

The Technical Core of Ontobroker

Stefan Decker, Dieter Fensel, Michael Erdmann, and Rudi Studer
University of Karlsruhe, Institute AIFB, 76128 Karlsruhe, Germany
Email: {decker, fensel, erdmann, studer}@aifb.uni-karlsruhe.de,
<http://www.aifb.uni-karlsruhe.de/WBS/broker>

Abstract. The World Wide Web can be viewed as the largest knowledge base that has ever existed. However, its support in query answering and automated inference is very limited. Therefore we developed Ontobroker that uses formal ontologies to enhance query access and inference service of the WWW. The paper describes its technical core that consists of formalisms and tools for formulating queries, for defining ontologies, and for annotating HTML documents with ontological information.

1 Introduction

The World Wide Web (WWW) contains huge amounts of knowledge about most subjects one 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 that use the HTML-links to browse through distributed information and knowledge units. However, taking *the metaphor of a knowledge base* as a way to look at the WWW brings the bottleneck of the web into mind. Its support of automated inference is very limited. Deriving new knowledge from existing knowledge is hardly supported. Actually, the main inference services the web provides are keyword-based search facilities carried out by different search engines, web crawlers, web indices, man-made web catalogues etc. Given a keyword, such an engine collects a set of knowledge portions from the web that use this keyword. This limited inference access to existing knowledge stems from the fact that there are only two main types of standardization for knowledge representation on the web. The HTML standard is used to represent knowledge in a (browser and) man-readable way and to define links between different knowledge units. Furthermore, mainly the English language is used to represent the knowledge units. [Luke et al., 1997] propose *ontologies* to improve the automatic inference support of the knowledge base WWW. An ontology provides “an explicit specification of a conceptualization” [Gruber, 1993]. Ontologies are discussed in the literature as means to support knowledge sharing and reuse ([Farquhar et al., 1997], [Fridman Noy & Hafner, 1997]). This approach to reuse is based on the assumption that if a modelling 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.

We designed and implemented a couple of 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 support their maintenance and provide a service that can be used more general for the purpose of knowledge management and for integrating knowledge-based reasoning and semiformal representation of documents (cf. [Skuce,1997]).

The general architecture of the ontology-based brokering service Ontobroker consists of three main elements: a *query interface*, an *inference engine*, and a *webcrawler* (called Ontocrawler). Each of these elements is accompanied with a formalization language: the query language for formulating queries, the representation language for specifying ontologies, and the annotation language for annotating web documents with ontological information. In this paper, we foccus on the technical core of Ontobroker. A more complete description can be found in [Fensel et al., 1998] and a more detailed discussion of the *use* of the query and inference support of Ontobroker is provided by [Fensel et al., submitted].

The contents of the paper is organised as follows. The languages used to represent ontologies, to formulate queries, and to annotate web documents with ontological information are discussed in section 2. In section 3 we discuss the three main tools of Ontobroker: its graphical and logic-based query interface, its inference engine, and its webcrawler. Related work and conclusions are given in section 4.

2 The Languages of Ontobroker

In the following, we discuss the formalisms used by Ontobroker. First, we describe the representation formalism used to define ontologies. Second, we discuss the query formalism that is used by a client asking for information. Finally we discuss the annotation formalism that is used by the knowledge provider to annotate web documents with ontological information.

2.1 The Representation Formalisms for Ontologies

The basic support we want to provide is query answering over instances of an ontology. The ontology may be described by taxonomies and rules. Since there are effective and efficient query evaluation procedures for Horn-logic like languages we based our inference engine on horn logic.¹ However, simple horn logic is not appropriate from an epistemological point of view for two reasons:

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

We will subsequently discuss how we bypassed both shortcomings.

¹ However, to support interchancability a translator from Ontobroker to Ontolingua is described in [Fensel et al., 1998].

2.1.1 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 via variable substitutions.

We use a slightly modified variant of Frame-Logic, which suits our needs. Mainly the following elementary modeling primitives are used:

- Subclassing: $C_1 :: C_2$, meaning that class C_1 is a subclass of C_2 .
- Instance of: $O : C$, meaning that O is an instance of class C .
- Attribute Declaration: $C_1[A \Rightarrow C_2]$, meaning that for the instances of class C_1 an attribute A is defined, whose value must be an instance of C_2 .
- Attribute Value: $O[A \Rightarrow V]$, meaning that the instance O has an attribute with value V .
- Part-of: $O_1 < O_2$, meaning that O_1 is a part of O_2 .
- Relations: predicate expressions like $p(a_1, \dots, a_2)$ can be used as in usual logic based representation formalisms, except that not only terms can be used as arguments, but also object expressions.

2.1.2 Complex Expressions

From the elementary expressions more complex ones can be built. 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 \leftarrow , 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: \rightarrow , implied by: \leftarrow , equivalent: \leftrightarrow , 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). A double rules is an expression of the form:

$$head \leftrightarrow body,$$

where the *head* and *body* are just conjunctions of elementary expressions. Examples of double rules are given in Table 1. An EBNF syntax description of the complete representation language is given in [Fensel et al., 1998].

2.1.3 An Illustration

Ontologies defined with this language consist mainly of two resp. 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 readability reasons.

- 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.

Table 1. Some Ontology Definitions

Concept Hierarchy	Attribute Definitions	Rules
Object[. Person :: Object. Employee :: Person. AcademicStaff :: Employee. Researcher :: AcademicStaff. Publication::Object.	Person[firstName =>> STRING; lastName =>> STRING; eMail =>> STRING; ... publication =>> Publication]. Employee[affiliation =>> Organization; ...]. Researcher[researchInterest =>> ResearchTopic; memberOf =>> ResearchGroup; cooperatesWith =>> Researcher]. Publication[author =>> Person; title =>> STRING; year =>> NUMBER; abstract =>> STRING].	FORALL Person1, Person2 Person1:Researcher [cooperatesWith ->> Person2] <- Person2:Researcher [cooperatesWith ->> Person1]. FORALL Person1, Publication1 Publication1:Publication [author ->> Person1] <-> Person1:Person [publication ->> Publication1].

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 also the publication has an author, who is the particular person, and vice versa. This kind of rules complete the knowledge and frees a knowledge provider to provide the same information at different places reducing development as well as maintenance efforts.

2.2 The Query Formalism

The query formalism is oriented towards Frame-Logic syntax, that defines the notion of instances, classes, attributes and values. The generic schema 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. If 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*, we achieve this with 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. That means, even if the researcher with 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. 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.

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

Mainly knowledge contained in the WWW is formulated using the Hyper-Text Mark-up Language (HTML). Therefore, we developed an extension to the HTML syntax to enable ontological annotation of web pages.² We will only provide the general idea (see [Fensel et al., 1998] for more details). An extract from an example page is given in Figure 1.

The idea behind our approach is to take HTML as a starting point and to add only few ontologically relevant tags. By these few 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 considerable. 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. Compare also section 4.

- 1) An object identified by an URL (Uniform Resource Locator) can be defined as an instance of a certain class.
- 2) The value of an object's attribute can be set.
- 3) 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:

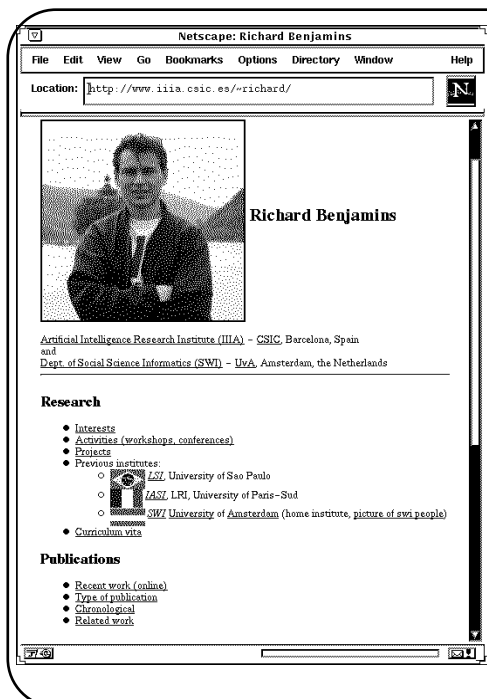
```
<a ...> ... </a>
```

Typically a provider of information first defines an object. This is done by stating which class of the ontology it is an instance of. 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.

```
<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 as a researcher is denoted 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.



```
<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. 1 An example HTML 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 Obj₁ to Obj_n this definition looks like this:

```
<a onto= "REL(Object1, Object2, Object3, ..., Objectn)" > ... </a>
```

The listed examples look rather clumsy, esp. because of their long object handles and the redundancy, due to writing information twice, once for the browser and second for Ontobroker. So the annotation language provides some means to ease annotating web pages and get rid of a big portion of the clumsiness and redundancy (cf. [Fensel et al., 1998]). 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, one can use the following kind of annotation:

```
<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.

```
<a onto="page[affiliation=href]" href="http://www.iiia.csic.es/">  
  IIIA - Artificial Intelligence Institute.</a>
```

Finally, the annotation

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

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

3 The Tool Set of Ontobroker

Ontobroker mediates between clients and knowledge providers with three tools. The query interface is used by clients to formulate queries and to receive answers. The inference engine derives answers based on the ontology and the facts that have been found at the web. The webcrawler (called Ontocrawler) collects annotations from the web and translates them into facts that can be processed by the inference engine.

3.1 The Query Interface

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 of queries in the above described query language. However, the direct formulation of the query string has two drawbacks: (1) The user has to know the syntax of the query language and (2) the user also has to know the ontology, when formulating a query.

To remedy the first drawback, the structure of the query language can be exploited: 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 fields. 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: one 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. Concerning 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 easy and quick navigation from one class in the hierarchy to another 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 visualisation techniques allows an quick navigation to classes far away from the center as well as the close 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 preselected 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.³ Based on these interfaces Ontobroker derives automatically the query in textual form and present the result of the query.

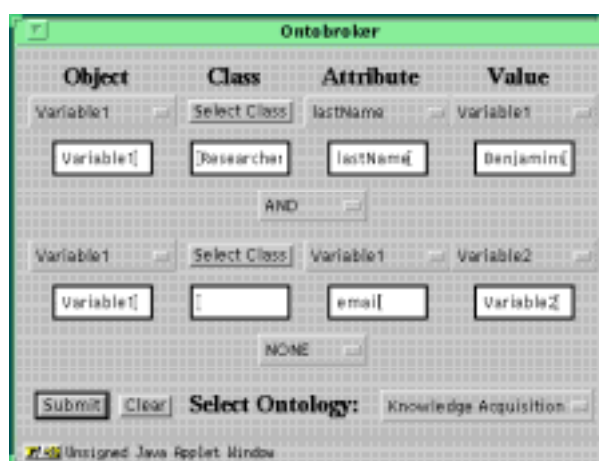


Fig. 2. The advanced query interface.

³. The hyperbolic ontology view is based on a Java-profiler written by Vladimir Bulatov and available on <http://www.physics.orst.edu/~bulatov/HyperProf/index.html>.

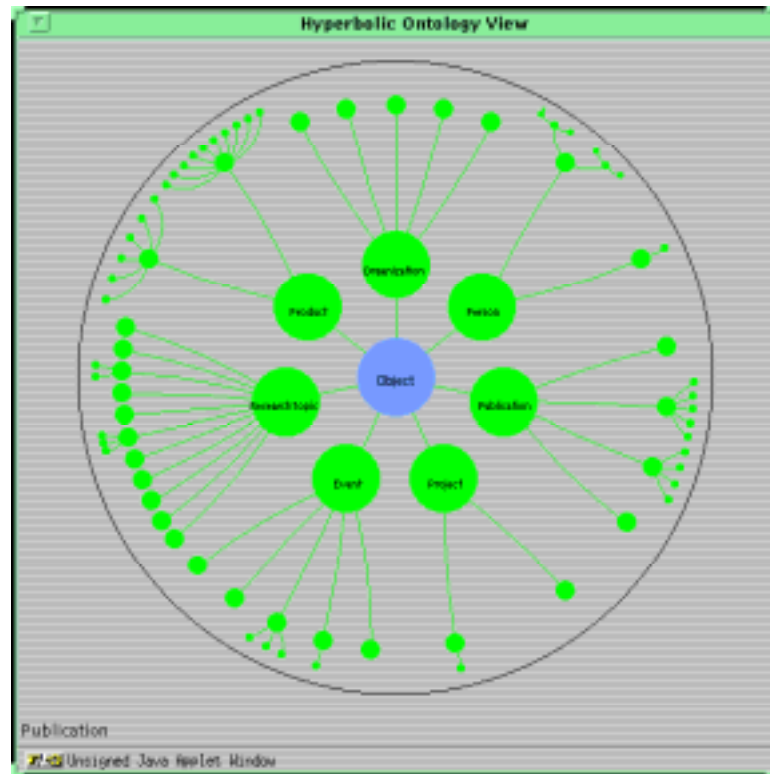


Fig. 3. The hyperbolic ontology view.

3.2 The Inference Engine of the Ontobroker

The inference engine of Ontobroker has two key components: the translation (and retranslation) process from the rich modelling language to a restricted one, and the evaluation of expressions in the restricted language. In the following, we describe both processes.

The input of the inference engine consists of the ontology, collected facts from the web and queries formulated in Frame-Logic. We have decided against direct evaluation of expressions of the rich modelling language. Due to the conceptual richness of the language evaluation techniques would be rather complicated and difficult to build. There are techniques known for evaluating Frame-Logic, see [Frohn et al, 1997], but they do not support the whole language and semantics we need (e.g. full first order rule bodies). Furthermore a direct evaluation approach would be very inflexible, a small change in the input language would result in changes of the whole system. The situation is very similar to compiler construction: usually a language is not compiled directly to a target language, but through several intermediate states and languages. This helps bridging the conceptual gap between the language and the target language. We adopted that approach: the input is processed and translated in several stages: The first step (besides the necessary parser) is the Frame-Logic-translator, that translates the Frame-Logic expressions to first-order logic expressions. Table 2 gives an idea of how this translation is performed, but it does not catch the complete translation: e.g. more complex Frame-logic expressions like

$$O[A->>V:C[AA->>VV]]$$

Table 2. Principle of Translating Frame Logic to Predicate Logic

Frame Logic	Meaning	Predicate Logic
$C_1 :: C_2$	class C_1 is a subclass of C_2	$\text{sub}(C_1, C_2)$
$O : C$	O is an instance of class C	$\text{isa}(O, C)$
$C_1[A \Rightarrow C_2]$	for the instances of C_1 an attribute A is defined, the value must be an instance of C_2	$\text{att_type}(C_1, A, C_2)$
$O[A \Rightarrow V]$	the instance O has an attribute A , the value is V	$\text{att_val}(O, A, V)$
$O_1 <: O_2$	O_1 is a part of O_2	$\text{part_of}(O_1, O_2)$

can also be translated. The output of this stage are generalized logic programs. After this translation the output has to be translated further to normal logic programs via a Lloyd-Topor transformation [Lloyd & Topor, 1984].

As a result we obtain a normal logic program. 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. If the resulting program is stratified, we use simple stratified semantics and evaluate it with a technique called dynamic filtering (cf. [Kifer & Lozinskii, 1986], [Angele, 1993]). But the translation of Frame Logic usually results in a logic program with only a limited number of predicates, so the resulting program is often not stratified. To deal with non stratified negation we have adopted the well-founded model semantics [Van Gelder et al., 1991] and compute this semantics with dynamic filtering and the alternating fixpoint approach [Van Gelder, 1993].

3.3 The Ontocrawler

Ontocrawler is a simple cgi-script that periodically caches the annotated pages from the web. For finding the pages it consults the index pages of each provider. For this purpose, the providers need to register.

4 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 the paper we introduced methods and tools for enhancing the Web to 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 which 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 with Web documents is offered as well avoiding 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. Ontobroker is the basis for realizing the Knowledge Acquisition Initiative (KA)² [Benjamins & Fensel, 1998] and for developing a knowledge management system for industrial designers concerning ergonomic questions. In the latter project, the same knowledge may be used by humans and for inferences 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.

One can situate Ontobroker in the general context of approaches that support the integration of *distributed* and *heterogeneous* information sources like e.g. Infomaster [Genesereth et al., 1997], Information Manifold [Levy et al., 1996], and SIMS [Arens et al., 1993]. Instead of assuming a global data scheme such systems have a *mediator* [Wiederhold, 1992] that translates user queries into sub-queries on 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 informations. Given the heterogeneity of any large collection of web pages this assumptions seems hardly 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 (cf. [Ashish & Knoblock, 1997]). 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 is soon out of date, too. However, writing wrappers for stable information sources that automatically generate factual knowledge processable by Ontobroker enables to broaden our approach to structured information sources that do not make use of our annotation language.

The approach closed to ours is SHOE [Luke et al., 1997] that introduced the idea of using ontologies to annotate information in the WWW. HTML pages are annotated via ontologies to support information retrieval based on semantic information. However, there exist main 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 portions because it cannot parse the entire WWW.

In contrast, Ontobroker relies on the notion of an *ontogroup* [Fensel et al., 1997] defining a group of Web users that agree on an ontology for a given subject. Therefore, both the informations 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 (see [Kandzia & Schleppehorst, 1997], [Fensel et al., to appear] for comparisons of both representation and reasoning paradigms).

Finally, we decided to design our annotation language as a small extension of HTML because most documents on the web use this formalism. However, there are some new trends which we have to be aware. The W3C—the international *World Wide Web Consortium* for developing and promoting standards for the web—currently develops the *resource description framework* (RDF) [RDF]. This format can be used to add meta information to documents, i.e. to include semantical information about documents. This approach shows a number of similarities with Ontobroker, however there are profound differences. The annotation information is tightly integrated into HTML in Ontobroker. This reduces redundancy of information on a web page to a minimum. Meta data defined in RDF have to be provided on an extra page or en-block inside of a web-page. Therefore, elements from a web page like text fragments or links cannot directly be annotated with semantics. These elements must be repeated for enriching them with meta-information. This design decision may cause significant problems for maintaining web documents due to the redundancy of the information. However, when a final version of RDF will be recommended by the W3C it will be an easy task to implement a wrapper that automatically generates RDF definitions from annotation in Ontobroker. In that sense the annotation language of Ontobroker can be seen as a maintenance tool for RDF description because it allows the direct annotations of elements of a web page and their separate content description will be generated automatically. Using automatically generated RDF descriptions makes the annotated knowledge available to agents and brokering services that searches the web for information.

Acknowledgements. 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 for L-KARL that is used by Ontobroker.

References

- [Angele, 1993] J. Angele: *Operationalisierung des Modells der Expertise mit KARL*, Ph.D. thesis, Infix, St. Augustin, 1993.
- [Arens et al., 1993] Y. Arens, C. Y. Chee, C.-N. Hsu and C. Knoblock: Retrieving and Integrating Data From Multiple Information Sources, *International Journal of Intelligent Cooperative Information Systems*, 2(2):127—158, 1993
- [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)*, Charleston, South Carolina, 1997.
- [Benjamins & Fensel, 1998] R. Benjamins and D. Fensel: Community is Knowledge! in (KA)². In *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW'98)*, Banff, Canada, April 18-23, 1998.

- [Farquhar et al., 1997] A. Farquhar, R. Fikes, and J. Rice: The Ontolingua Server: a Tool for Collaborative Ontology Construction, *International Journal of Human-Computer Studies (IJHCS)*, 46(6):707—728, 1997.
- [Fensel et al., 1997] D. Fensel, M. Erdmann, and R. Studer: Ontology Groups: Semantically Enriched Subnets of the WWW. In *Proceedings of the 1st International Workshop Intelligent Information Integration during the 21st German Annual Conference on Artificial Intelligence*, Freiburg, Germany, September 9-12, 1997.
- [Fensel et al., 1998] D. Fensel, S. Decker, M. Erdmann, and R. Studer: Ontobroker: How to make the WWW Intelligent, research report, Institute AIFB, 1998. <http://www.aifb.uni-karlsruhe.de/WBS/broker>.
- [Fensel et al., to appear] D. Fensel, M.-C. Rousset, and S. Decker: Workshop on Comparing Description and Frame Logics, to appear in *Data and Knowledge Engineering*.
- [Fensel et al., submitted] D. Fensel, S. Decker, M. Erdmann, and R. Studer: Ontology-based Query and Inference Service for the WWW, submitted to *IEEE Expert, Special Issue on the Use of Ontologies*, available via <http://www.aifb.uni-karlsruhe.de/WBS/broker>.
- [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.
- [Frohn et al, 1997] J. Frohn, R. Himmeröder, P.-Th. Kandzia, G. Lausen, and C. Schleppehorst: FLORID - A Prototype for F-Logic, In: *Proceedings of the International Conference on Data Engineering (ICDE, Exhibition Program)*, Birmingham, 1997.
- [Genesereth et al., 1997] M. R. Genesereth, A. M. Keller, and O. M. Duschka: Infomaster: An Information Integration System. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Tucson, AZ, May 1997.
- [Gruber, 1993] T. R. Gruber: A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*, 5(2), 1993.
- [Kandzia & Schleppehorst, 1997] P.-T. Kandzia and C. Schleppehorst: DOOD and DL - Do We Need an Integration. In *Proceedings of the 4th KRDB Workshop*, Athens, Greece, August 30, 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.
- [Kifer & Lozinskii, 1986] M. Kifer, E. Lozinskii: A Framework for an Efficient Implementation of Deductive Databases. In *Proceedings of the 6th Advanced Database Symposium*, Tokyo, 1986.
- [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
- [Levy et al., 1996] A. Y. Levy, A. Rajaraman, and J. J. Ordille: Query-Answering Algorithms for Information Agents. In *Proceedings of the AAAI-96*, Portland, Oregon, August 4-8, 1996.
- [Lloyd & Topor, 1984] J. W. Lloyd and R. W: Topor: Making Prolog more Expressive, *Journal of Logic Programming*, 3:225—240, 1984.
- [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.
- [Ontobroker] <http://www.aifb.uni-karlsruhe.de/WBS/broker>
- [RDF] Resource Description Framework, <http://www.w3.org/Metadata/RDF/Group/WD-rdf-syntax>
- [Skuce,1997] D. Skuce: Hybrid KM: Integrating Documents, Knowledge Bases, Databases, and the Web. In: *Proceedings of AAAI Spring Symposium on Artificial Intelligence in Knowledge Management*, 1997. URL: <http://ksi.cpsc.ucalgary.ca/AIKM97/AIKM97Proc.html>
- [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.

[Van Gelder et al., 1991] A. Van Gelder, K. Ross, J. S. Schlipf: The Well-Founded Semantics for General Logic Programs, *Journal of the ACM*, 38(3): 620—650, 1991.

[Wiederhold, 1992] G. Wiederhold: Mediators in the Architecture of Future Information Systems, *IEEE Computer*, 25(3):38—49, 1992.

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