice of the vocabulary when building the ontology. In some cases we, therefore, had to include synonyms of the ontology-keywords in the rewriting rules.[5]

3.2.2 General Vocabulary Rules

Elements of the DRS not handled by the ontology-model specific keyword rules are passed to the *general-vocabulary rules* (② in Fig 5.). These rules distinguish between *simple* and *complex* sentences. Simple sentences don’t contain embedded relative sentences or a coordination of sentences (using connectors such as “and,” “or,” etc.) whereas complex sentences consist of more than one sentence.

Simple Sentences Rules. If a simple sentence is identified (③ in Fig. 5) the framework differentiates between *complement and adjunct clauses*. Complements correspond to the mandatory elements of a sentence (also called *arguments*). Adjuncts comprise elements of a sentence that are not required by the sentence’s main verb. Consider the sentence: “Which service shows the campus restaurants over the internet?” The main verb “shows” calls for the arguments “which service” and “the campus restaurants”. The prepositional phrase “over the internet” provides additional, non-mandatory information to the sentence. The motivation for this distinction in our framework derives from the idea that complements contribute more pivotal information to a sentence’s meaning than adjunctive structures. Exploiting this aspect for query formulation, we mirror the syntactic structure of an ACE sentence in the query using the ontology model.

---

[5] We intend to extend our framework with automated keyword expansion using WordNet.

ACE: Which service **shows** the campus restaurants?	
DRS	**N3-style PQL Query**

A B C
query(A, which)
object(A, service, object)
predicate(B, event, show, A, C)
object(C, 'campus restaurant', object)
	`?process <#name> “*show*” .`

Ex. 2. Transformation of the sentence’s main verb “show”

Complement Structures Rules. Complements consist of verb phrases, noun phrases, prepositional phrases, and adjective phrases. They are interpreted as simple literal values. For example, the verb “show” in the above query “Which service shows the campus restaurants?” is represented in the DRS as “predicate(B,event,show,A,C)”. It is treated as a literal value and translated as shown in Ex. 2 (firing rules ④ and ⑤ in Fig. 5).

ACE: Which service shows the menus **of the campus restaurants**?	
DRS	**N3-style PQL Query**

A B C D
query(A, which)
object(A, service, object)
predicate(B, event, show, A, C)
object(C, menu, object)
relation(C, menu, of, D)
object(D, ‘campus restaurant’, object)
	`?process <#name> “*show*” ,`
	`“*menu*” ,`
	`“*campus*” ,`
	`“*restaurant*” .`

Ex. 3. Transformation of “Which service shows the menus of the campus restaurants?”

In ACE any noun phrase can furthermore be coordinated (e.g., *menus and drinks*), modified by adjectives (e.g., *the different restaurants*), *of*-prepositional phrases (e.g., *the menus of the restaurants*), and possessive elements (e.g., *the restaurants’ menus*). These modifiers are also treated as literal values. Ex. 3 shows a simple sentence consisting of a verb and two complements with a modifying *of*-prepositional phrase in the object complement (firing rules ④ and ⑥). Note that the rewriting framework splits the compound “campus restaurants” into its constituents to improve recall.

Adjunct Structures Rules. If a simple sentence consists of complement elements as well as adjunct elements, the resulting query inherits this linguistic differentiation. Consider the query “Which service shows the menus of the campus restaurants over the internet?”. Here, the prepositional phrase “over the internet” indicates that the menus are shown using the internet as an instrument, which is noted in the sentence’s DRS. As instruments, or rather their synonym “mechanisms”, are included in the Process Handbook ontology-model as the HAS-MECHANISM property, we can translate the phrase “over the internet” into the PQL query in Ex. 4 (firing rules ⓪, ⑦, and ⑧).

ACE: Which service shows the menus of the campus restaurants **over the internet**?	
DRS	**N3-style PQL Query**
A B C D E	

query(A, which)
object(A, service, object)
predicate(B, event, show, A, C)
object(C, menu, object)
relation(C, menu, of, D)
object(D, 'campus restaurant', object)
object(E, internet, object)
modifier(B, location, over, E)
	`?process  <#has-mechanism> ?mechanism .`
	`?mechanism ?var “*internet*” .`
	`?var  <#subpropertyof>  <#attribute>`[6] `.`

Ex. 4. Transformation of the prepositional phrase “over the internet”

Complex Sentences Rules. Similar to adjunct structures, complex sentences initiate a search in the ontology-model for corresponding relationships indicating that the nested syntactic structures of ACE queries are used to phrase structured queries. Complex sentences are composed of more than one sentence. In the sentence “Which service provides all available pizza couriers that are in the city?” the compound “pizza couriers” is modified by a relative sentence turning the simple sentence “Which service provides all available pizza couriers?” into a complex sentence. The complex syntactic structure is exploited in our translation framework resulting in a query that searches for corresponding relationships in the ontology (rules ⑨ and ⑩) as shown in Ex. 5.

We emphasize the linguistic difference between the main sentence and the embedded relative sentence by searching for the relative sentence’s literal values not only in the specific attribute “Name” of the process’ subparts but in all attributes of the subparts. The query becomes less restrictive in order to improve recall.

ACE: Which service provides all available pizza couriers **that are in the city**?	
DRS	**N3-style PQL Query**
A B C D E	

query(A, which)
object(A, service, object)
object(B, 'pizza delivery', object)
predicate(C, event, provide, A, B)
object(D, city, object)
predicate(E, state, be, B)
modifier(E, location, in, D)
	`?process  <#name> “*provide*” ,`
	`                   “*pizza*” ,`
	`                   “*courier*” .`
	`?process  <#has-part> ?part .`
	`?part ?var “*city*” .`
	`?var  <#subpropertyof>  <#attribute> .`

Ex. 5. Transformation of “Which service provides all available pizza couriers that are …”

If sentences are coordinated by conjunction (*and*) or disjunction (*or*) the result is again a complex sentence. An example is “Which service provides all available pizza couriers over the internet and which service takes orders 24-hours-a-day?” Each coordinated sentence is translated into a separate set of query statements according to the simple sentences rules (③). In addition, the conjunction “and” triggers the translation rules for complex sentences (⑨) which ensure that the overall sentence is translated into one cohesive query.

---

[6] The subproperty statement ensures that ?var is only unified to an attribute property preventing the unification with a structure property.

3.2.3 Post Processing Rules

At the end of the rewriting procedure the framework applies some post processing rules priorizing the fired rewriting rules or simplifying the resulting query. For example: If the search in the ontology-model results in no corresponding relationships, then the structure is simplified by treating the modifiers as literals. The following example illustrates the simplification of the modifier “24 hours a day” in the sentence “Which pizza courier takes orders 24 hours a day?”.

Query according to general vocabulary rules:

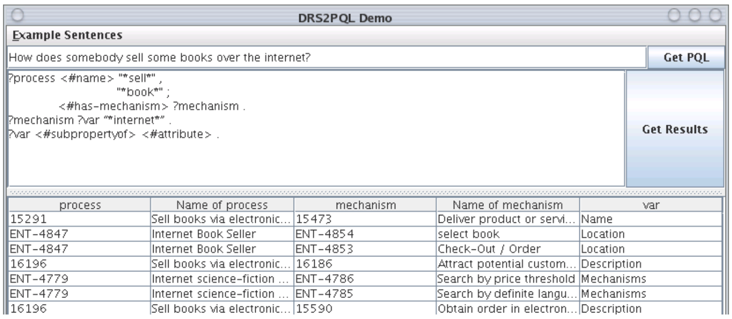
```
    ?process  <#has-part>  ?part .
    ?part ?var "*24 hours a day*" .
    ?var  <#subpropertyof> <#attribute> .
```

Simplified query according to the post processing rules:

```
    ?process ?var "*24 hours a day*" .
    ?var  <#subpropertyof> <#attribute> .
```

4 Validation

For the implementation of the validation prototype we combined Prolog and Java components, as APE and the rewriting framework are programmed in SICStus Prolog, and the user interface and the query engine are programmed in Java (see Fig. 1). Currently, ACE queries are entered into the user interface and then passed to APE using the *Jasper* Java-to-Prolog bridge. The resulting DRSs are forwarded to the rewriting framework that generates the semantic web query language queries. These are then evaluated by the query engine that passes the result back to the user interface (Fig. 6).



How does somebody sell some books over the internet?

```
?process <#name> "*sell*" ,
                  "*book*" ;
         <#has-mechanism> ?mechanism .
?mechanism ?var "*internet*" .
?var <#subpropertyof> <#attribute> .
```

process	Name of process	mechanism	Name of mechanism	var
15291	Sell books via electronic...	15473	Deliver product or servi...	Name
ENT-4847	Internet Book Seller	ENT-4854	select book	Location
ENT-4847	Internet Book Seller	ENT-4853	Check-Out / Order	Location
16196	Sell books via electronic...	16186	Attract potential custom...	Description
ENT-4779	Internet science-fiction ...	ENT-4786	Search by price threshold	Mechanisms
ENT-4779	Internet science-fiction ...	ENT-4785	Search by definite langu...	Mechanisms
16196	Sell books via electronic...	15590	Obtain order in electron...	Description

Fig. 6. The user interface of the query engine showing an ACE query, its corresponding N3-style PQL representation, and the results from the database matching the query

We chose this mixed-programming language approach as we used APE as a black-box (indeed we did not make any changes to its source code) and found that the prolog-style data-structures generated by APE where easiest processed in a rewriting framework using the same language.

Using the prototype we validated our approach in three ways. First we tried to generate correct translations for real-world queries. Next, we confronted users with tasks

tasks in which they had to retrieve answers from a semantic web database and measured the users' performance as well as utilized a standardized usability test to assess the ease of usage compared to using a formal query language. Additionally, we measured precision and recall of the resulting answers. Last, we compared the retrieval performance of our framework to two different keyword-based retrieval approaches using an exemplary query.

4.1 Validation of the Rewriting Framework with Real-world Queries

To ensure the correct translation of real-world queries we asked masters students to phrase queries, which search for web services that would be of interest to them. We also asked them to enter the queries in a query-by-example-style form. Fig. 7 shows a selection of these queries sorted by increasing syntactic complexity. We received 50 queries, reformulated them in ACE, and ran them through our query interface. All reformulations were very simple (such as adding articles/determiners or using relative sentences instead of certain types of connectors). The system translated all queries correctly taking an average processing time of about 2 seconds (on a standard PC with a 2 GHz Celeron processor and 512 MB RAM).

Which service provides a shoe cleaning service?
Which service helps with the classes and the exams?
Which service provides the summaries of the different courses for free over the web?
Which service provides an internet streaming server that streams the requested tracks over the internet?
Which service provides a car renting and uses a web interface that allows a keyword search?
Which service takes the groceries orders via a website and delivers the food within 24 hours?
Where does somebody enrol to a university and choose the courses and get a personal university scheduler?
Which internet page shows the movies that are on in the city and provides a seat booking?
Where does somebody enter some hardware components and the service returns a list which has a sorting by price?
Which service provides the songs of the different artists and the customers pick the desired songs over the internet?

Fig. 7. A selection of real-world ACE queries for which the query interface generated correct N3-style PQL queries

4.2 Usability and Performance Evaluation in a Retrieval Task

We also wanted to evaluate the *interface's usability in a concrete usability task*. To that end we used the NLP database interface evaluation tasks defined by [11], in which 1770 queries are defined to be run on three different databases. We translated the databases into OWL to make it accessible from our query processor. We then randomly chose 30 questions of varying complexity and asked 20 users to compose queries both using our system as well as a simplified version of SQL. As a preparation, the subjects, whom we recruited from the computer science and computer linguistics departments, read a 2-page instruction on how to construct correct ACE sentences and a ½ page refresher on SQL.

We found that users where significantly faster in writing the ACE queries than the SQL queries (t-test with p = 2.84E-05). Using the standardized SUS-test [12] for usability, we found that ACE performed significantly better than SQL in the SUS test questions “I found the various functions of ACE were well integrated”, “I think there was too much inconsistency in ACE”, and “I would imagine that most people would learn to use ACE very quickly” (at p = 2.8%, 0.4%, and 4.1%). Furthermore, people overall preferred ACE over SQL, barely missing significance at the 5% level (with a t-test result of 5.6%). None of the questions in which SQL performed better on average yielded significant results.

Note that these results are influenced by the subject pool, which is composed of people who are very familiar with both computers and formalized languages. Experiences with logic-based query languages suggest that the average population will experience more problems with a language like SQL and even perform worse than the computer and logic educated subjects we had [1, 2]. Consequently, we have reason to believe that the general population will have an even larger affinity towards ACE, but will also have to climb a slightly steeper learning curve to learn it.

To evaluate the *retrieval performance of the overall system* we executed the 30 queries formulated in ACE by the users and partly corrected to valid ACE sentences on the Mooney Natural Language Learning Data [11]. The retrieved answers achieved a precision of 100% and a recall of 90%.

The performed retrieval task highlights that subjects with no previous familiarity with ACE can translate real-world queries to correct ACE queries (faster than to SQL), which in turn are processed correctly by our rewriting system resulting in a very good retrieval performance.

4.3 Exemplary Validation with a Complicated Query Sentence

We also executed a number of highly complex queries and compared their retrieval performance with two keyword-based retrieval approaches: one using a TFIDF-style ranking [13], the other one searching for the conjunction of keywords. Both of those approaches have a proven track record of being suitable for end-users. We then hand-coded the database to find the correct results for the natural language queries.

For the non-trivial query presented in Ex. 6 the database contained four correct answers. Our NLP query interface found three correct answers, missing one. The TFIDF-ranking found the correct answers at the 2nd, 35th, 47th, and 183rd positions, which provides an overall better recall than our approach but at the cost of an abysmal precision. The simple keyword matcher returned no answers as the conjunction of all keywords overconstrained the query. This example indicates that our approach – while maintaining natural language simplicity – provides a performance akin to logic-based retrieval engines that usually outperform precision and recall of keyword engines.

Summarizing our evaluation results, we have found that ACE can correctly process real-word queries, which are slightly reformulated from students’ textual descriptions. Using the standardized usability tests we also found that people prefer ACE-querying over SQL, even though most of the subjects had no ACE but good SQL knowledge before the usability task. Executing the ACE queries with our framework and comparing the retrieved answers with the results of [11], we achieved a very good retrieval performance. Last, a non-trivial exemplary query indicated that ACE-queries also have the potential to be used to compose complex queries that are easily understood by users.

ACE: Which sales process informs its customers over the internet and avoids the unwanted solicitations with an opt-out list?

DRS

```
A B C D E F G H
query(A, which)
object(A, sales_process, object)
object(B, customer, person)
predicate(C, event, inform, A, B)
object(D, internet, object)
modifier(C, instrument, over, D)
object(E, solicitation, object)
property(F, unwanted, E)
predicate(G, event, avoid, A, E)
object(H, opt_out_list, object)
modifier(G, instrument, with, H)
```

PQL

```
?process  <#name>  “*sale*” ;
          <#has-part> ?part .
?part  ?varpart  “*inform*” .
?varpart  <#subpropertyof>  <#attribute> .
?part  ?varpart  “*customer*” .
?varpart  <#subpropertyof>  <#attribute> .
?part  <#uses-mechanism>  ?mechanism .
?mechanism  ?varmech  “*internet*” .
?varmech  <#subpropertyof>  <#attribute> .
?part  <#has-exception>  ?exception .
?exception  ?varex  “*unwanted*” .
?varex  <#subpropertyof>  <#attribute> .
?exception  ?varex  “*solicitation*” .
?varex  <#subpropertyof>  <#attribute> .
?exception  <#is-avoided-by>  ?handler .
?handler  ?varhand  “*opt-out*” .
?varhand  <#subpropertyof>  <#attribute> .
?handler  ?varhand  “*list*” .
?varhand  <#subpropertyof>  <#attribute> .
```

Ex. 6. Transformation of a complex query “Which sales process informs its customers over the internet and avoids the unwanted solicitations with an opt-out list?”

5 Limitations of Our Approach and Future Research

We can think of three limitations to the work presented in this paper. First, the use of a controlled language imposes a cost on the user since the language has to be learned. Users might be discouraged from employing a language they have to learn, but experience with ACE – and with other controlled languages such as Boeing Simplified English [14] – has shown that learning a controlled language to phrase statements and queries is much easier than learning logic, and takes only a couple of days for the basics and two weeks for full proficiency, which is beyond what users need to write queries. As our evaluation above shows, educated users (i.e., members of a computer science or computer linguistics department) were able to use ACE querying reasonably well after reading a 2-page explanatory text. Furthermore, some researchers are currently developing query interfaces that will help people to write correct controlled English sentences by guiding them as they write [15]. Last and most importantly, Malhotra [7] has shown that users tend to use a limited language when querying a knowledge base as opposed to conversing with other people indicating that the limitation might not be as grave. Similar results have recently been found by Dittenbach et al. [16] through the implementation of a multilingual natural language interface to a real web-based tourism platform. They show that most natural language queries are formulated in a simple manner and don't consist of complex sentence constructs even when users are neither limited by a conventional search interface nor narrowed by a restricted query language.

Second, our current prototype requires some manual adaptation of the rewrite rules when using it with a new ontology or new knowledge base. Given our experience with hand-adaptation, we found that most of the time an inspection of the meta-model was sufficient. Motivated by the work of Cimiano [17] we believe that the rules can be automatically generated based on the ontology model and intend to investigate this avenue in future work.

Last, the validations shown in this paper are slightly limited by the choice and size of the subject pool from among computer scientists/linguists. We, therefore, intend to extend the evaluation to more subjects with different backgrounds and compare our system's performance with other semantic-web query interfaces allowing us to investigate how people's retrieval performance and affinity to different tools is related to their background.

6 Related Work

We hardly found any other application of controlled natural language querying semantic web content. The most closely related work we encountered is the GAPP project [18], a question-answering system developed for querying the *Foundational Model of Anatomy* (FMA) knowledge base. GAPP takes natural language questions as input and translates them into the structured query language *StruQL*, a database language designed for querying graphs. The system then returns the results of a query as an XML document. Similar to our interface GAPP analyses English questions and divides them into the three elements *Subject*, *Relationship*, and *Object*. Along with pattern-matching and word-combination techniques, which resemble our ontology-model specific keywords rules, GAPP's parser exploits the syntactic structures in order to generate the appropriate structured queries. The results of the evaluation, where the generation of the correct query was considered to be a correct response, show that GAPP provides an intuitive and convenient way for anatomists to browse the FMA knowledge base. The approach differs from ours in that its query construction and, therefore, its overall application are highly restricted to one semantically constrained domain. Furthermore, their model doesn't use a full-fledged rewriting grammar but seems to be limited to a set of domain-specific user-defined pattern matching rules. Another project addressing a similar task is the MKBEEM project [19]. In contrast to our approach it focuses, largely, on adding multilinguality to the process of automated translation and interpretation of natural language user requests.

We also found that work on natural language interfaces to data bases (not ontologized knowledge bases) has largely tapered off since the 80's [20], even though the need for them has become increasingly acute. Accordingly, a few approaches in the area of database interfaces have emerged recently [21-23]. Among them the most closely related approach is the PRECISE project [24] that proposes a natural language interface to relational databases. PRECISE uses a data-base augmented tokenization of a query's parse tree to generate the most likely corresponding SQL statement. It is, consequently, limited to a sublanguage of English, i.e., the language defined by the subject area of the database. In contrast, our approach limits the possible language constructs and not the subject domain. Our interface will not return any useful answers when none can be found in the queried ontology. It will, however, be able to generate an appropriate triple-based statement. We hope to be able to include an empirical comparison between these two approaches in our future work.

7 Conclusions

People's familiarity with natural language might be the key to simplify their interaction with ontologies. Our approach provides exactly such a natural language interface. Following Malhotra's [7] and Dittenbach et al.'s [16] findings, which state that using a subset of English is sufficient to query knowledge bases, we could forgo the need for a full natural language processing machinery avoiding all the computational and linguistic complexities involved with such an endeavor. The result is a simple but adaptive approach to controlled English querying of the semantic web – a potentially important component for bridging the gap between real-world users and the logic-based underpinnings of the semantic web.

Acknowledgements

The authors would like to thank Norbert Fuchs and his Attempto team for providing APE, the MIT Process Handbook project for making available the data on which the evaluation is based, Ray Mooney and his group for having generously supplied the databases, English questions, and corresponding queries to us, and the anonymous reviewers for their helpful comments. This work was partially supported by the Swiss National Science Foundation (200021-100149/1).

References

1. Spoerri, A.: InfoCrystal: A Visual Tool for Information Retrieval Management. Second International Conference on Information and Knowledge Management. Washington, D.C. (1993) 11-20
2. Bowen, P.L., Chang, C.-J.A., Rohde, F.H.: Non-Length Based Query Challenges: An Initial Taxonomy. Fourteenth Annual Workshop on Information Technologies and Systems (WITS 2004). Washington, D.C. (2004) 74-79
3. Fuchs, N.E., et al.: Attempto Controlled English (ACE). (2003) http://www.ifi.unizh.ch/attempto
4. Fuchs, N.E., et al.: Extended Discourse Representation Structures in Attempto Controlled English. Technical Report IfI-2004. University of Zurich, Zurich (2004)
5. Klein, M., Bernstein, A.: Towards High-Precision Service Retrieval. IEEE Internet Computing 8/1 (2004) 30-36

6. Miller, L., Seaborne, A., Reggiori, A.: Three Implementations of SquishQL, a Simple RDF Query Language. The International Semantic Web Conference (ISWC2002). Sardinia, Italy (2002) 423-435
7. Malhotra, A.: Design Criteria for a Knowledge-based English Language System for Management: An Experimental Analysis. Ph.D. MIT Sloan School of Management, Cambridge, MA (1975)
8. Kamp, H., Reyle, U.: From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language. Kluwer, Dordrecht Boston London (1993)
9. Bernstein, A., et al.: Talking to the Semantic Web: A Controlled English Query Interface for Ontologies. Fourteenth Annual Workshop on Information Technologies and Systems (WITS 2004). Washington, D.C. (2004) 212-217
10. Malone, T.W., et al.: Tools for Inventing Organizations: Toward a Handbook of Organizational Processes. Management Science 45/3 (1999) 425-443
11. Tang, L.R., Mooney, R.J.: Using Multiple Clause Constructors in Inductive Logic Programming for Semantic Parsing. 12th European Conference on Machine Learning (ECML-2001). Freiburg, Germany (2001) 466-477
12. Brooke, J.: SUS - A "quick and dirty" Usability Scale. In: Jordan, P.W., et al., Editors: Usability Evaluation in Industry. Taylor & Francis, London (1996)
13. Salton, G., McGill, M.J.: Introduction to modern information retrieval. McGraw-Hill computer science series. McGraw-Hill, New York (1983)
14. Wojcik, R.H.: Personal Communication. Richard H. Wojcik is Manager of the Boing Simplified English Project, (2004)
15. Schwitter, R., Tilbrook, M.: Dynamic Semantics at Work. International Workshop on Logic and Engineering of Natural Language Semantics. Kanazawa, Japan (2004) 49-60
16. Dittenbach, M., Merkl, D., Berger, H.: A Natural Language Query Interface for Tourism Information. 10th International Conference on Information Technologies in Tourism (ENTER 2003). Helsinki, Finland (2003) 152-162
17. Cimiano, P.: ORAKEL: A Natural Language Interface to an F-Logic Knowledge Base. 9th International Conference on Applications of Natural Language to Information Systems (NLDB 2004). Salford, UK (2004) 401-406
18. Distelhorst, G., et al.: A Prototype Natural Language Interface to a Large Complex Knowledge Base, the Foundational Model of Anatomy. American Medical Informatics Association Annual Fall Symposium. Philadelphia, PA (2003) 200-204
19. MKBEEM: Multilingual Knowledge Based European Electronic Market Place. (2005) http://mkbeem.elibel.tm.fr/
20. Androutsopoulos, I., Ritchie, G.D., Thanisch, P.: Natural Language Interfaces to Databases - An Introduction. Natural Language Engineering 1/1 (1995) 29-81
21. Guarino, N., Masolo, C., Vetere, G.: OntoSeek: Content-Based Access to the Web. IEEE Intelligent Systems 14/3 (1999) 70-80
22. Andreason, T.: An Approach to Knowledge-based Query Evaluation. Fuzzy Sets and Systems 140/1 (2003) 75-91
23. Minock, M.: A Phrasal Approach to Natural Language Interfaces over Databases. Umeå Techreport UMINF-05.09. University of Umeå, Umeå (2005)
24. Popescu, A.-M., Etzioni, O., Kautz, H.: Towards a Theory of Natural Language Interfaces to Databases. 8th International Conference on Intelligent User Interfaces. Miami, FL (2003) 149-157