Ontobroker: How to Enable Intelligent Access to the WWW

Dieter Fensel, Stefan Decker, Michael Erdmann, and Rudi Studer

University of Karlsruhe, Institute AIFB, 76128 Karlsruhe, Germany
Email: {fensel, decker, erdmann, studer}@aifb.uni-karlsruhe.de,
http://www.aifb.uni-karlsruhe.de/WBS/broker

Abstract.
The World Wide Web (WWW) is currently one of the most important electronic information sources. However, its query interfaces and the provided reasoning services are rather limited. Ontobroker consists of a number of languages and tools that enhance query access and inference service in the WWW. It provides languages to annotate web documents with ontological information, to represent ontologies, and to formulate queries. The tool set of Ontobroker allows us to access information and knowledge from the web and to infer new knowledge with an inference engine based on techniques from logic programming.

1 Introduction

The World Wide Web (WWW) contains huge amounts of knowledge about almost all subjects you can think of. HTML documents enriched by multi-media applications provide knowledge in different representations (i.e., text, graphics, animated pictures, video, sound, virtual reality, etc.). Hypertext links between web documents represent relationships between different knowledge entities. Based on the HTML standard, browsers are available that present the material to users and use the HTML-links to browse through distributed information and knowledge units. However, retrieving information from the web is only weakly supported. Actually, the main query answering services the web provides are keyword-based search facilities carried out by different search engines, web crawlers, web indices, man-made web catalogues etc. (see [Mauldin, 1997], [Selberg & Etzioni, 1997]). Given a keyword, such an engine collects a set of knowledge bits from the web that use this keyword. [Luke et al., 1997] propose *ontologies* to improve the query answering support of the „knowledge base“ WWW. Ontologies are discussed in the literature as a means to support knowledge sharing and reuse [Fridman Noy & Hafner, 1997]. This approach to reuse is based on the assumption that if a modeling scheme -- i.e. *an ontology* -- is explicitly specified and agreed upon by a number of agents, it is then possible for them to share and reuse knowledge.

Clearly, we cannot expect that ontologies will be used by every web user and even if everybody used ontologies to annotate his web pages it will hardly ever be possible to negotiate on a worldwide standard for representing knowledge about all possible subjects. Therefore, we use the *metaphor of a newsgroup* to define the role of such an ontology. It is used by a group of people who share a common subject and a related point of view on this subject. Thus it allows them to annotate their documents to provide an *intelligent brokering service* that enables informed access to their web documents.

We designed and implemented some tools necessary to enable the use of ontologies for enhancing the web. We developed a broker architecture called Ontobroker [Ontobroker] with three core elements: a query interface for formulating queries, an inference engine used to derive answers, and a webcrawler used to collect the required knowledge from the web. We provide a *representation language* for formulating ontologies. A subset of it is used to formulate queries, i.e. to define the *query language*. A formal semantics is defined to enable automatic reasoning by the inference engine. An *annotation language* is offered to enable knowledge providers to enrich web documents with ontological information. The strength of our approach is the tight coupling of informal, semiformal and formal information and knowledge. This supports their maintenance and provides a service that can be used more generally for the purpose of *knowledge management* and for integrating knowledge-based reasoning and semiformal representation of documents (cf. [Kühn & Abecker, 1997]).

This paper is organized as follows. The languages and tools used to represent ontologies, formulate queries, and annotate web documents with ontological information are successively discussed in section 2, section 3, and section 4. Related work and conclusions are given in section 5.

2 The Query Formalism

The query formalism is oriented toward a frame-based representation of ontologies that defines the notion of instances, classes, attributes and values. The generic scheme for this is

```
O: C[A ->>V]
```

meaning that the object O is an instance of the class C with an attribute A that has a certain value V. At each position in the above scheme variables, constants or arbitrary expressions can be used. Furthermore because the ontology is part of the knowledge base itself, the ontology definitions can be used to validate the knowledge base. In the following we will provide some example queries to illustrate our approach. The ontology we show is developed as part of the *Knowledge Annotation initiative of the Knowledge Acquisition community (KA)*[2] [Benjamins et al., 1998]. It is used to describe research groups, topics and products of the knowledge acquisition community and some of its parts will be subsequently introduced in the paper. The following query asks for all known objects which are instances of the class researcher.

```
FORALL R <- R:Researcher.
```

Because the object identifier of a researcher is his/her homepage-URL, this query would result in a large list of URLs. This is one of the simplest possible queries. However, usually we are not interested in all researchers, instead we are interested in information about researchers with certain properties. e.g. we want to know the homepage, the last name and the email address of all researchers with first name *Richard.* To achieve this we can use the following query:

```
FORALL Obj, LN, EM <-
    Obj:Researcher[firstName ->> Richard;
                   lastName ->> LN; email ->> EM].
```

In our example scenario the Ontobroker gives the following answer (actually, there is only one researcher with first name *Richard* in the knowledge base).

```
Obj = http://www.iiia.csic.es/~richard/index.html
LN = Benjamins
EM = mailto:richard@iiia.csic.es
```

Another example is:

```
FORALL Obj , CP <-
    Obj:Researcher[lastName->>„Motta";
                   cooperatesWith ->> CP].
```

The interesting point with this query is that the ontology contains a rule specifying the symmetry of cooperating. This means that even if the researcher with the last name *Motta* has not specified a cooperation with another researcher, Ontobroker would derive such a cooperation if a second researcher has specified the cooperation. The ontology contains another strong rule that is used to abductively complete types. The relation *cooperatesWith* is defined for researchers. Therefore, for each instantiation for *CP* that cooperates with *Motta* or another researcher, Ontobroker also derives that this instantiation is an element of the class researcher. Both rules are examples of how Ontobroker can be used to derive new knowledge that is not directly represented on the WWW.

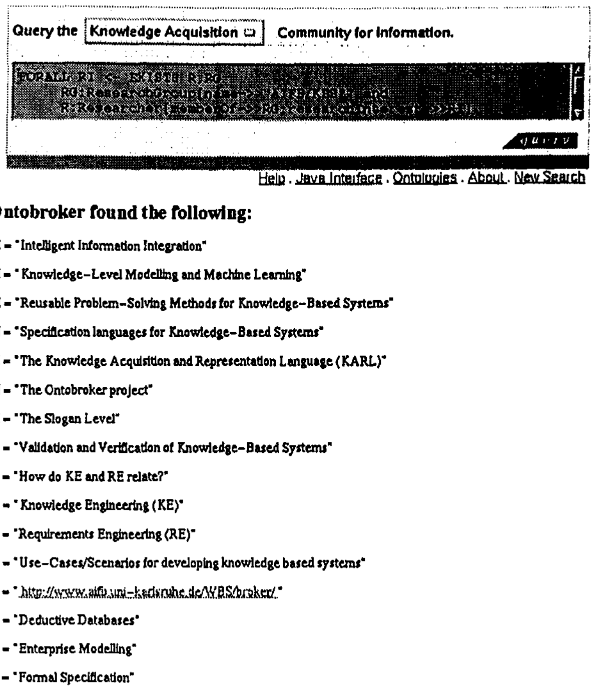
Ontobroker can also be used to collect distributed information. The query in Figure 1 collects all research topics of the members of the research group on knowledge-based systems at the Institute AIFB, i.e. it retrieves the research topics of a research group that are distributed at the different homepages of the researcher.

Another possibility is to query the knowledge base for information about the ontology itself, e.g. the query

```
FORALL Att, T <- Researcher[Att =>> T].
```

asks for all attributes of the class *Researcher* and their associated classes.

Ontobroker provides two query interfaces: a text based interface for expert users and a graphical interface for naive users. The text based interface allows the direct formulation



Query the Knowledge Acquisition Community for Information.

```
FORALL RI <- EXISTS R,RG
      RG:ResearchGroup[name->>"AIFB/KBS"] and
      R:Researcher[memberOf->>RG; [illegible] ->>RI].
```

Ontobroker found the following:

RI = "Intelligent Information Integration"
RI = " Knowledge-Level Modelling and Machine Learning"
RI = "Reusable Problem-Solving Methods for Knowledge-Based Systems"
RI = "Specification languages for Knowledge-Based Systems"
RI = "The Knowledge Acquisition and Representation Language (KARL)"
RI = "The Ontobroker project"
RI = "The Slogan Level"
RI = "Validation and Verification of Knowledge-Based Systems"
RI = "How do KE and RE relate?"
RI = " Knowledge Engineering (KE)"
RI = "Requirements Engineering (RE)"
RI = "Use-Cases/Scenarios for developing knowledge based systems"
RI = "http://www.aifb.uni-karlsruhe.de/WBS/broker/"
RI = "Deductive Databases"
RI = "Enterprise Modelling"
RI = "Formal Specification"

Fig. 1 The textual query interface

of queries in the above described query language. However, the direct formulation of the query string has two drawbacks:

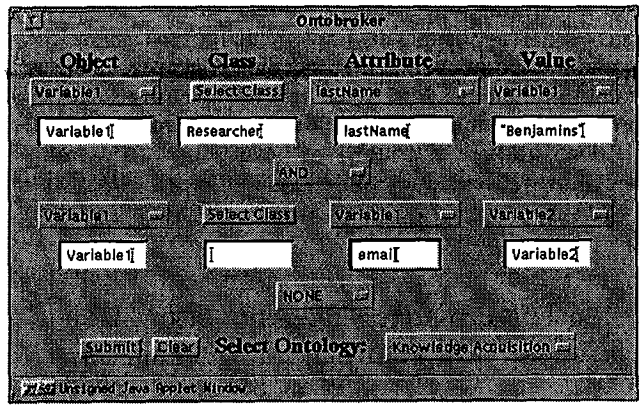
- The user has to know the syntax of the query language.
- The user also has to know the ontology when formulating a query.

The structure of the query language can be exploited to remedy the first drawback: the general structure of an elementary expression is:

```
Object:Class[Attribute ->> Value]
```

This provides the guidance when designing a query interface. Each part of the above depicted elementary expression can be related to an entry field. Possible values of the entry field can then be selected from a menu (e.g. variable names). This frees users from typing and understanding logical expressions as much as possible. The simple expressions can then be combined by logical connectives as shown in Figure 2 which asks for the researchers with last name *Benjamins* and their email addresses.

This does not resolve the second drawback: we also need support for selecting classes and attributes from the ontology. To allow the selection of classes, the ontology has to be presented in an appropriate manner. Usually a ontology can be represented as a large hierarchy of concepts. In regard to the handling of this hierarchy a user has at least two requirements: first he wants to scan the vicinity of a certain class looking for classes better suitable to formulate a certain query. Second a user needs an overview over the whole hierarchy to allow an quick and easy navigation from one class in the hierarchy to another



Ontobroker

Object	Class	Attribute	Value
Variable1	Select Class	lastName	Variable1
Variable1	Researcher	lastName	"Benjamins
	AND		
Variable1	Select Class	Variable1	Variable2
Variable1		email	Variable2
	NONE		

Submit Clear Select Ontology: Knowledge Acquisition

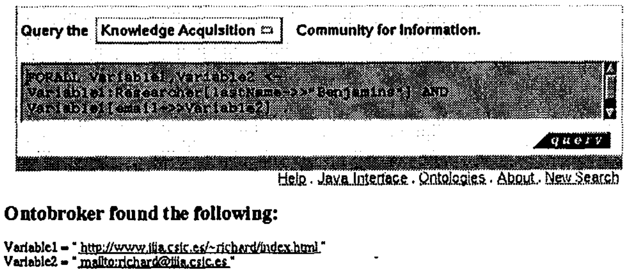
Unsigned Java Applet Window

Fig. 2 The advanced query interface.

class. These requirements are met by a presentation scheme based on Hyperbolic Geometry [Lamping et al., 1995]: classes in the center are depicted with a large circle, whereas classes at the border of the surrounding circle are only marked with a small circle (see Figure 3). The visualization techniques allows a quick navigation to classes far away from the center as well as a closer examination of classes and their vicinity. When a user selects a class from the hyperbolic ontology view, the class name appears in the class field and the user can select one of the attributes from the attribute choice menu because the pre-selected class determines the possible attributes. The interface is programmed in Java as an applet, thus it is executable on all major platforms where a Web-browser with Java support exists.[1] Based on these interfaces Ontobroker automatically derives the query in textual form and presents the result of the query (see Figure 4).

3 The Representation Formalisms and Inference Engine

The basic support we want to provide is query answering about instances of an ontology. The ontology may be described by taxonomies and rules. Since there are effective



Query the Knowledge Acquisition Community for Information.

```
FORALL Variable1,Variable2 <-
Variable1:Researcher[lastName->>"Benjamins"] AND
Variable1[email->>Variable2].
```

query

Help . Java Interface . Ontologies . About . New Search

Ontobroker found the following:

Variable1 = " http://www.iiia.csic.es/~richard/index.html "
Variable2 = " mailto:richard@iiia.csic.es "

Fig. 4 The textual query interface with automatically derived query

[1]. The hyperbolic ontology view is based on a Java-profiler written by Vladimir Bulatov and available at http://www.physics.orst.edu/~bulatov/HyperProf/index.html

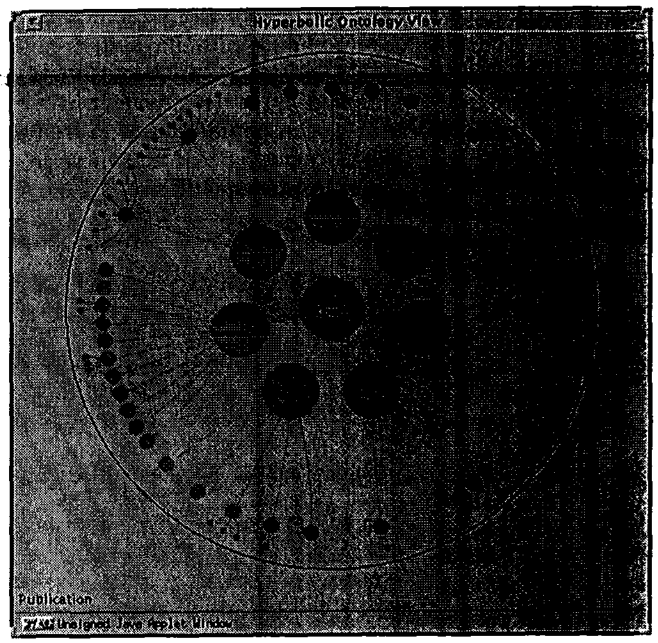


Fig. 3 The hyperbolic ontology view

and efficient query evaluation procedures for Horn logic like languages we based our inference engine on Horn logic. However, simple horn logic is not appropriate from an epistemological point of view for two reasons:

- First, the epistemological primitives of simple predicate logic (of which Horn logic is a subset) are not rich enough to support adequate representations of ontologies.
- Second, it is often very artificial to express logical relationships via Horn clauses.

We will subsequently discuss how we bypassed both shortcomings.

3.1 Elementary Expressions

Usually, ontologies are defined via concepts or classes, is-a relationships, attributes, further relationships, and axioms. Therefore an adequate language for defining the ontology has to provide modeling primitives for these concepts. Frame-Logic [Kifer et. al., 1995] provides such modeling primitives and integrates them into a logical framework providing a Horn logic subset. Furthermore, in contrast to most Description Logics, expressing the ontology in Frame-Logic allows for queries that directly use parts of the ontology as first class citizens. That is, not only instances and their values but also concept and attribute names can be provided as answers by means of variable substitutions.

3.2 Complex Expressions

More complex expressions can be built from the elementary ones. We distinguish between the following complex expressions: facts, rules, double rules, and queries. Facts are ground elementary expressions. A rule consists of a head, the implication sign <-, and the body. The head is just a conjunction of elementary expressions (connected using *AND*). The body is a complex formula built from elementary expressions and the usual predicate logic connectives (implies: ->, implied by: <- , equivalent: <->, *AND, OR*, and *NOT*). Variables can be introduced in front of the head (with an *FORALL*-quantifier) or anywhere in the body (using *EXISTS* and *FORALL*-quantifiers).

3.3 An Illustration

Ontologies defined with this language consist mainly of two or three parts:

- The concept hierarchy, which defines the subclass relationship between different classes, together with the attribute definitions. These two parts can be split for reasons of readability.
- A set of rules defining relationships between different concepts and attributes.

A part of an example ontology (see [Ontobroker] for the entire ontology) defining a small concept hierarchy, some attributes, and two rules relating different concepts are provided in Table 1.

The concept hierarchy consists of elementary expressions declaring subclass relationships. The attribute definitions declare attributes of concepts and the valid types that a value of an attribute must have. The first rule ensures symmetry of cooperation and the second rule specifies that whenever a person is known to have a publication then the publication also has an author who is the particular person and vice versa. This kind of rule completes the knowledge and frees a knowledge provider to provide the same information at different places reducing development as well as maintenance efforts.

3.4 The Inference Engine

The *inference engine* of Ontobroker has two key components: the translation (and re translation) process from the rich modeling language to a restricted one and the evaluation of expressions in the restricted language. For technical reasons we have decided against direct evaluation of expressions of the rich modeling language (see [Decker et al., submitted] for more details). The expressions are translated into generalized logic programs that are translated further into normal logic programs via a Lloyd-Topor transformation. Standard techniques from deductive databases are applicable to implement the last stage: the bottom up fixpoint evaluation procedure. Because we allow negation in the clause body we have to carefully select an appropriate semantics and evaluation procedure. To deal with non stratified negation we have adopted the well-founded model semantics and compute this semantics with dynamic filtering and the alternating fixpoint approach [Van Gelder, 1993].

4 The Provider Side: Annotating Web-Pages with Ontological Information

Knowledge contained in the WWW is generally formulated using the Hyper-Text Mark-up Language (HTML). Therefore, we developed an extension to the HTML syntax to enable the ontological annotation of web pages.[2] We will only provide the general idea (see [Ontobroker] for more details). An extract from an example page is given in Figure 5.

The idea behind our approach is to take HTML as a starting point and to add only few ontologically relevant tags. With these minor changes to the original HTML pages the knowledge contained in the page is annotated and made accessible as facts to the Ontobroker. This approach allows the knowledge providers to annotate their web pages gradually, i.e. they do not have to completely formalize the knowledge contained therein. Further, the pages remain readable by standard browsers like Netscape Navigator or MS Explorer. Thus there is no need to keep several different sources up-to-date and consistent, reducing development as well as maintenance efforts considerably. All factual ontological information is contained in the HTML page itself.

We provide three different epistemological primitives to annotate ontological information in web documents:

[2] We did not make use of the *extensible Markup Language (XML)* [XML] to define our annotation language as an extension of HTML because many existing HTML pages are not well-formed XML documents, i.e., the document type HTML defined in XML is more restrictive than HTML as it is widely used now. However, we will provide a wrapper to generate descriptions in the Resource Description Format (RDF) as soon as RDF will become a standard.

Table 1. Some Ontology Definitions

Concept Hierarchy	**Attribute Definitions**	**Rules**

Object[].
Person :: Object.
Employee :: Person.
Researcher :: Employee.
Publication::Object.
Organization::Object.
	Person[
	firstName =>> STRING;
	lastName =>> STRING;
	eMail =>> STRING;
	publication =>> Publication].
	Employee[
	affiliation =>> Organization;
	...].
		FORALL Person1, Publication1
		Publication1:Publication[author ->> Person1] <->
		Person1:Person[publication ->> Publication1].

- An object identified by an URL (Uniform Resource Locator) can be defined as an instance of a certain class.
- ~~The value of an object's attribute can be set.~~
- A relationship between two or more objects may be established.

All three kinds are expressed by using an extended version of a frequent HTML tag, i.e. the anchor tag:

```
<a ...> ... </a>
```

Typically a provider of information first defines an object. This is done by stating the class of the ontology of which it is an instance. For example, if Richard Benjamins would like to define himself as an object, he would say he is an instance-of the class *Researcher*. To express this in our HTML extension he would use the following line on his home page (see Figure 5).

```
<a onto=" 'http://www.iiia.csic.es/~richard' : Researcher"> </a>
```

This line states that the object denoted by the handle *'http://www.iiia.csic.es/~richard'* is an instance of class *Researcher*. Actually the handle given above is the URL of Richard Benjamins home page, thus from now on he is denoted as a researcher by the URL of his home page.

Each class is possibly associated with a set of attributes. Each instance of a class can define values for these attributes. To define an attribute value on a web page the knowledge provider has to name the object he wants to define the value for, he has to name the attribute and associate it with a value. For example, the ontology contains an attribute *email* for each object of class *Researcher*. If Richard Benjamins would like to provide his email address, he would use this line on his home page.

```
<a onto=" 'http://www.iiia.csic.es/~richard'
        [email='mailto:richard@iiia.csic.es'] ">
</a>
```

This line states that the object denoted by the handle *'http://www.iiia.csic.es/~richard'* has the value *'mailto:richard@iiia.csic.es'* for the attribute *email*.

Several objects and attributes can be defined on a single web page, and several objects can be related to each other explicitly. Given the name of a relation *REL* and the object handles Obj_1 to Obj_n this definition looks like this:

```
<a onto= "REL(Obj1, Obj2, Obj3, ... Objn)" > ... </a>
```

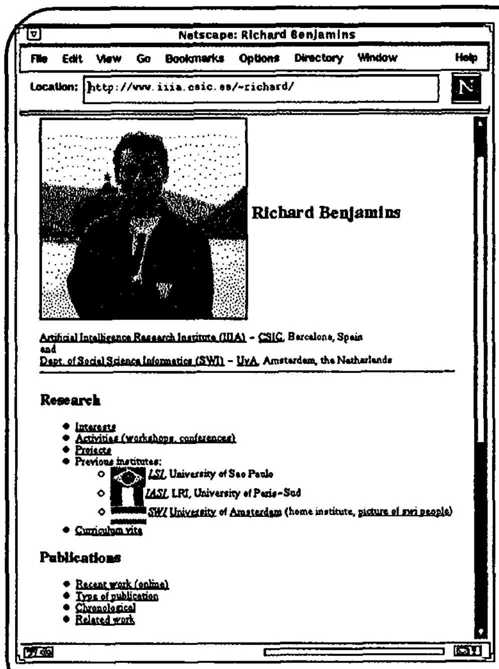
The listed examples look rather clumsy, especially because of their long object handles and the redundancy coming from writing information twice, once for the browser and again for Ontobroker. So the annotation language provides some means to facilitate annotating web pages and eliminating a large share of the clumsiness and redundancy (cf. [Ontobroker]). For example, to define on a web page that an object is an instance of a class, e.g. that Richard Benjamins is a *Researcher*, we can use the following kind of annotation (see Figure 5):

```
<a onto= "page:Researcher"> </a>
```

The following annotation defines the *affiliation* attribute of the object denoted by the URL of the current *page* and takes the value from the anchor-tag's *href*-attribute (see Figure 5).

```
<a  onto="page[affiliation=href]"
    href="http://www.swi.psy.uva.nl/">
            Department of Social Science Informatics (SWI)
</a>
```

Finally, the annotation


Netscape: Richard Benjamins
File Edit View Go Bookmarks Options Directory Window Help
Location: http://www.iiia.csic.es/~richard/

Richard Benjamins

Artificial Intelligence Research Institute (IIIA) - CSIC, Barcelona, Spain
and
Dept. of Social Science Informatics (SWI) - UvA, Amsterdam, the Netherlands

Research

- Interests
- Activities (workshops, conferences)
- Projects
- Previous institutes:
  - LSI, University of Sao Paulo
  - IASI, LRI, University of Paris-Sud
  - SWI University of Amsterdam (home institute, picture of swi people)
- Curriculum vita

Publications

- Recent work (online)
- Type of publication
- Chronological
- Related work

```
<html>
<head><TITLE> Richard Benjamins </TITLE>
<a onto="page:Researcher"> </a>
</head>

<H1> <A HREF="pictures/id-rich.gif">
<IMG align=middle SRC="pictures/richard.gif"></A>
<a onto="page[photo=href]"
HREF="http://www.iiia.csic.es/~richard/pictures/richard.gif" ></a>

<a onto="page[firstName=body]">Richard</a>
<a onto="page[lastName=body]">Benjamins </a>
</h1> <p>

<A onto="page[affiliation=body]" HREF="#card">
Artificial Intelligence Research Institute (IIIA)</A> -
<a href="http://www.csic.es/">CSIC</a>, Barcelona, Spain <br>
and <br>
<A onto="page[affiliation=body]" HREF="http://www.swi.psy.uva.nl/">
Dept. of Social Science Informatics (SWI)</A>
-
<A HREF="http://www.uva.nl/uva/english/">UvA</A>,
Amsterdam, the Netherlands

<HR>
```

Fig. 5 An example HTML page

<a onto="page[firstName=body]"> Richard </a>

defines *Richard* (contained between <a ...> and </a>) as the value of the attribute *firstName* of the object which is denoted by *page* (see Figure 5). Through this convention the annotation of web pages becomes more concise and redundancy can be nearly avoided.

5 Conclusions and Related Work

Up to now, the inference capabilities of the World Wide Web are very limited. In essence, they are restricted to keyword-based search facilities which are offered by the various Web search engines. In this paper we introduced methods and tools for enhancing the Web to form a knowledge-based WWW. We proposed ontologies as a means to annotate WWW documents with semantic information and used the metaphor of a newsgroup to define a collection of people who share a common view on a subject and thus a common ontology. To define various subnets in the WWW, different ontologies can be used to annotate Web documents. We use Frame logic for defining ontologies and an appropriate subset for specifying (semantic) queries to the Web. An annotation language for attaching ontological information to Web documents is offered which avoids redundancy as far as possible. Our Ontobroker tool includes a query interface for formulating queries, an inference engine for deriving answers to the posed queries, and a web crawler for searching through the various subnets and translating the ontological annotations into facts for the inference engine. In this manner, the web crawler implements a wrapper which hides the syntactical structure of annotations from the inference engine and the query client.

Ontobroker is the basis for realizing the Knowledge Acquisition Initiative $(KA)^2$ ([Benjamins et al., 1998]) and for developing a knowledge management system for industrial designers in regard to ergonomic questions. In the latter project, the same knowledge may be used by users, i.e. industrial designers, and as input and output for inference processes of the system. This twofold use of the same piece of knowledge is enabled through the tight coupling of semiformal and formal knowledge in Ontobroker. In the paper, we presented Ontobroker mainly as a tool to enhance information access. However, *maintenance* of distributed and heterogeneous information sources may become an even more important topic given the steadily increasing amount of knowledge that is provided by semiformal knowledge sources like web documents. Annotating parts of documents with semantical information enable automatic support for modifying these documents. Instead of searching by hand through several documents that may contain the same or parts of the same information that needs to be changed one can automatically propagate such modifications without changing the semiformal nature of the documents.

The approach closest to ours is SHOE, which introduced the idea of using ontologies to annotate information in the WWW [Luke et al., 1997]. HTML pages are annotated via ontologies to support information retrieval based on semantic information. However, there are major differences in the underlying philosophy: In *SHOE*, providers of information can introduce arbitrary extensions to a given ontology. Furthermore, no central provider index is defined. As a consequence, when specifying a query the client may not know all the ontological terms which have been used to annotate the HTML pages and the web crawler may miss knowledge fragments because it cannot parse the entire WWW. Thus the answer may miss important information and the web crawler may miss knowledge bits. In contrast, Ontobroker relies on the notion of an *ontogroup* defining a group of Web users who agree on an ontology for a given subject. Therefore, both the information providers and the clients have complete knowledge of the available ontological terms. In addition, the provider index of the Ontocrawler provides a complete collection of all annotated HTML pages. Thus, Ontobroker can deliver complete answers to the posed queries. The philosophy of Ontobroker is also tailored to homogeneous intranet applications, e.g. in the context of knowledge management within an enterprise. SHOE and Ontobroker also differ with respect to their inferencing capabilities. SHOE uses description logic as its basic formalism and currently offers rather limited inferencing capabilities. Ontobroker relies on Frame-Logic and supports rather complex inferencing for query answering. One can situate Ontobroker in the general context of approaches that support the integration of *distributed* and *heterogeneous* information sources using a *mediator* [Wiederhold & Genesereth, 1997] that translates user queries into sub-queries for the different information sources and integrates the sub-answers. Wrappers and content descriptions of information sources provide the connection of an information source to the mediator. However, these approaches assume that the information sources have a stable syntactical structure that a wrapper can use to extract semantic information. Given the heterogeneity of any large collection of web pages, this assumption seems hardly to be fulfilled in our application area. Therefore, we delegated the semantical enrichment of the information sources to the provider and make no assumptions about the format of the information source and its changes. However, wrapper and annotation-based approaches are complementary. [Ashish & Knoblock, 1997] distinguish three types of information sources at the web: multiple-instance sources, single-instance sources, and loosely-structured sources. The former two types have a stable format that can be used by a wrapper to extract information. The latter type covers home pages of persons etc. where the layout is neither standard nor stable over time. Writing wrappers for this type of sources would be a time-consuming activity which would be soon out of date. However, writing wrappers for stable information sources that automatically generate factual knowledge processable by Ontobroker enables us to broaden our approach to include structured information sources that do not make use of our annotation language.

Acknowledgments

We thank Richard Benjamins and Rainer Perkuhn for their helpful comments and Asun Gomez-Perez for providing the Ontolingua translation. Special thanks to Jürgen Angele who developed the inference engine that is used by Ontobroker.

References

[Ambite & Knoblock, 1997] J. L Ambite and C,. A. Knoblock: Agents for Information Gathering, *IEEE Expert*, September/October 1997.

[Ashish & Knoblock, 1997] N. Ashish and C. Knoblock: Semi-automatic Wrapper Generation for Internet Information Sources. In *Proceedings of the IFCIS Conference on Cooperative Information Systems (CoopIS)*, Charlston, South Carolina, 1997.

[Benjamins et al., 1998] V. R. Benjamins, D. Fensel, A. Gomez-Perez, S. Decker, Michael Erdmann, E. Motta, and M. Musen: Knowledge Annotation Initiative of the Knowledge Acquisition Community $(KA)^2$. In *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge Based System Workshop (KAW'98)*, Banff, Canada, April 18-23, 1998.

[Decker et al., submitted] S. Decker, D. Fensel, M. Erdmann, and R. Studer: The Technical Core of Ontobroker. Submitted, available via [Ontobroker].

[Euzenat, 1996] J. Euzenat: Corporate Memory through Cooperative Creation of Knowledge Bases and Hyper-documents. In: *Proceedings of the 10th Banff Knowledge Acquisition Workshop (KAW 96)*, Banff, Canada, November 1996

[Fridman Noy & Hafner, 1997] N. Fridman Noy and C. D. Hafner: The State of the Art in Ontology Design, *AI Magazine*, 18(3):53--74, 1997.

[Kifer et al., 1995] M. Kifer, G. Lausen, and J. Wu: Logical Foundations of Object-Oriented and Frame-Based Languages, *Journal of the ACM*, 42, 1995.

[Kühn & Abecker, 1997] Otto Kühn and Andreas Abecker: Corporate Memories for Knowledge Management in Industrial Practice: Prospects and Challenges, *Journal of Universal Computer Science, Special Issue on Information Technology for Knowledge Management.* Springer Science Online, 3(8), August 1997.

[Lamping et al., 1995] L. Lamping, R. Rao, and Peter Pirolli.: A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems.* 1995

[Luke et al., 1997] S. Luke, L. Spector, D. Rager, and J. Hendler: Ontology-based Web Agents. In *Proceedings of First International Conference on Autonomous Agents*, 1997.

[Mauldin, 1997] M. L. Mauldin: Lycos: Design Choices in an Internet Search Engine, *IEEE Expert*, January-February 1997. http://www.lycos.com

[Ontobroker] http://www.aifb.uni-karlsruhe.de/WBS/broker

[RDF] Resource Description Framework, http://www.w3.org/Metadata/RDF/Group/WD-rdf-syntax

[Selberg & Etzioni, 1997] E. Selberg and O. Etzioni: The MetaCrawler Architecture for Resource Aggregation on the Web, *IEEE Expert*, January-February 1997. http://www.metacrawler.com

[URL] Uniform Resource Locator, http://www.w3.org/pub/WWW/Protocols

[Van Gelder, 1993] A. Van Gelder: The Alternating Fixpoint of Logic Programs with Negation, *Journal of Computer and System Sciences*, 47(1):185-221, 1993.

[Wiederhold & Genesereth, 1997] G. Wiederhold and M. Genesereth: The Conceptual Basis for Mediation Services, *IEEE Expert*, September/October, pp. 38-47,1997.

[XML] Extensible Markup Language, http://www.w3.org/TR/PR-xml-971208