Talking to the Semantic Web –
A Controlled English Query Interface for Ontologies*

Abraham Bernstein, Esther Kaufmann, Norbert E. Fuchs, June von Bonin
Department of Informatics
University of Zurich, Switzerland
bernstein@ifi.unizh.ch, kaufmann@ifi.unizh.ch, fuchs@ifi.unizh.ch

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 expression. 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 front-end to semantic web querying. The front-end 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 the semantic web querying language PQL. As examples show, our approach offers great potential for bridging the gap between the semantic web and its real-world users, since it allows users to query the semantic web without having to learn an unfamiliar formal language.

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. Experience in information retrieval, for example, demonstrates that users are better at understanding graphical query interfaces than simple Boolean queries [Spoerri 1993]. As queries from web search engines reveal, the great majority of users simply do not use Boolean expressions. *So how can we bridge the gap between the 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 front-end to the semantic web*. In its current form the front-end provides users with a controlled natural language interface to formulate queries. The controlled natural language used, Attempto Controlled English (ACE) [Fuchs et al. 2003; Fuchs et al. 2004], is an unambiguous subset of English, which is translated *automatically* [Bonin 2004] into the semantic web query language PQL [Klein et al. 2004] providing 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. We, therefore, believe that the approach presented here has great potential in bridging the gap between the semantic web and its end-users and becoming a major enabler for the growth of the semantic web.

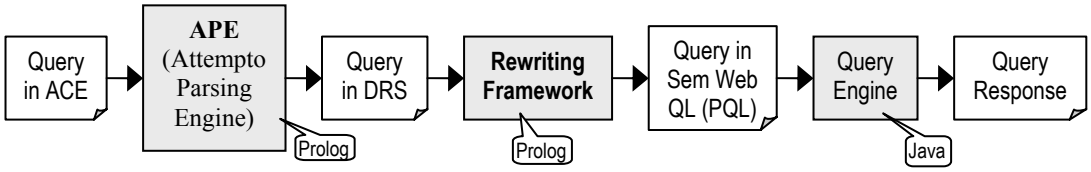

Figure 1: Overall data flow of the controlled English query front-end

* To appear in the proceedings of the 14th Workshop on Information Technology and Systems, held December 11-12, 2004, Washington D.C., USA.

The rest of this paper closely follows the data flow of the query front-end (Figure 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 collaborators [Kamp et al. 1993]. Section 3 introduces the rewriting framework that translates the DRS to the semantic web query language PQL. PQL queries are evaluated by a standard query engine that we do not discuss in this paper. In section 4 we provide a first assessment of the approach posing some real-world queries to the knowledge base. We close with a discussion of the current limitations as well as related and future work.

2. Attempto Controlled English as a Query Language

Our query front-end automatically processes queries expressed in Attempto Controlled English (ACE), a controlled natural language originally designed for requirements specifications and knowledge representation [Fuchs et al. 2003; Fuchs et al. 2004]. 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". To express the second alternative "a girl with a telescope" one could, for instance, write "A man sees a girl that has a telescope." making use of another – complementary – interpretation rule. Thus ACE's interpretation rules eliminate ambiguity without reducing expressability.

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

Figure 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 multisentential – ACE text into a discourse representation structure (DRS) that logically represents the information of the text [Kamp et al. 1993]. 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 Figure 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. Figure 2 shows on the right the first-order formula equivalent to the DRS. [1]

[1] To emphasize the principle of the translation we radically simplified the DRS. As will be seen later, real DRSs are more complex to adequately represent a wide range of linguistic phenomena.

3. From DRS to the Semantic Web Query Language PQL

As the next step, the rewriting framework (an extension of [Bonin 2004]) translates the DRS produced by APE into the semantic web query language PQL, which is then used to query an ontology. As an exemplary ontology we chose the *MIT Process Handbook* [Malone et al. 1999] that 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. The process query language (PQL) presented in [Klein et al. 2004] allows to pose queries which are then evaluated against the process ontology. PQL essentially allows the composition of process fragments that result in a query-by-example style specification of the sought after processes. PQL's two major statement types are ATTRIBUTE and RELATION. ATTRIBUTE statements query literal properties of objects in the ontology, whereas RELATION statements match properties of objects whose range are again objects (for an example see Figure 3). As such, any PQL query can be mapped to a standardized RDF-QL statement. Consequently, none of our findings are limited to the Process Handbook and PQL. They apply analogously to other semantic web query languages such as, for instance, SquishQL [Miller et al. 2002].

Full-text and Keywords	PQL
"Find all processes that sell books over the internet." *Keywords:* "sell book internet"	(ATTRIBUTE "Name" OF ?process INCLUDES "sell") ∧ (ATTRIBUTE "Name" OF ?process INCLUDES "book") ∧ (RELATION ?process USES-MECHANISM ?mechanism) ∧ (ATTRIBUTE "Name" OF ?mechanism INCLUDES "internet")

Figure 3: An example full-text query with its corresponding keywords and derived PQL query

In order to translate the DRS generated by APE into PQL queries, we developed rewriting rules for the typical DRS structures. Each 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 RELATION statement. 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. an ATTRIBUTE statement.

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 specialization" in the query "Which process has a specialization?" is identified as the ontology-model relationship HAS-SPECIALIZATION and, hence, translated into the following PQL statement:

RELATION ?process HAS-SPECIALIZATION ?specialization

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.

Elements of the DRS not handled by the ontology-model specific keyword rules are passed to the *general-vocabulary rules.* Simple sentence structures, i.e., sentence structures not containing relative sentences, adverbs, or prepositional phrases, can now be interpreted as simple literal values. For example, the verb "sell" in a query like "How does somebody sell consumer electronics?" is represented in the DRS as "predicate(D,event,sell,A,C)". It is treated as a literal value and translated into:

ATTRIBUTE "Name" OF ?process INCLUDES "sell"

Complex structures initiate a search in the ontology-model for corresponding relationships. As an example, consider the query "How does somebody sell consumer electronics over the internet?" Here, the prepositional phrase "over the internet" indicates that a good is sold using the internet as an instrument, which is noted in the sentence's DRS (see also Example 2). As instruments, or rather their synonym

"mechanisms", are included in the Process Handbook ontology-model as the USES-MECHANISM relationship or property, we can translate the phrase "over the internet" into the following PQL statement:

RELATION ?process USES-MECHANISM ?mechanism
ATTRIBUTE "Name" OF ?mechanism INCLUDES "internet"

If the search in the ontology-model results in no corresponding relationships, then the structure is reduced to a simple structure by treating the modifiers as literals resulting in an ATTRIBUTE statement.
A full discussion of all rewrite rules is beyond the space limitations of this paper. Even so, we will try to convey the extent of the rules discussing three realistic query examples. For each example we show the ACE query, its DRS generated by APE, and the resulting PQL query. Example 1 shows the application of the simple general-vocabulary rules. Here the lexical elements "consumer", "electronics", and "sell" are treated as literal values. Note that the rewriting framework splits the compound "consumer electronics" into its constituents to improve recall. Example 2 illustrates the combination of simple and complex structures treated by the general-vocabulary rules. Finally, Example 3 uses a combination of ontology-model specific keyword rules and both types of general-vocabulary rules. Here "Which sales process…" results in the first two statements of the PQL query, which can be interpreted as "Find all processes which are sales processes and which have a subtask that…" Note that the straightforward ontology-based translation of ACE queries to PQL queries allows the user to directly grasp the system-inherent logic rather then having the system "guess" the user's intention based on some heuristics.

ACE	DRS	PQL
How does somebody sell consumer electronics?	A B C D structure(A, dom) object(C, **consumer_electronic**, object) structure(C, atomic) quantity(C, cardinality, count_unit, B, eq, 1) predicate(D, event, **sell**, A, C) modifier(D, manner, none, **how**) query(D, **how**)	(ATTRIBUTE "Name" OF ?process INCLUDES "sell") ∧ (ATTRIBUTE "Name" OF ?process INCLUDES "consumer") ∧ (ATTRIBUTE "Name" OF ?process INCLUDES "electronic")

Example 1: Transformation of "How does somebody sell consumer electronics?"

ACE	DRS	PQL
How does somebody sell consumer electronics over the internet?	A B C D structure(A, dom) object(B, **consumer_electronic**, object) predicate(C, event, **sell**, A, B) object(D, **internet**, object) modifier(C, instrument, **over**, D) modifier(C, manner, none, **how**) query(C, **how**)	(ATTRIBUTE "Name" OF ?process INCLUDES "sell") ∧ (ATTRIBUTE "Name" OF ?process INCLUDES "consumer") ∧ (ATTRIBUTE "Name" OF ?process INCLUDES "electronic") ∧ (RELATION ?process **USES-MECHANISM** ?mechanism) ∧ (ATTRIBUTE "Name" OF ?mechanism INCLUDES "internet")

Example 2: Transformation of "How does somebody sell consumer electronics over the internet?" [2]

ACE	DRS	PQL
Which sales process informs its customers over the internet?	A B C D 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)	(ATTRIBUTE "Name" OF ?process INCLUDES "sale") ∧ (RELATION ?process **HAS-PART** ?part) ∧ (ATTRIBUTE "Name" OF ?part INCLUDES "inform") ∧ (ATTRIBUTE "Name" OF ?part INCLUDES "customer") ∧ (RELATION ?part USES-MECHANISM ?mechanism) ∧ (ATTRIBUTE "Name" OF ?mechanism INCLUDES "internet")

Example 3: Transformation of "Which sales process informs its customers over the internet?"

[2] Unprocessed DRS conditions such as *structure* and *quantity* are omitted in all further DRSs to improve readability.

4. Validation – Query Performance of a Non-trivial Example

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 Figure 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 PQL queries. These are then evaluated by the query engine that passes the result back to the user interface.

Using the prototype we executed a number of real-world queries – including all examples in this paper – and compared its retrieval performance with two keyword-based retrieval approaches: one using a TFIDF-style ranking [Salton et al. 1983], 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.

ACE	DRS	PQL
Which sales process informs its customers over the internet and avoids unwanted solicitations with an opt-out list?	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)	(ATTRIBUTE “Name” OF ?process INCLUDES “sale”) ∧ (RELATION ?process HAS-PART ?part) ∧ (ATTRIBUTE “Name” OF ?part INCLUDES “inform”) ∧ (ATTRIBUTE “Name” OF ?part INCLUDES “customer”) ∧ (RELATION ?part USES-MECHANISM ?mechanism) ∧ (ATTRIBUTE “Name” OF ?mechanism INCLUDES “internet”) ∧ (RELATION ?part HAS-EXCEPTION ?exception) ∧ (ATTRIBUTE “Name” OF ?exception INCLUDES “unwanted”) ∧ (ATTRIBUTE “Name” OF ?exception INCLUDES “solicitation”) ∧ (RELATION ?exception IS-AVOIDED-BY ?handler) ∧ (ATTRIBUTE “Name” OF ?handler INCLUDES “opt-out”) ∧ (ATTRIBUTE “Name” OF ?handler INCLUDES “list”)

Example 4: Transformation of "Which sales process informs its customers over the internet and avoids unwanted solicitations with an opt-out list?"

For the non-trivial query presented in Example 4 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. 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.

5. Limitations of Our Approach, Future Research, and Related Work

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 [Wojcik 2004] – has shown that learning a controlled language is much easier than learning logic, and takes only a couple of days for the basics and 4-6 weeks for full proficiency. Furthermore, some researchers are currently developing query interfaces that will help people to write correct controlled English sentences by guiding them as they write [Schwitter et al. 2004].

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, and we believe that the rules could be automatically generated based on the ontology structure.

Last but not least, the exemplary evaluation shown in this paper is clearly limited and can only provide an idea of the potential of this approach. Consequently, the approach needs to be thoroughly evaluated. This evaluation should include giving people retrieval tasks and comparing their performance using our front-end with respect to other semantic web query tools based on plain logic, query by example, etc.

Furthermore, we would have to investigate how people's retrieval performance is related to their background.

We did not find any other application of controlled natural language querying of semantic web content. Furthermore, we found that work on natural language interfaces to data bases (not ontologized knowledge bases) has largely tapered off since the 80's [Androutsopoulos et al. 1995], even though the need for them has become increasingly acute. The most closely related work we found is the PRECISE project [Popescu et al. 2003] 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. Obviously, our front-end will not return any useful answers when none can be found in the ontology. It will, however, be able to generate an appropriate PQL statement. We hope to be able to include an empirical comparison between the two approaches in our future work.

The approach presented in this paper is clearly in its infancy. While ACE has been under development for many years, the ontology-based transformation rules are very new. Nevertheless, we believe that people's familiarity with natural languages might be the key to simplify their interaction with vast ontologies and that our approach, therefore, has the promise to provide an important step in bridging the gap between the semantic web and its users.

Acknowledgements

The authors would like to thank the MIT Process Handbook project for making available the data on which the evaluation is based, Stefan Höfler, and the anonymous reviewers for their helpful comments. This work was partially supported by the Swiss National Science Foundation (200021-100149/1).

References

Androutsopoulos, I., Ritchie, G.D., and Thanisch, P. "Natural Language Interfaces to Databases - An Introduction," *Natural Language Engineering* (1:1) 1995, pp 29-81.

Bonin, J. von, "From Discourse Representation Structures to Process Query Language - A Controlled Natural Language Front-end to the Process Handbook," unpublished Diploma Thesis, Department of Informatics, University of Zurich, 2004.

Fuchs, N.E. et al., Attempto Controlled English (ACE), www.ifi.unizh.ch/attempto, 2003.

Fuchs, N.E., Höfler, S., Schneider, G., and Schwertel, U. "Discourse Representation Structures of ACE 4 Sentences," IfI-2004, Technical Report, Department of Informatics, University of Zurich, 2004.

Kamp, H., and Reyle, U. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language*, Kluwer, Dordrecht, Boston, London, 1993.

Klein, M., and Bernstein, A. "Towards High-Precision Service Retrieval," *IEEE Internet Computing* (8:1), January 2004, pp 30-36.

Malone, T.W., Crowston, K., Lee, J., Pentland, B., Dellarocas, C., Wyner, G., Quimby, J., Osborn, C., Bernstein, A., Herman, G., Klein, M., and O'Donnell, E. "Tools for inventing organizations: Toward a handbook of organizational processes," *Management Science* (45:3) 1999, pp 425-443.

Miller, L., Seaborne, A., and Reggiori, A. "Three Implementations of SquishQL, a Simple RDF Query Language," The International Semantic Web Conference, Sardinia, Italy, 2002, pp. 423-435.

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, pp. 149-157.

Salton, G., McGill, M.J. *Introduction to modern information retrieval*, McGraw-Hill, New York, 1983.

Schwitter, R., and Tilbrook, M. "Dynamic Semantics at Work," JSAI, Kanazawa, Japan, 2004.

Spoerri, A. "InfoCrystal: A visual tool for information retrieval management," Second International Conference on Information and Knowledge Management, Washington, D.C., 1993, pp. 11-20.

Wojcik, R.H., Personal Communication, 2004 (Richard H. Wojcik is Manager of the Boeing Simplified English Project).