ORAKEL: A Natural Language Interface to an F-Logic Knowledge Base

Philipp Cimiano

Institute AIFB, University of Karlsruhe

Abstract. In this paper we present ORAKEL, a natural language interface which translates wh-questions into logical queries and evaluates them with respect to a given knowledge base. For this purpose, ORAKEL makes use of a compositional approach in order to construct the semantics of a wh-question. The system is in principle able to deal with arbitrary logical languages and knowledge representation paradigms, i.e. relational models, frame-based models, etc. However, in this paper we present a concrete implementation based on F-Logic and Ontobroker as underlying inference engine.

1 Introduction

For many applications it is desirable to query knowledge stored in a database or knowledge base through a natural language interface. Actually, this is an important research problem which has received special attention in the mid 80's (see [1] or [6] for two good surveys of the field). Certainly it is possible to query a knowledge base by using some logical query language, but it is not feasible to assume that non-computer-scientists will find such a language intuitive to use. Another option is to make use of boolean queries such as those used in WWW query interfaces as for example Google[1] or Altavista[2]. This sort of queries are certainly much more intuitive than logical ones, but suffer from a very reduced expressiveness. In this sense the challenge is to use an expressive logical query language in the background while at the same time hiding the complexity of such a language to the user by allowing him to formulate queries in natural language. In this paper we present ORAKEL, a natural language interface for a knowledge base which implements a compositional semantics approach in order to translate wh-questions into logical form (compare [3]). In particular, motivated by the growing importance of object-oriented database systems, we present a translation into F(rame)-logic [12]. F-Logic is a fully-fledged first order logic with a model-theoretic semantics. The logic was originally defined to account for the logical properties of object-oriented systems such as frames, inheritance etc. As underlying F-Logic inference engine we make use of Ontobroker [7].
The remainder of this paper is organized as follows: Section 2 presents the architecture of the system and describes its main components. In Section 3 we

[1] http://www.google.de
[2] http://www.altavista.com

show some results of the lexicon generation component. Section 4 concludes the paper.

2 System Architecture and Main Components

The main features of ORAKEL are on the one hand that it makes use of a compositional semantic construction approach as in the JANUS question answering system ([10]) thus being able to handle questions involving quantification, conjunction and negation in a classical way. On the other hand, ORAKEL automatically generates the lexicon needed to interpret the wh-questions from the knowledge base itself. In this respect it differs from earlier systems in which either general-purpose lexicons were used ([2]), developed by the interface engineer or database expert with support of tools as in TEAM ([9]) or developed and incrementally refined through interaction with the users as in RENDEZVOUS ([5]). The architecture of ORAKEL is depicted in Figure 1. In brief, the user asks a wh-question which is parsed by the *parsing and semantic construction component* which uses the *general* and *domain* lexicons as resources. The latter is automatically generated out of the F-Logic knowledge base by a *lexicon generation component.* The resulting F-Logic query is sent to the *Ontobroker* inference engine ([7]) which evaluates the query and directs the answer to the user. The two core components, i.e. the parsing and semantic construction as well as the lexicon generation component are described in the following sections.

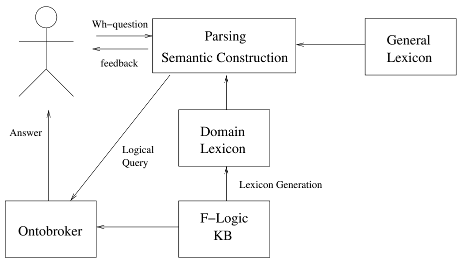

Fig. 1. System Architecture

2.1 Parsing and Semantic Construction

In order to translate wh-questions into F-logic queries we make use of a compositional semantic construction approach presented in [3]. As underlying syntactic theory we build on a variant of *Lexicalized Tree Adjoining Grammar* (LTAG) [11] introduced in [13]. LTAG is especially interesting in this context because of its extended domain of locality and thus the natural way in which subcategorization is treated. However, we extend the formalism to also include ontological information (compare [4]). For further details about the approach to semantic construction the reader is referred to [3]; in this paper we focus in particular on the system's lexicon generation component.

2.2 Lexicon Generation

An important question for any natural language interface and in general any syntax-semantics interface is where the necessary lexical entries come from. ORAKEL is novel in this respect in the sense that it automatically generates the lexicon - i.e. the elementary trees - out of the knowledge-base in question. First of all, it is important to mention that the parser makes use of two different lexicons: the *general lexicon* and the *domain lexicon* (compare Figure 1). The general lexicon includes closed-class words such as determiners i.e. *the, a, every*, etc., as well as question pronouns, i.e. *who, what, which, where*, etc. and thus is domain independent. The domain lexicon varies from application to application and is generated out of the knowledge base. For this purpose, ORAKEL makes use of subcategorization information automatically acquired from a big corpus, in our case the British National Corpus (BNC). In particular, we parse the corpus with LoPar ([15]), a statistical left-corner parser and extract the following syntactic frame types: intransitive, transitive, intransitive+PP, transitive + PP for verbs and N+PP and N+PP+PP for nouns. For each verb or noun and for each frame, we then take the synset from WordNet ([8]) which best generalizes the selectional preferences at each argument position in line with [14].
At a second step, we then take each method name in the knowledge base and look for a subcategorization frame with the same arity. We then check if the concepts are compatible, i.e. if there is a mapping M from the arguments in the subcategorization frame to the ones of the F-Logic method signature such that the concepts specified in the method are more special than the ones in the subcategorization frame with regard to the WordNet lexical hierarchy. Out of these compatible subcategorization frames we choose the one maximizing the product $\frac{1}{\sum_{(i,j)\in M} \Delta_{WN}(c(i),c(j))} \times p(s_v|v)$, where $(i,j) \in M$ says that position i of the subcategorization frame has been mapped to position j of the method signature and $c(i)$ is the value of the concept at position i in the method and $c(j)$ is the ID of the WordNet synset best generalizing the argument at position j in the subcategorization frame. Further, $\Delta_{WN}(c(i),c(j))$ is then the distance with regard to WordNet between the two concepts and $p(s_v|v)$ is simply the conditional probability of the frame s_v given the predicate (verb) v. In fact, there are different syntactic frames with an appropriate arity for the method $own[person \Rightarrow company]$ for example, so we take the one with the mapping maximizing the above product, i.e. the transitive use of *own* in this case. The synset IDs of the subject and object position both refer to the synset to which *entity* and *something* belongs. As *person* and *company* are both hyponyms of this synset, it is a valid syntactic frame according to our method. In this particular case, the overall distance remains the same independently of which frame argument is mapped to which method argument, so that we keep the argument order thus mapping the subject to the first and the object to the second position of the *own*-method. Then, we automatically generate the corresponding elementary trees to be able to ask for the subject as in *Who owns every company?*, the object as in *What/Which company does Bill Gates own?* as well as both, i.e. *Who owns what?*. In addition, we also generate the passive forms of *own* such that

dp^+:company $\sigma = \lambda P.P(microsoft)$ — microsoft

np^+:company $\sigma = \lambda x.company(x)$ — company

np^+:organization $\sigma = \lambda x.organization(x)$ — organization

Fig. 2. Elementary trees in the domain lexicon

we can also ask: *What/Which company is owned by Bill Gates?* or *What/Which company is owned by whom?*. For this purpose, we look up the derived forms of *own* (past participle, 3rd person singular present tense) in the lexicon provided with LoPar.

Additionally, for the method names we look up the WordNet-synonyms of the most frequent sense and repeat the same procedure as above for the synonyms in WordNet. For *own*, WordNet contains two synonyms in the most frequent sense, i.e. *have* and *possess*. As expected, the corresponding subcategorization frames are quite similar to the ones of *own*, such that we automatically create similar elementary trees as above for *have* and *posses* mapping to the *own*-method.

Though the mapping as well as generation of elementary trees for method names corresponding to a noun - for example *boss* - works basically along these lines, there is one further complication to take into account. In fact, for such type of methods, there will always be one argument position which is not realized in a noun+PP or noun+PP+PP frame as it is the argument which is typically specified in a copula construction such as *The boss of Microsoft is Bill Gates*. Thus, if the method name corresponds to a noun as in $company[boss \Rightarrow person]$, we search for compatible frames without considering the last method argument and choose the mapping maximizing the above product. Take for example the *boss* method. As synonyms in WordNet we find: *foreman, chief, gaffer, honcho*. If we then apply our approach we map the subcategorization frames boss(of:100001740) and chief(of:100017954) to the *boss*-method, thus being able to ask *Who is the boss/chief of Microsoft?*. It is important to mention that it can not be assumed that all method names are so 'pure', so that in case the method name does not match a verb or noun we have subcategorization information for, we first substitute punctuation symbols from the method name by spaces, i.e. *wine-region* becomes *wine region* and we consult the subcategorization lexicon again. In case this also doesn't produce any result, we then take the word at the end of the expression, *wine* in our case and consult the lexicon again.

Finally, we derive the entities and np's from membership/subclass statements such as microsoft:company or company::organization thus yielding the elementary trees depicted in Figure 2.

3 Evaluation

We conducted a small experiment to assess the quality of our lexicon generation component. For this purpose, we selected 5 rather general ontologies from the

Ontology	#Properties	Domain+Range	Non-Composite	Correct	% Correct
Beer	9	4	3	2	66.67%
Wines	10	10	9	6	44.44%
Personal	25	10	8	6	75%
General	28	17	6	6	100%
University	27	11	2	1	50%
Total	99	42	28	21	75%

Fig. 3. Results of the Lexicon Generation Component Evaluation

DAML Ontology Library[3] about the following topics: *beer*[4], *wine*[5], *general information about organizations*[6], *personal information*[7] and *university activities*[8]. From all these ontologies we took the properties (binary relations) and tested if we could find an appropriate subcategorization frame with our lexicon generation component from which to generate appropriate elementary trees. Table 3 gives the following figures: (1) the ontology in question, (2) the number of DAML properties in the ontology, (3) the number of properties in the ontology with a *domain* and a *range* specified, (4) the number of properties from (3) with a non-composite name, (5) the number of the relations from (4) for which our method found a correct subcategorization frame and (6) the percentage of correct answers for the properties in (4). The results show that for 3/4 of the properties in (4) we get correct subcategorization frames. Of course, these results are limited as we have only tested binary properties. It is also clear that the approach needs to be extended to also handle properties with composite names to achieve better results. Nevertheless, these first results are encouraging.

4 Conclusion and Outlook

We have presented ORAKEL, a natural language interface to an F-Logic knowledge base which makes use of a compositional semantics approach to translate wh-questions into F-Logic queries. For this purpose, we have assumed the declarative formulation of LTAG in [13] as well as the extension in [4] as underlying syntactic theory. The approach has been implemented in Java as a parser operationalizing the calculus in [13] and taking into account ontological information as described in [4]. Further, we have presented in detail and evaluated the system's lexicon generation component.

We are currently devising several experiments to evaluate our system. First we

[3] http://www.daml.org/ontologies/

[4] http://www.cs.umd.edu/projects/plus/DAML/onts/beer1.0.daml

[5] http://ontolingua.stanford.edu/doc/chimaera/ontologies/wines.daml

[6] http://www.cs.umd.edu/projects/plus/DAML/onts/general1.0.daml

[7] http://www.cs.umd.edu/projects/plus/DAML/onts/personal1.0.daml

[8] http://www.cs.umd.edu/projects/plus/DAML/onts/univ1.0.daml

intend to acquire typical wh-questions in a Wizard-of-Oz style experiment, manually translate them into F-Logic queries and then evaluate our system in terms of precision/recall with regard to them. This will show how good our system is in dealing with typical questions. Second, we intend to evaluate the usability of the system by comparing the number of successful/failed questions, elapsed time etc. between people asking wh-questions and people directly formulating logical queries to the Ontobroker system.

References

1. I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. Natural language interfaces to databases–an introduction. *Journal of Language Engineering*, 1(1):29–81, 1995.
2. B.K. Boguraev and K. Sparck Jones. How to drive a database front end to databases with evaluative feedback. In *Proceedings of the Conference on Applied Natural Language Processing*, 1983.
3. P. Cimiano. Translating wh-questions into f-logic queries. In R. Bernardi and M. Moortgat, editors, *Proceedings of the CoLogNET-ElsNET Workshop on Questions and Answers: Theoretical and Applied Perspectives*, pages 130–137, 2003.
4. P. Cimiano and U. Reyle. Ontology-based semantic construction, underspecification and disambiguation. In *Proceedings of the Prospects and Advances in the Syntax-Semantic Interface Workshop*, 2003.
5. E.F. Codd. Seven steps to RENDEZVOUS with the casual user. In J. Kimbie and K. Koffeman, editors, *Data Base Management*. North-Holland publishers, 1974.
6. A. Copestake and K. Sparck Jones. Natural language interfaces to databases. *Knowledge Engineering Review*, 1989. Special Issue in the Applications of Natural Language Processing Techniques.
7. S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In *Database Semantics: Semantic Issues in Multimedia Systems*, pages 351–369. Kluwer, 1999.
8. C. Fellbaum. *WordNet, an electronic lexical database*. MIT Press, 1998.
9. B.J. Grosz, D.E. Appelt, P.A. Martin, and F.C.N. Pereira. Team: An experiment in the design of transportable natural language interfaces. *Artificial Intelligence*, 32:173–243, 1987.
10. EW Hinrichs. The syntax and semantic of the janus semantic interpretation language. Technical Report 6652, BBN Laboratories, 1987.
11. A.K. Joshi and Y. Schabes. Tree-adjoining grammars. In *Handbook of Formal Languages*, volume 3, pages 69–124. Springer, 1997.
12. M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the Association for Computing Machinery*, May 1995.
13. Reinhard Muskens. Talking about trees and truth-conditions. *Journal of Logic, Language and Information*, 10(4):417–455, 2001.
14. Philip Resnik. Selectional preference and sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, 1997.
15. Helmut Schmid. Lopar: Design and implementation. In *Arbeitspapiere des Sonderforschungsbereiches 340*, number 149. 2000.