Ontology-Based Free-Form Query Processing for the Semantic Web

A Thesis Proposal
Presented to the
Department of Computer Science
Brigham Young University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

By Mark Vickers
December 2005

1. Introduction

With estimates of more than 11.5 billion indexable pages [GS05], the web is an incredibly rich source of information. Google [GOO] alone claims to have had 8 billion web pages indexed in 2004 [Sul05]. The challenge of harnessing this information, making it easier to search and query, has been the focus of much research. While great strides have been made towards this goal, searching is still an application with significant room for improvement [GMM03].

A proposed framework, known as the semantic web, promises to significantly enhance web querying. Where the current web contains information that is human readable, the semantic web extends the information to be machine-readable as well. To achieve this goal, the semantic web relies heavily on the use of ontologies. An ontology is a formal, explicit specification of a conceptualization [Gru93]. Semantic web search programs will "look for … pages that refer to a precise concept instead of all the ones using ambiguous keywords" [BHL01].

Even with the upcoming semantic web framework, the details of how humans are to query the semantic web are unclear. Two major issues to consider for this problem are the usability of the interface, and the effectiveness of the query processing. The user interface should require a minimal learning curve yet still allow complex queries. Queries should be processed in a way that takes advantage of semantic content on the web, aiming at interpreting a query's meaning instead of viewing it as a set of keywords. Whether this ideal can be achieved remains unclear.

The proposed thesis introduces a system that will be called AskOntos. AskOntos uses a novel approach to query processing that contributes to the realization of enhanced searching on the semantic web. The approach relies on extraction ontologies, which are ontologies capable of extracting and structuring relevant information from unstructured documents. Figure 1 gives a high level view of AskOntos's process flow. AskOntos applies information extraction (IE) to a natural language (NL) query via extraction ontologies found in a repository, and then chooses the ontology that provides the best conceptual context for the query. With the extracted information from the query and the chosen ontology, AskOntos formulates a query which it runs over values that have been extracted from web pages by the chosen ontology, resulting in the query answer. AskOntos

returns database-like tables of extracted values; each tuple in the table contains a link to the document from which the original information was extracted. Because of its use of IE, AskOntos offers two significant benefits: 1) it converts free-form queries into structured queries using ontologies, and 2) answers queries with extracted values. In the initial version being proposed here, AskOntos will only be expected to handle conjunctive queries—queries where atomic conditions must all hold.

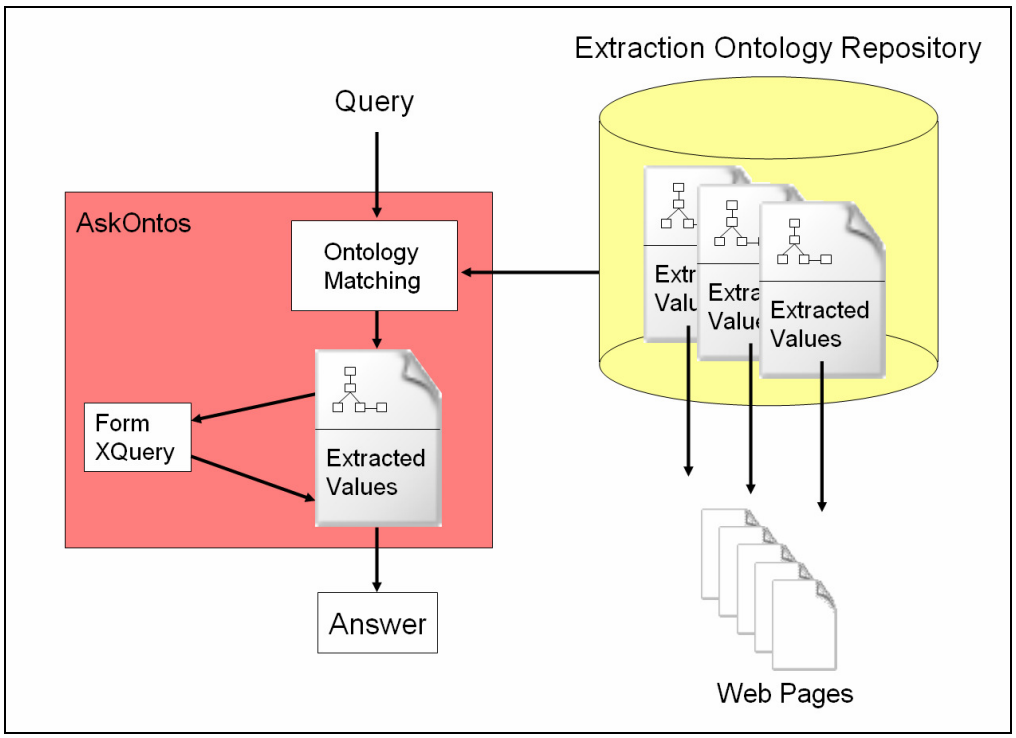

Figure 1. A high level view of AskOntos's process flow.

Works from several other research efforts are similar to AskOntos in that they process queries over semantic web pages. One system, QUEST [BKK+99], facilitates expressing complex queries on the semantic web through a graphical query language. Their interface to the semantic web consists of a semantic view (an ontology-like graph), and a visual view (HTML). QUEST users choose ontological categories and express constraints. Query results come in the form of generated documents and graphs. The graphical interface has many advantages, but requires the user to be familiar with underlying graph structures. AskOntos is similar in that it relies heavily on the structure of ontologies, but differs in that it has a natural language interface. A natural language interface requires no system knowledge from the user and, therefore, has no learning curve.

As with QUEST, the SHOE [HH00] approach has the user enter a query by interacting directly with ontologies. The interface is form-based rather than graph-based. The user can drill down an ontology structure and set constraints through pull-down menus and text areas. The SHOE approach searches only web pages annotated by SHOE, but allows the users to optionally submit keyword queries to a popular information retrieval (IR) style search engine. While SHOE shares AskOntos's ability to return answers in tabular form, their interfaces differ.

The authors of [BKG+05] share the idea of making a natural language front end for semantic web queries. The natural language they propose, however, is limited to a subset of English called Attempto Controlled English (ACE). The system translates ACE queries into discourse representation structures (DRS). DRS terms match against ontology keywords and relations to form a process query language (PQL) statement [KB04], which queries an ontology. While using ACE helps overcome some major natural language processing (NLP) pitfalls, it does require the user to learn the rules of the ACE language.

AQUA [VM04] is an ontology-driven question answering system with a natural language interface that uses computational linguistics, logic, question classification, and IR technologies. A single, large ontology (domain ontology) assists the query processor in generating logical formulas as Query Logic Language (QLL) statements. AQUA validates the generated QLL relations by running a similarity algorithm, which matches relational words in the query to ontology relations. After being transformed to a Prolog-style syntax, QLL statements run over the ontology. If the proof over the knowledge base fails, the query gets sent to a backup IR system.

One difference between AQUA and AskOntos is the query target. AQUA's target is a company intra-net, where a single domain ontology is used. AskOntos's target is the semantic web, where many unrelated documents are described by a variety of ontologies. Another difference between the two systems is that AQUA segments the query into subject, verb, prepositional phrases, adjectives and objects. AskOntos does not attempt to recognize any parts of speech or syntactically motivated phrases from the query. AQUA is also different from AskOntos in that AQUA returns extracted *passages* from documents, while AskOntos returns extracts *values* from documents.

Ever since the early sixties researchers have been working on building natural language interfaces to databases (NLIDBs) [CJ90]. The ideal NLIDB system would appropriately interpret an unrestricted natural language query from an inexperienced user. Due to many problems and limitations in natural language parsing technology [CCJ04], state of the art systems are far from meeting these lofty goals. A typical NLIDB architecture has an analyzer and a translator. Using a generalized grammar, lexicon, and domain knowledge the analyzer does syntactic and semantic processing on the input query, converting it into a logical, intermediate representation. The translator takes the intermediate representation and does task-specific and pragmatic processing, producing a database query.

While both AskOntos and NLIDB systems take a single natural language query as input and produce a formal query, the former to run over XML and the latter generally for a relational database, there are some important distinctions between them. One difference between the systems lies in the initial grammar-based processing. NLIDBs use generalized grammars to do syntactic analysis, which deal with the input's structural form, and try (often not very successfully) to handle fragmented and ill-formed sentences. AskOntos has domain specific grammars (in the form of regular expressions) encoded in each extraction ontology and makes no structural analysis and therefore handles fragmented and ill-formed sentences with ease. Another difference between the systems is in how new domains are introduced. Porting an NLDB to a new domain typically requires low-level system expertise as well as the know-how to modify lexicons, grammars, and task-specific processing done in the NLIDB's translator module. While the AskOntos engine itself is domain independent, extraction ontologies must be carefully designed and created for each domain. It is not trivial to add or change domains with either system, but AskOntos appears to require less computer knowledge and training. Another difference between NLIDBs and AskOntos is that NLIDBs do pragmatic and task-specific processing after the query has been syntactically and semantically processed and converted into an intermediate form, while the pragmatic and task-specific processing in AskOntos is done immediately as extraction ontologies extract from the input query.

2. Thesis Statement

From the onset of the semantic web, the problem of making semantic content effectively searchable for the general public emerges. Demanding an understanding of ontologies, or

familiarity with a new query language would likely frustrate semantic web users and prevent any widespread success of such a system. Given this need, the hypothesis of the proposed thesis is that

> *we can build a system that will show how extraction ontologies can be used to execute conjunctive, free-form queries over semantically annotated web pages.*

If successful, as measured by the system's ability to translate informal natural language queries into formal queries over semantic web pages, the proposed technique will not only eliminate a learning curve, but will effectively use semantic annotation to help realize the vision of the semantic web.

3. Methods

This section describes the proposed query processing in detail. Listed below are the four major steps to the process.

1. Parse Query
2. Find Corresponding Ontology
3. Formulate XQuery Expression
4. Run XQuery Expression over Ontology's Extracted Data

Before giving a detailed explanation of each of these steps, section 3.1 introduces (in greater detail) extraction ontologies, which are a fundamental component of the approach. Following the introduction to extraction ontologies are the four processing steps comprising sections 3.2 through 3.5.

3.1. Introduction to Extraction Ontologies

An extraction ontology is a type of conceptual-model capable of performing domain-specific information extraction over plain text. It consists mainly of object sets, relationship sets, and constraints. Figure 2 shows an example of an extraction ontology for a car, in graphical form. The boxes are object sets and represent a collection of instances. Object sets drawn with dashed lines are lexical object sets, meaning the values in the sets can be expressed textually, while object sets with solid lines are non-lexical. The arrow with a dot in the *Car* object set denotes that it is the primary object, which means it is the

main idea being described by the ontology. Lines are relationship sets. Numbers separated by colons, such as "0:1", specify the participation constraints for an object set in relation. A "*" denotes unlimited cardinality. Black and clear triangles indicate aggregation and generalization/specialization respectively.

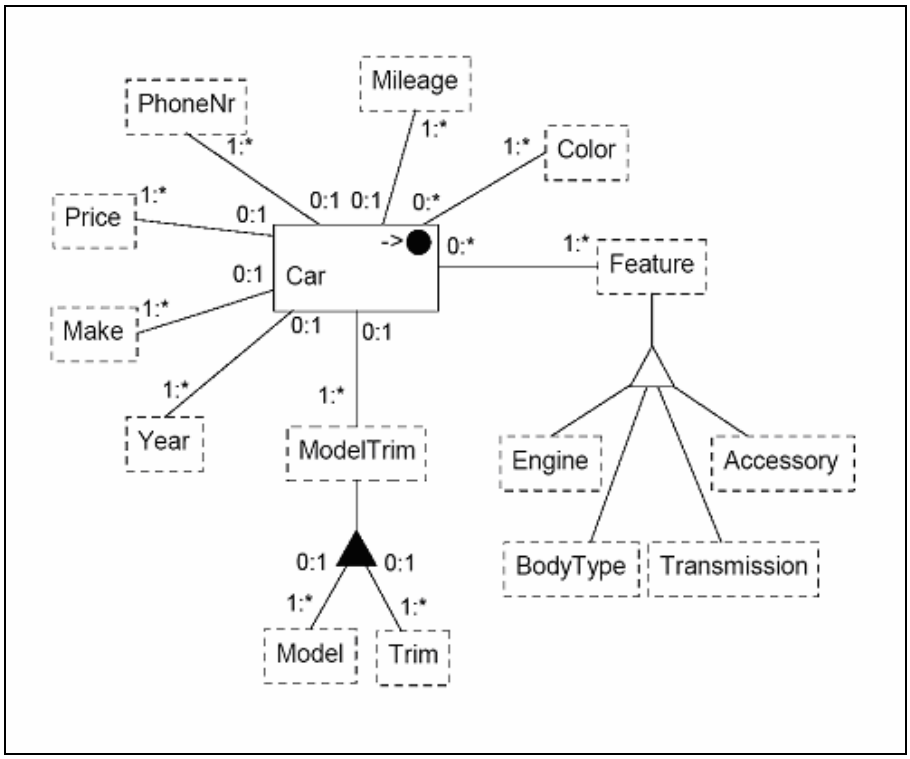

Figure 2: A simple extraction ontology describing the concept of a car.

A distinguishing feature of extraction ontologies is that each object set has an associated *data frame* [Emb80]. A data frame "[encapsulates] the essential properties of everyday data items" and is responsible for extracting values. Data frames contain value phrase recognizers, keyword phrase recognizers, operation phrase recognizers, lexicons, and conversion methods. Value phrase recognizers consist of regular expressions that allow the data frame to recognize values in plain text that belong to its associated object set. For example, a data frame for a *Price* object set may recognize "5,000", "2300", and "0.75" as potential prices. Value phrase recognizers not only have regular expressions for the values, but also for immediate left and right context and exception expressions as well. Keyword phrase recognizers differ from value phrase recognizers only in that they match text that is likely to be near values, rather than the values themselves. For example, "dollars", "$", "¢", "USD", or "price" might be found in the neighboring text of a price value, further

solidifying the interpretation of the value. Operation phrase recognizers match words that suggest some kind of operation, such as a comparison, that may be applied to values. The *Mileage* data frame, for example would have several operation recognizers, including a greater-than operator recognizer, which would recognize words such as "more than", "greater than", and "over". Data frames also have conversion methods to convert the extracted values to a canonical form.

Each extraction ontology can save its extracted values to an associated XML file. Yihong Ding, of BYU's Data Extraction Group, designed the XML syntax based on the popular semantic web ontology description language, OWL (OWL Web Ontology Language) [OWL04] that stores the extracted values.

3.2. Parse Query

Although documents are typically the target for information extraction [ECJ+99], AskOntos's first step is to perform extraction over the user query using an extraction ontology. To illustrate the process, suppose the user enters the query:

> "Find me the price and mileage of all red Nissans – I want a 1998 or newer."

Consider the car extraction ontology in Figure 2. The word "price", in the query, matches the name of the *Price* object set. Had the user used the word "cost" instead of "price", "cost" would still match with the *Price* data frame keyword phrase recognizer. As with "price", the word "mileage" matches *Mileage*. The *Color* data frame's value phrase recognizer matches "red" as a value, similarly the *Make* data frame identifies "Nissan" from the word "Nissans" as a value, and *Year* matches "1998" as a value. Lastly, the greater-than-or-equal operator recognizer in the Year data frame matches the words "or newer".

3.3. Find Corresponding Ontology

The purpose of this step is to find the ontology that best serves as a context for the concepts referred to in the query. (It is assumed that there is a large collection of ontologies.) Each ontology goes through the parsing process described in section 3.2 and receives a *similarity value* based on the number of matching query words. AskOntos chooses the ontology with the highest similarity value. The precise algorithm to calculate the similarity value will be

part of the research. One possible (and straightforward) way to compute the similarity value is to simply count the number of matches made from query words (or phrases).

To illustrate how the calculation of a similarity value might work, consider the Car extraction ontology in Figure 2 and the example query in section 3.2. From the parsing process described in section 3.2, the Car ontology has a total of six matches: "price", "mileage", "red", Nissan" "1998", and "or newer". The similarity value for Car is 6. Now consider the diamond ontology in Figure 3. The word "price" matches *Price*'s name, and "red" might match *Color*'s value phrase component, so its similarity value is 2. Since the Car ontology scores a higher similarity value, it corresponds better to the query than the Diamond ontology.

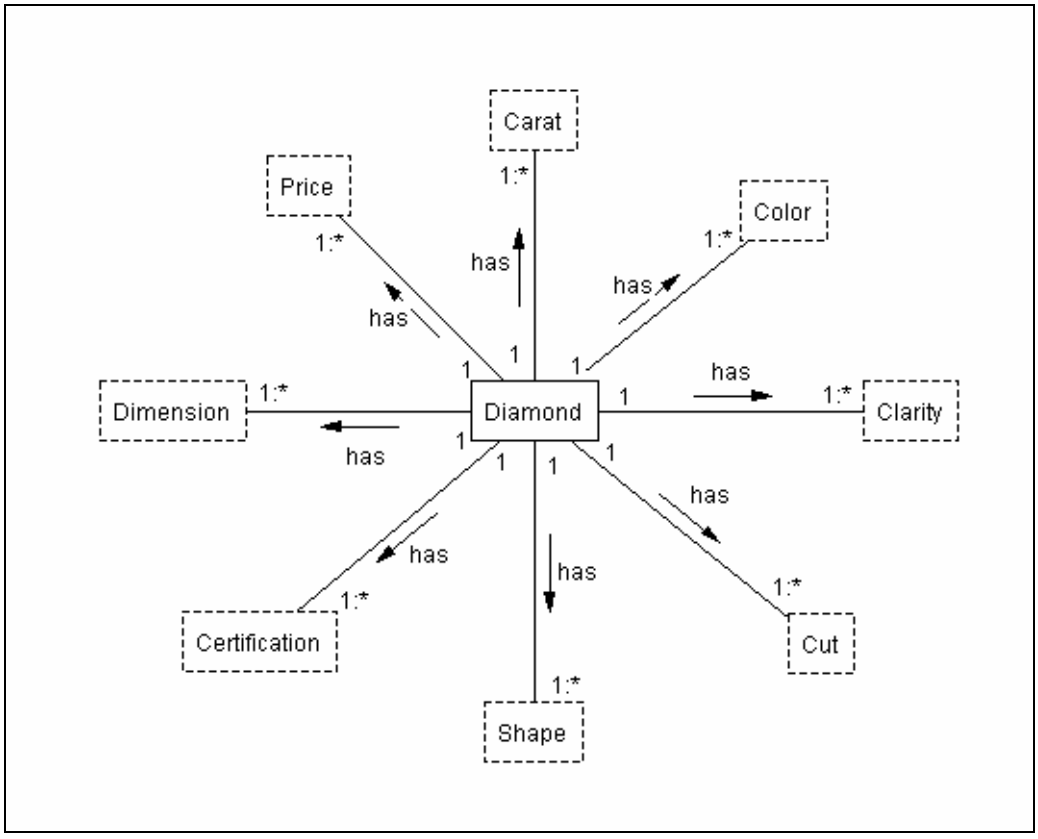

Figure 3. A simple extraction ontology depicting diamonds.

3.4. Formulate XQuery Expression

As was mentioned in the previous section, there is an assumed repository of extraction ontologies in the form of XML files. Each ontology file contains not only the ontology

description, but also a collection of instance values that have been extracted from normal web documents by that ontology. It is over these values that the XQuery expression will run. Figure 4 shows a single record of values extracted from the web by the car ontology. The `Car` element's `rdf:ID` attribute indicates that this is the seventh extracted record for this ontology. The `owl:Thing` element at the bottom shows that *Year*, *Make*, and *Color* values were extracted for this record. The `Year`, `Make`, and `Color` elements each contain their extracted values, the unique identifier assigned to those values, and an offset into a cached copy of the web page indicating the location of the extracted values.

```
<Car rdf:ID="CarIns7">
        <CarValue rdf:datatype="&xsd;string">7</CarValue>
</Car>

<Year rdf:ID="YearIns7">
        <YearValue rdf:datatype="&xsd;string">1999</YearValue>
        <ontos:URI rdf:datatype="&xsd;string">YearIns7</ontos:URI>
        <offset rdf:datatype="&xsd;nonNegativeInteger">41641</offset>
</Year>

<Make rdf:ID="MakeIns7">
        <MakeValue rdf:datatype="&xsd;string">Nissan</MakeValue>
        <ontos:URI rdf:datatype="&xsd;string">MakeIns7</ontos:URI>
        <offset rdf:datatype="&xsd;nonNegativeInteger">41893</offset>
</Make>

<Color rdf:ID="ColorIns7">
        <ColorValue rdf:datatype="&xsd;string">red</ColorValue>
        <ontos:URI rdf:datatype="&xsd;string">ColorIns7</ontos:URI>
        <offset rdf:datatype="&xsd;nonNegativeInteger">42186</offset>
</Color>

   ...

<owl:Thing rdf:about="#CarIns7">
        <hasMake    rdf:resource="#MakeIns7" />
        <hasYear    rdf:resource="#YearIns7" />
        <hasColor   rdf:resource="#ColorIns7" />
        <hasMileage rdf:resource="#MileageIns7" />
        <hasPrice   rdf:resource="#PriceIns7" />
</owl:Thing>
```

Figure 4: Snippet from the Car ontology file, showing the extracted Year, Make and Color values.

AskOntos produces conjunctive XQuery expressions, which restrict the resulting records to satisfy zero or more constraints. AskOntos forms constraints by applying Boolean operators to value-phrase matching query words and the associated object set (equality is

the default operator). For example, since the word "red" matched the value phrase of the *Color* data frame in the running example, the constraint that the color is red must hold. Similarly, the condition that the make is Nissan must hold. Since the greater-than-or-equal operator recognizer and value phrase recognizer of *Year*'s data frame matched "or newer" and "1998" respectively, the condition that the year be greater than or equal to 1998 must hold as well.

Figure 5 shows the resulting XQuery expression created from the example query, and the Car ontology. Line 1 states that the query will iterate over `rdf:RDF` elements at the URL "*`file:///c:/ontos/owlLib/Car.OWL`*", which contains the Car extraction ontology definition and extracted values (shown partially in Figure 4). Line 2 specifies an inner iteration over each `owl:Thing` element. Lines 4-9 use the XQuery `let` phrase to define and assign values to variables. For example, in line 4, the variable `$id` receives the instance number of the `owl:Thing` element by taking the substring after the "CarIns" part of the `rdf:about` attribute. In line 5, `$Color` receives the text found in the `car:ColorValue` element whose parent is an element named `car:Color` and has an `rdf:ID` attribute equal to "ColorIns" concatenated with the instance number in the variable `$id`. This assures that the returned values are grouped according to the records as they were extracted. The where clause in lines 11-13 contains the query conditions, and lines 14-20 specify the object set values to be returned. Notice that the "or empty" condition in the select portion allows records that have no color specified, for example, to be returned in the result.

3.5. Run XQuery Expression over Ontology's Extracted Data

AskOntos runs the XQuery expression over the pre-extracted data of the selected ontology, in XML form, using Qexo 1.7, a GNU implementation of an XQuery engine for Java. As is evident from lines 14-20 in Figure 5, the query returns results in the form of XML. AskOntos uses XSLT to translate the results into an HTML table. Figure 6 shows the results from the example query. The result illustrates that resulting records can come from multiple sites assuming that the ontology has extracted over multiple sites.

```
1:  for $doc in document("file:///c:/ontos/owlLib/Car.OWL")/rdf:RDF
2:  for $Record in $doc/owl:Thing
3:
4:  let $id := substring-after(xs:string($Record/@rdf:about), "CarIns")
5:  let $Color := $doc/car:Color[@rdf:ID=
                    concat("ColorIns", $id)]/car:ColorValue/text()
6:  let $Make := $doc/car:Make[@rdf:ID=
                    concat("MakeIns", $id)]/car:MakeValue/text()
7:  let $Year := $doc/car:Year[@rdf:ID=
                    concat("YearIns", $id)]/car:YearValue/text()
8:  let $Price := $doc/car:Price[@rdf:ID=
                    concat("PriceIns", $id)]/car:PriceValue/text()
9:  let $Mileage := $doc/car:Mileage[@rdf:ID=
                    concat("MileageIns", $id)]/car:MileageValue/text()
10:
11:  where($Color="red" or empty($Color)) and
12:       ($Make="Nissan" or empty($Make)) and
13:       ($Year>="1998" or empty($Year))
14:  return <Record ID="{$id}">
15:           <Price>{$Price}</Price>
16:           <Mileage>{$Mileage}</Mileage>
17:           <Color>{$Color}</Color>
18:           <Make>{$Make}</Make>
19:           <Year>{$Year}</Year>
20:     </Record>
```

Figure 5: XQuery expression derived from the example user query and the CarAd ontology.

It may be the case that a single record contains multiple values for one attribute, for example "A/C" and "sun roof" might both be Accessory values. If the list of values is reasonably small, they are displayed as a comma separated list. If the list of values is long, AskOntos will generate a button in the place of the values that, when pushed, expands to show all values.

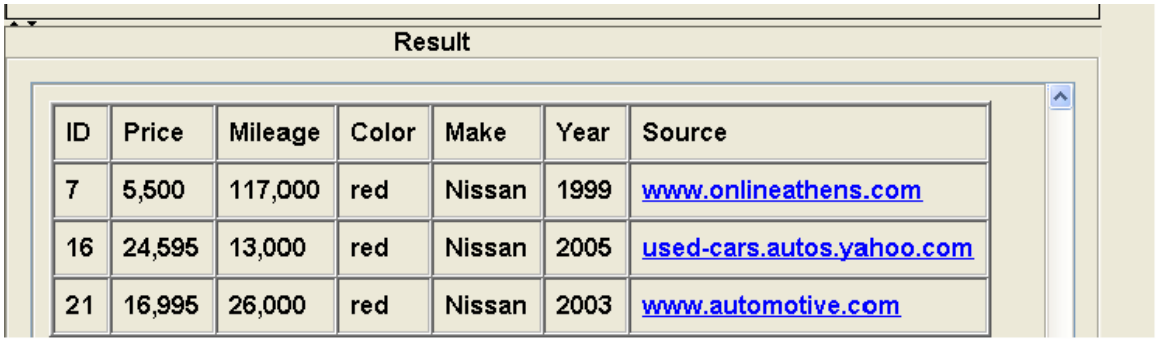

Figure 6: Results of XQuery expression for the example query.

Each record returned contains a link that points to a cached copy of the page over which extraction was originally performed. Clicking the link opens the cached page in a browser,

scrolled to the section where the record was extracted. AskOntos highlights the extracted values to help the user see the record values in their original context. Figure 7 shows the page linked to www.onlineathens.com.

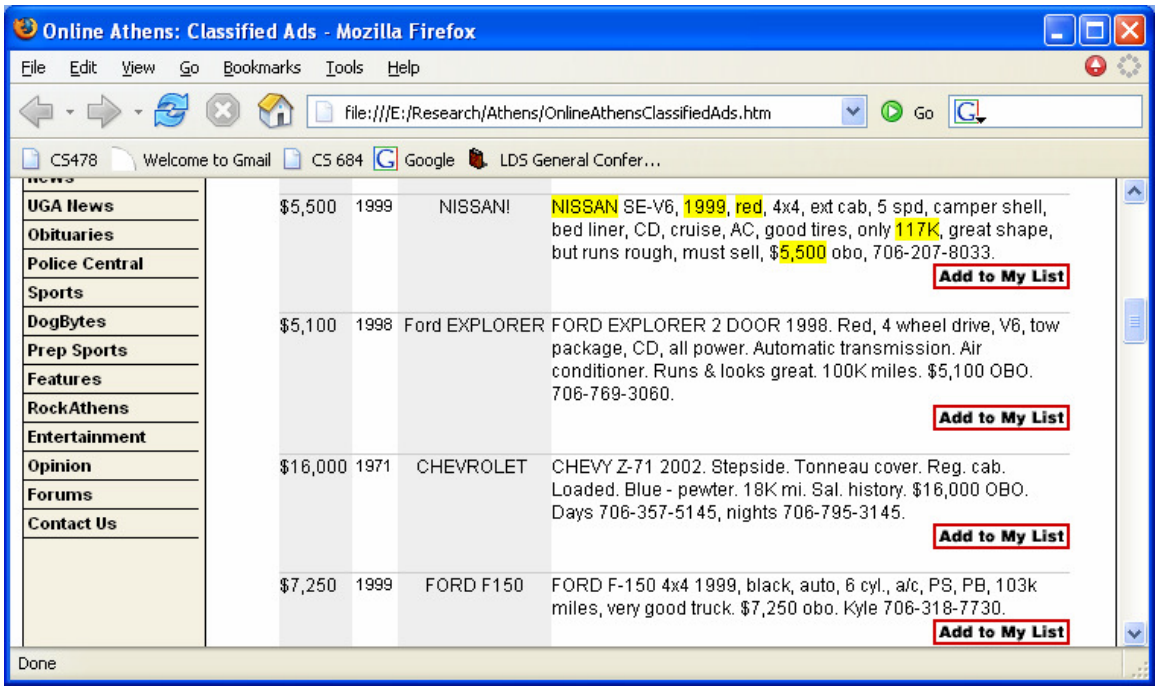

Figure 7: Capture of the web page from which the Nissan record was extracted. Query-relevant values are highlighted.

4. Evaluation of AskOntos

We plan to measure the success of AskOntos by its ability to translate informal natural language queries into formal queries over semantic web pages. AskOntos's ability to correctly match a query to an appropriate ontology is a major factor for query translation. A series of experiments will be run to measure a) how well AskOntos chooses an appropriate ontology for a given query, and b) its ability to translate a natural language query to a formal XQuery expression. It is important to note that the quality of the web extraction is not the issue of this proposal, for this reason precision and recall of the extracted results will not be included in the experiments.

The metrics for the two above-mentioned measurements will be as follows. To measure AskOntos's success at choosing the appropriate ontology, the metric will be the number of questions correctly matched to the domain divided by the total number of questions. To

automate this process, we will pre-label each test question with the domain it belongs to. To test AskOntos's ability to translate natural language queries into formal XQuery expressions, we will manually translate each test question (beforehand) into an intermediate form that indicates the selection and projection portions of the query. For example, the question "Find me the price and mileage of all red Nissans – I want a 1998 or newer" will translate to "PROJECT: {Price, Mileage, Color, Make, Year} SELECT: {(Color, =, "red"), (Make, =, "Nissan"), (Year, >=, "1998")}. AskOntos will automatically convert the XQuery expression it produces into the same intermediate form, the project values generated from the 'return' clause of the XQuery expression, and the select values generated from the 'where' clause. AskOntos will assign each portion of the intermediate query a precision and recall value. For example, if AskOntos generates an intermediate query with PROJECT values of Price, Make, Model, and Year, (notice Mileage is missing) the project portion of this query will score a precision value of 100%, and a recall value of 80%.

To perform the experiments, we will create a repository of extraction ontologies. The repository will consist of five domains, including car ads, diamonds, ski resorts, theatre schedules, and real estate. Experiments will consist of 100 natural language questions (20 for each domain). The questions will be factoid type questions, or questions similar to queries one might ask a database. Due to the limited domain of topics, we plan to collect queries from members of our research group.

5. Contribution to Computer Science

The thesis will contribute the following to the semantic web:

- Web queries that use semantic annotations
- Web queries returning extracted data
- Handling conjunctive free-form queries over ontologies

6. Delimitations of the Thesis (1/2 page)

The following will not be addressed by the proposed thesis:

- The process of storing extracted values from web pages in an XML file
- The format or creation of the ontology repository

- Refreshing extracted values when pages change
- Queries with disjunctions and negations
- Exploiting probabilistic extraction techniques for IE
- Scalability of AskOntos to the World Wide Web

7. Thesis Outline

1. Introduction and Related Work (3-5 pages)
2. Methodology (15-20 pages)
    a. Introduction to Extraction Ontologies
    b. Parse Query
    c. Find Corresponding Ontology
    d. Formulate XQuery Expression
    e. Run XQuery Expression over Extracted Data
3. Experimental Results and Analysis (5-7 pages)
4. Conclusions and Future Work (2-3 pages)

8. Thesis Schedule

Literature Search and Reading	May 2005 – October 2005
Design and Coding	November 2005 – January 2005
Experiments	February 2006
Chapter 2	February 2006 – March 2006
Chapter 3	March 2006
Chapters 1 & 4	March 2006
Thesis Revision and Defense	April 2006

9. Bibliography

[BKK+99] Z. Bar-Yossef, Y. Kanza, Y. Kogan, W. Nutt, and Y. Sagiv, "Querying Semantically Tagged Documents on the World Wide Web," In *Proceedings of the 4th Workshop on Next Generation Information Technologies and Systems* (*NGITS'99*), pages 2-19, Zikhron-Yaakov, Israel, July 1999.

The authors present a system called QUEST, which is a question answering system that queries over semantically enriched documents annotated with OHTML. The user creates a query through interacting with an ontological graph. Constraints may be entered for complex queries. QUEST shows results in the form of a graph. Ontology node values can be strings or regular expressions.

[BHL01] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, 284(5):34-43, 2001.

This reference presents a vision of the semantic web. Among other things, the authors discuss knowledge representation, ontology use, and agents. This article agrees that the semantic web will provide an environment where queries can be answered with less ambiguity.

[BKG+05] A. Bernstein, E. Kaufmann, A. Gohring, and C. Kiefer, "Query Ontologies: A Controlled English Interface for End-users," In *Proceedings of the 4th International Semantic Web Conference*, pages 112-126, Galway, Ireland, November 2005.

The authors propose making a natural language front-end to semantic web queries by limiting the natural language to subset of natural English called Attempto Controlled English (ACE). Their approach translates ACE queries into Discourse Representation Structures (DRS), a variant of the first-order logic introduced by Kamp and collaborators (1993), then into process query language (PQL). To form PQL statements, the system matches DRS structures against ontology keywords (or their morphological or syntactic variants). This approach forces the user to learn ACE, which the author indicated should take only 2 days to learn the basics and 4 to 6 weeks to be proficient. This approach has not been evaluated extensively, so the performance of the system is unknown. Results were very encouraging (but not scientifically significant because of the low number of documents involved).

[CCJ04] S. Conlon, J. Conlon, and T. James, "The Economics of Natural Language Interfaces: Natural Language Processing Technology as a Scarce Resource," *Decision Support Systems*, 38(1):141-159, October 2004.

This paper compares natural language interfaces (NLI) with competing interface approaches, including (a) other types of user-friendly interfaces, such as icon, menu, and Query By Example (QBE) systems, and (b) training of users so that they become comfortable with less user-friendly interfaces. They conclude that, since current NLI technology is still limited, considerable application-specific customization is necessary.

[CJ90] A. Copestake and K.S. Jones, "Natural Language Interfaces to Databases," *Knowledge Engineering Review*, 5(4):225-249, 1990.

This paper gives a great explanation of how natural language interfaces to databases have evolved. It also explains the architecture of a typical system

that uses a conceptual schema to enhance query conversion and improve portability.

[ECJ+99] D. Embley, D. Campbell, Y. Jiang, S. Liddle, Y. Ng, D. Quass, R. Smith, "Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages," *Data & Knowledge Engineering*, 31(3):227-251, 1999.
This paper explains how a conceptual-modeling approach to extraction can extract data from unstructured documents and structure it according to a generated database schema. This extraction targets data-rich, multiple-record documents, whereas AskOntos applies the same extraction technique over natural language queries.

[Emb80] D. Embley, "Programming with Data Frames for Everyday Data Items," In *Proceedings of the 1980 National Computer Conference*, pages 301-305, Anaheim, California, May 1980.
This paper introduces data frames. Each object set in an extraction ontology has an associated data frame. Data frames are similar to abstract data types and classes. They define value, keyword, and operator recognizers. These recognizers can identify and extract values that belong to the object set associated with the data frame.

[GOO] Google: http://google.com, October 2005.
Google is one of the most popular web based search engines. It is a powerful information retrieval system that returns documents for a given keyword based query.

[Gru93] T. R. Gruber, "A Translation Approach to Portable ontologies," *Knowledge Acquisition*, 5(2):199-220, 1999.
In this article, the author provides a complete and concise definition for ontologies. He describes ontologies as a formal, explicit specification of a conceptualization.

[GMM03] R. Guha, R. McCool, and E. Miller, "Semantic Search," In *Proceedings of the 12th International Conference on World Wide Web*, pages 700-709, Budapest, Hungary, May 2003
This paper points out that "search is both one of the most popular applications on the Web and an application with significant room for improvement." The paper introduces a system that augments traditional web searches with semantic web based results. The augmented results appear as links to the right of traditional search results. The system uses the semantic web infrastructure, TAP. The query is sent to both the Semantic Search application and Google. The Semantic Search application maps search words to an ontology. They assume all ontologies have an rdf:label, and rdf:title. It appears that they look for matches only on these strings. Once one (or maybe more) node(s) are matched, the system selects and displays information close to that node.

[GS05] A. Gulli and A. Signorini, "The Indexable Web is More than 11.5 Billion Pages," In *Proceedings of the 14th International Conference on World Wide Web*, pages 902-903, Chiba, Japan, May 2005.

The authors of this paper present a method for making a reasonable estimate of the number of indexable web pages.

[HH00] J. Heflin and J Hendler, "Searching the Web with SHOE," In *Artificial Intelligence for Web Search. Papers from the AAAI Workshop*, pages 35-40, Menlo Park, California, 2000.

The SHOE approach to searching the Semantic Web is unique in that it allows the user to choose a context from which to search. Users submit requests through a GUI by selecting an ontology, then filling in forms with ontology specific attributes. Like AskOntos, SHOE extracts and stores information. Results are tabular form, showing attribute values of the ontology. The disadvantage of this query system is the need for the user to learn a user interface.

[KB04] M. Klein and A. Bernstein, "Towards High-Precision Service Retrieval," *IEEE Internet Computing*, 8(1):30-36, January 2004.

This paper describes a novel approach to service retrieval (like document retrieval, only for software applications, components, and models). It is based on the sophisticated use of process ontologies. In the paper, they explain that PQL is a process query language made for ontologies.

[OWL04] Web Ontology Language (OWL): http://www.w3.org/TR/owl-features/, February 2005.

This W3C recommendation specifies a language for representing information as a conceptual model, or ontology, for computer processing.

[Sul05] D. Sullivan, "Search Engine Sizes," http://searchenginewatch.com/reports/article.php/2156481, August 2005.

This web article compares the size of seven popular search engines in terms of number of pages indexed. Google is reported to have about 8 billion web pages indexed.

[VM04] M. Vargas-Vera and E. Motta, "AQUA – Ontology-based Question Answering System," In *Proceedings of the Third International Mexican Conference on Artificial Intelligence*, pages 26-30, Mexico City, Mexico, April 2004.

AQUA is a system whose goals are similar to those of AskOntos. It is a QA system for the Semantic Web with a natural language interface. AQUA uses ontologies to derive the names of relations in their logic statements (the novel "Similarity Algorithm" does this). The system assumes that there is one large domain ontology. It relies on computational linguistic techniques as well as WordNet.

10. Artifacts

Aside from the thesis itself, the following will be produced as artifacts:

AskOntos, a semantic web query system that uses the Data Extraction Group's framework for extraction ontologies.

11. Signatures

This thesis proposal by Mark Vickers is accepted in its present form by the Department of Computer Science of Brigham Young University as satisfying the thesis proposal requirement for the degree of Master of Science.

_____	_____
Date	David W. Embley, Committee Chair
	_____
	Eric Ringger, Committee Member
	_____
	Mike Jones, Committee Member
	_____
	Parris Egbert, Graduate Coordinator