

Ontobroker: How to make the WWW Intelligent

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 can be viewed as the largest knowledge base that has ever existed. However, its support in query answering and automated inference is very limited. We propose formalized ontologies as means to enrich web documents for representing semantic information to overcome this bottleneck. Ontologies enable informed search as well as the derivation of new knowledge that is not represented in the WWW. The paper describes a software tool called Ontobroker that provides the necessary support in realizing this idea. Basically it provides 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. (see [Mauldin, 1997], [Selberg & Etzioni, 1997]). 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.

Deriving semantic information automatically from sentences in natural language is still an unsolved problem. Inference by keyword search may deliver some results but it also leads to a lot of unrelated information and at the same time it may miss a lot of important information (i.e., precision and recall are low). [Luke et al., 1996] and [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

([Swartout et al., 1996], [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.

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-used standard for representing knowledge about all possible subjects. Therefore, we used the *metaphor of a newsgroup* in [Fensel et al., 1997] 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 knowledge to enable automatic inference based on the shared ontology. We create the term *Ontogroup* to refer to such a group of web users who agree upon a joint ontology.

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. [Euzenat, 1996], [Skuce,1997]).

The contents of the paper is organised as follows. First we provide a general motivation of our approach in section 2. Then in section 3, we sketch the general architecture of the Ontobroker and its different parts. The languages used to represent ontologies, to formulate queries, and to annotate web documents with ontological information are discussed in section 4. In section 5 we discuss the three main tools of Ontobroker: its graphical and logic-based query interface, its inference engine, and its webcrawler. A discussion of the possibilities and limitations of Ontobroker is provided in section 6 and related work and conclusions are given in section 7.

2 The Bottlenecks of the WWW that are Bypassed by Ontobroker

The WWW provides huge amounts of information in informal and semi-structured representations. This is one of the key factors that enabled its incredible success story. The representation formalisms are simple and retain a high degree of freedom in how to present the information. In consequence, we strictly follow the basic design paradigm of web documents. Our approach does not restrict the information providers in deciding *how* they want to represent their information. They are able to choose and modify the formats of their web documents without being hampered by using our techniques. Also, we did not introduce a new and difficult language for defining semantics but introduced a small extension of HTML. We will discuss later how this extension relates to emerging web standards like XML

[XML] and RDF [RDF].

Having said that our approach incorporates the basic paradigm that made the WWW a success we will now sketch some shortcomings of the WWW that motivated our approach. Freedom in information representation and simple representation formalisms cause serious bottlenecks in accessing information from the web because of the growing amount of information it contains (i.e., the same factors that led to its success may also hamper its further development). Basically there are two different search techniques available at the moment: human browsing through textual and graphical representations following hyperlinks and keyword based search engines that retrieve further hyperlinks for this browsing process. The query answering and inference service of the WWW is very limited when compared to relational or deductive databases that enable precise queries and inference service for deriving new knowledge. In the following we will discuss some examples that illustrate limitations of current WWW access.

- Imagine that you want to find out about the research subjects of a researcher named *Smith* or *Feather*.¹ Consulting a search engine will result with a huge set of pages containing the key words *Feather*. Preciseness, recall, and presentation are limited. All pages containing the string *Feather* are returned and many of these pages are completely irrelevant. The important page may be missing. Imagine that he has a headline like “*Topics of interest*” at the page that is imported by a framed homepage. Such a page does not contain any of the assumed keywords. Even if the pages of the person are identified it requires a significant human search effort to investigate these pages until the page that contains the required information has been found. Even search engines specialized in retrieving homepages of persons cannot make use of the information that he is a researcher and are specialized in retrieving address information and not in making sophisticated queries about what a person is doing etc.
- The format of query responses is a list of hyperlinks and textual and graphical information that is denoted by them. It requires human browsing and reading to extract the relevant information from these information sources. Remember, we were looking for the research subjects of Mr. *Feather*. We would like to get a list of research topics like: “*World Wide Web, Ontologies, Knowledge Acquisition, Software Engineering*“. However, it requires further human extraction to retrieve this information. This burdens web users with an additional loss of time and seriously limits information retrieval by automatic agents that miss all common sense knowledge required to extract such informations from textual representations. A further consequence is that the outcome of a web query cannot directly be processed by another software tool because a human has to extract and to represent it in a way that fits some standard representation.
- Still, the above mentioned problems are rather trivial compared to queries that refer to the content of several pages. Imagine that you want to find the research subjects of a research group. You have to figure out whether this is written on a central page or whether each researcher enumerates them on his pages. Then you have to determine all members of this research group and go through all their pages. The required search effort and lack of recall make such queries impractical for a large, distributed and heterogeneous group of people (i.e., web sources). Imagine that you want to extract the research topics of all researchers who also work on ontologies. This shows fairly clearly

¹. Not to mention the case where his name is *Cook*.

that the current information access to the WWW cannot handle information that is distributed at several locations and pages.

- Finally, each current retrieval service can only retrieve information that is represented by the WWW. This sounds trivially true, but it significantly limits query answering capability. Imagine that *Feather* writes on his homepage that he cooperates with another researcher *E. Motta* on investigating formal specifications of problem-solving methods. However, you will completely miss this information for *E. Motta* if he does not repeat the information (with the reverse direction) on his homepage and you are only consulting his page. However, an answering mechanism that can make use of the implicit symmetry of cooperation could provide you with this answer. Similarly, because *Smith* is a researcher and he cooperates on research issues with *E. Motta* one can derive that *E. Motta* is also a researcher and may want to receive this information even if it is not explicitly stated on one of *E. Motta*'s pages. Here we would make use of a type information of a relationship.

Summing up our discussion we identify the following limitations of information access of the WWW that we will bypass with our approach:

- We want to use semantic information for guiding the query answering process.
- We want to enable answers with a well-defined syntax and semantics that can directly be understood and further processed by automatic agents or other software tools.
- We want to enable a homogeneous access to information that is physically distributed and heterogeneously represented in the WWW.
- We want to provide information that is not directly represented as facts in the WWW but which can be derived from other facts and some background knowledge.

3 The Architecture of Ontobroker

The general architecture of the ontology-based brokering service Ontobroker is shown in Figure 1. It 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.

The user communicates with the inference engine using a query interface. Two different interfaces are available: a graphical one, that hides the syntax of the query language and enables a direct access to the ontology, and a conventional, text base interface, where a user can directly type in queries in the query language. Both interfaces are realized with internet/intranet technologies. The graphical interface is realized as a Java-applet, and thus it is executable within modern web browsers like Netscape and Explorer. It uses specific techniques for information visualisation [Lamping & Rao, 1994][Lamping et al., 1995], that presents the ontology in a accessible way and exploits the structure of the query language. With these ingredients it enables a guided query formulation. The textual query interface of Ontobroker is realized as a HTML-form, where a user can type in the query as plain text.

The *inference engine* receives the query of a client and uses two information sources for

deriving an answer: It uses the ontology chosen by the clients and it uses the facts that were found by the Ontocrawler in the WWW. The basic inference mechanism of the inference engine is the derivation of a minimal model of a set of Horn clauses (see section 5.2 for more details). However, the language for representing ontologies is syntactically enriched. First, ideas of [Lloyd & Topor, 1984] were used to get rid of some of the limitations of Horn logic without requiring a new inference mechanism. Second, languages with richer epistemological primitives than predicate logic are provided. Frame logic [Kifer et al., 1995] is used as the representation language for ontologies. It incorporates objects, relations, attributes, classes, and subclass-of and element-of relationships within a first-order semantic framework. To improve the accessibility of our service we are currently realizing a translator to Ontolingua [Farquhar et al., 1997], see section 4.1 for more details.

Ontocrawler searches through a fragment of the WWW that makes use of one of the ontologies and collects these knowledge fragments and it implements a wrapper that translates annotated web documents into facts formulated in the representation language. Neither the inference engine nor the query client have to be aware of the syntactical way, the facts are represented in the web. The Ontocrawler provides this abstraction mechanism. Only a knowledge *provider* has to use the annotation language. Each provider of an ontologically annotated knowledge portion has to do an index of his annotated documents and he has to use the annotation language and an ontology of the Ontobroker to annotate these documents.

The answering of queries and the collection of facts from the web are decoupled in the current architecture. The collection of facts is done independently from the queries. For efficiency reasons in query answering, the Ontocrawler periodically caches the annotations. In addition, it can be started by a knowledge *provider* that submits new knowledge fragments to the web. Our experience will show whether a stronger coupling mechanism will be required.

Subsequently we will discuss the different languages and tools that are provided by the Ontobroker.

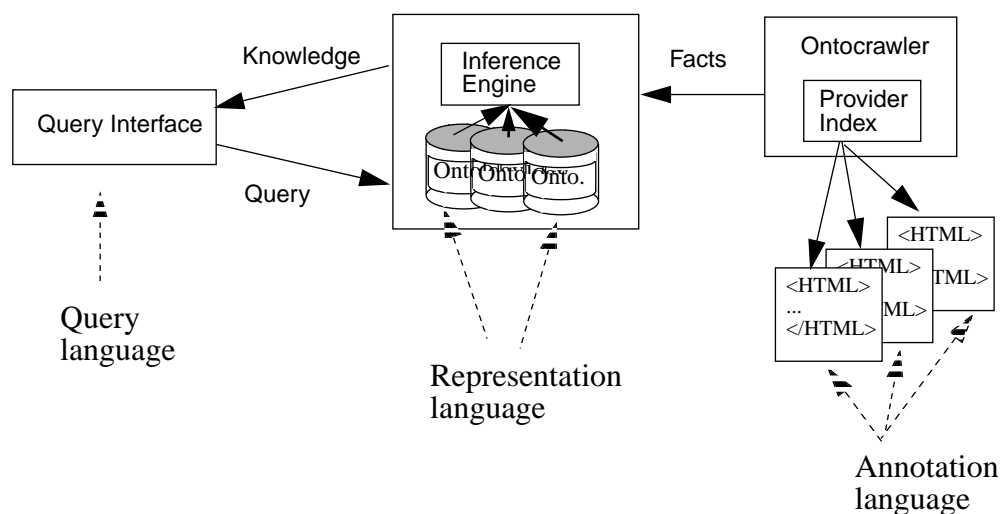


Fig. 1 The architecture of the Ontobroker

4 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.

4.1 The Representation Formalisms for Ontologies

Many different formal approaches for defining ontologies have emerged, among them e.g. approaches based on Description Logics; Ontolingua and conceptual graphs that use syntactical extensions of Full Predicate Logic; and Horn logic oriented approaches from deductive databases and logic programming. Therefore, we had to make a choice between these different paradigms. One of the key requirements for our design choice was the ability to provide an effective and efficient inference service enabling query answering.

Ontolingua [Gruber, 1993] (based on KIF [Genesereth & Fikes, 1992]) is intended for portable ontology specifications. It does not provide any inference support. In addition, an inference engine would be very inefficient, because the language is based on full predicate logic, so a full fledged theorem prover would be needed to support this language.

Systems based on Description Logics (like LOOM [MacGregor, 1990]) provide mainly two kinds of inference: consistency checking of ontologies (resp. subsumption checking) and classification of instances [Nebel, 1990]. However, our application scenario is query answering over instances. Techniques from deductive databases evolved from relational databases and are explicitly designed for this type of query answering facilities. These inference techniques for deductive databases and logic programming are developed for returning ground instantiations of predicates as answers.

Since there are effective 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 (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.

4.1.1 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 Logics², expressing the ontology in

² [Badea, 1997] recently proposed to extend Description Logics a step into the direction of Frame-Logic.

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 . This, however, is more a syntactic convention. Because there are so many different characterizations of *part-of* (e.g. transitive and non-transitive) the exact behaviour of part-of is application and domain dependent.
- Relations: predicate expressions like $p(a_1, \dots, a_n)$ can be used as in usual logic based representation formalisms, except that not only terms can be used as arguments, but also object expressions.

Compared with the original Frame-Logic of [Kifer et al., 1995] we skipped functional attributes. Functional attributes define a kind of integrity constraint. Two syntactically different values of a functional attribute applied to the same object must either be semantically equal or the set of axioms is not well-defined. However, knowledge provided by the web may be redundant and slightly differently presented by different knowledge providers. Therefore we decided not to include any integrity constraints in the basic primitives of our language.

4.1.2 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 rule is an expression of the form:

$$head \leftrightarrow body,$$

where the *head* and *body* are just conjunctions of elementary expressions. A double rule can be understood as an abbreviation of

$$head \leftarrow body \text{ and } body \leftarrow head.$$

An example of a rules where just predicates (relations) are used as elementary expressions is given below:

$$\text{FORALL } X, Y \text{ subset}(X, Y) \leftarrow \text{set}(X) \text{ AND } \text{set}(Y) \text{ AND FORALL } Z \text{ in}(Z, X) \rightarrow \text{in}(Z, Y).$$

START	← {(Query Rule DoubleRule Fact) }* "EOF."
Query	← ["FORALL" VarList] ("<->" "?->") Formula) ". "
Rule	← ["FORALL" VarList] MoleculeConjunction ("<->" ":->") Formula ". "
DoubleRule	← ["FORALL" VarList] MoleculeConj "<->" MoleculeConj ". "
Fact	← ["FORALL" VarList] MoleculeConj ". "
MoleculeConj	←Molecule {("AND" ",") Molecule }*.
Formula	← ("EXISTS" "FORALL") VarList Formula. Formula ("AND" "OR" "<->" "<->" ">->") Formula. "NOT" Formula . "(" Formula ")" Molecule .
VarList	← Identifier {"," Identifier }*.
Molecule	← FMolecule PMolecule.
FMolecule	← Reference Specification.
Reference	← Object ([Specification] ("#" "##") MethodApplication)*.
Specification	← (":" "::" "<:") Object ["[" [ListOfMethods] "]"]) "[" [m=ListOfMethods] "]" .
PMolecule	← Identifier ["(" ListOfPaths ")"] .
Path	← Object [Specification] (("#" "##") MethodApplication [Specification]) *.
ListOfPaths	← Path ("," Path)*.
Object	←ID_Term "(" Path ")"
Method	← MethodApplication MethodResult .
ListOfMethods	← Method ("," Method)*.
MethodApplication	← Object ["@(" ListOfPaths)"] .
MethodResult	← (">->" "=>" "*->" ">>" "=>>" "*->>") Path). (">>" "=>>" "*->>") "{" ListOfPaths "}" .
IDTerm	← IDENTIFIER ["(" ListOfPaths ")"] INTEGER_LITERAL FLOATING_POINT_LITERAL STRING_LITERAL "[" [ListOfPaths ["]" Path]] "]" .

Fig. 2 EBNF-syntax of the representation language.

Examples of double rules are given in Table 1. An EBNF syntax description of the complete representation language is given in Figure 2.

4.1.3 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.

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

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; worksAtProject ==> Project; headOfGroup ==> ResearchGroup]. AcademicStaff[supervises ==> PhDStudent]. 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].

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.

4.1.4 Translation of Ontologies

To allow the usage of ontologies defined in different languages, we plan to provide translators to other languages. Currently we investigate a translator from and to Ontolingua because from Ontolingua translators to other representation languages (e.g. LOOM, KIF etc.) are already available (cf. [Farquhar et al., 1997]). Fortunately, the main building blocks for formulating ontologies are very similar, if we are using the frame-ontology of Ontolingua. So we are able to sketch a translation based on syntactical transformation from F-Logic to Ontolingua using the Frame-Ontology. For the translation from and to Ontolingua we have mainly to provide support for the class definitions, the attribute definitions and the rules. Class definitions of Frame-Logic are translated into class definitions in Ontolingua (see first row of Table 2). Subclass expressions need an additional subclass-expression (Table 2 third row). Attributes are translated into relations of Ontolingua, therefore we need an additional relation definition (Table 2 second row). If rules are defining attribute values of classes, they have to be incorporated into the relation definition (sometimes at more than one attribute

definition (Table 2 second and sixth row)).

Table 2. Translating between F-Logic and Ontolingua

F-Logic Definitions	Ontolingua Definitions
Object[.].	(define-class Object (?Object))
Person :: Object[publication ==>> Publication] FORALL Person1, Publication1 Publication1:Publication[author ->> Person1] <-> Person1:Person[publication ->>Publication1].	(define-class Person (?Person) :def (and (Object ?Person) (has-some ?Person has-publication))) (define-relation has-publication (?Person ?Publication) :def (and (Person ?Person) (Publication ?Publication) (has-author ?Publication ?Person)))
Employee :: Person.	(define-class Employee (?Employee) :def (Person ?Employee))
AcademicStaff :: Employee.	(define-class AcademicStaff (?AcademicStaff) :def (Employee ?AcademicStaff))
Researcher :: AcademicStaff [cooperatesWith ==>> Researcher]. FORALL Person1, Person2 Person1:Researcher [cooperatesWith ->> Person2] <-> Person1:Researcher [cooperatesWith ->> Person1].	(define-class Researcher (?Researcher) :def (and (AcademicStaff ?Researcher) (has-some ?Researcher cooperatesWith))) (define-relation cooperatesWith (?Res1 ?Res2) :def (and (Researcher ?Res1) (Researcher ?Res2) (cooperatesWith ?Res2 ?Res1)))
Publication:::Object[author ==>> Person].	(define-class Publication(?Publication) :def (and (Object ?Publication) (has-some ?Publication has-author))) (define-relation has-author(?Publication ?Author) :def (and (Publication ?Publication) (Person ?Author) (has-publication ?Author ?Publication)))

Currently, we are working on mechanising this translation process.

4.2 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

$O:C[A \rightarrow V]$

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.

FORALL $R \leftarrow R:Researcher$.³

This query asks for all known objects, which are instances of the class 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 $Obj, LN, EM \leftarrow$
 $Obj:Researcher[first\ Name \rightarrow Richard; last\ Name \rightarrow LN; email \rightarrow EM]$.

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:

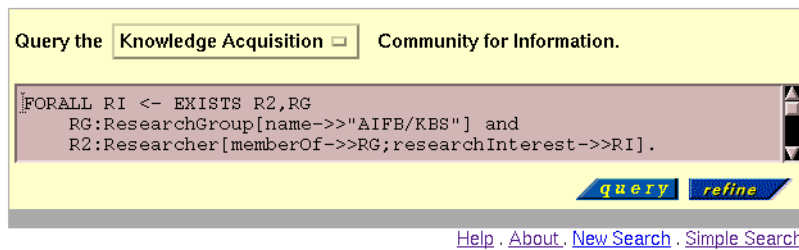
FORALL $Obj, CP \leftarrow$
 $Obj:Researcher[last\ Name \rightarrow "Motta"; cooperates\ With \rightarrow CP]$.

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. 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 at the WWW.

Ontobroker can also be used to collect distributed information. The query in Figure 3 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

³. A query is a rule without a head.



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. 3 The textual query interface.

`FORALL C <- C::Object[]`

asks for all classes defined in the ontology which are subclasses of the class *Object*. The query

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

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

The query interface of Ontobroker either directly uses the query language or provides further abstractions by query form and graphical representations that significantly simplify the query process for naive users (see section 5.1).

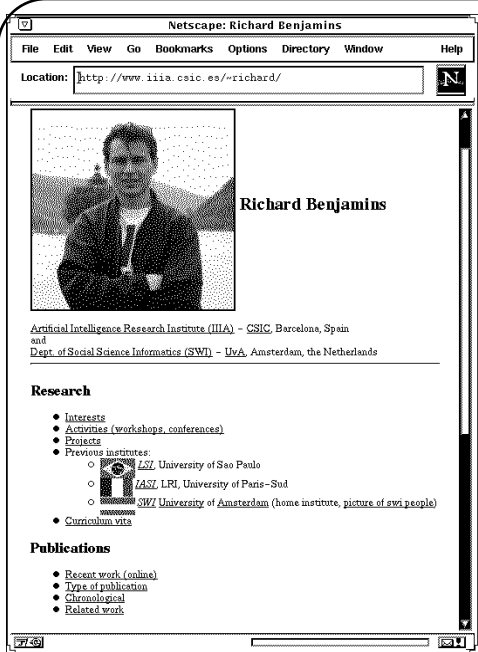
4.3 The Provider Side: Annotating Web-Pages with Ontological Information

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.⁴ First, we will provide the general idea. Then we introduce some more details that significantly economise the annotation effort for knowledge providers. For an overview of our extensions to HTML see the EBNF-syntax definition of the

anchor	←	“<“ “A“ attributes “>“ [body] ““.
attributes	←	attribute*.
attribute	←	“name“ “=” CDATA “href“ “=” URL “rel“ “=” CDATA “rev“ “=” CDATA “title“ “=” CDATA “onto“ “=” ontoInfo.
ontoInfo	←	“\“ “ ontoInfo1 “\“ “\“ “ ontoInfo1 “\“.
ontoInfo1	←	host “:“ class host “[“ ontoAttribute “=” value “]“ relation “(“ object (“,“ object)* “)“.
host	←	object.
value	←	object.
class	←	ID.
relation	←	ID.
object	←	URL CDATA.

Fig. 4 EBNF-syntax of the annotation language.

annotation language as provided in Figure 4. An extract from an example page is given in Figure 5.



```

<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.

4. We did not make use of the *extensible Markup Language (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 6.

4.3.1 The general idea

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.

The names of classes and relationships are already provided by the ontology that is defined using the representation language (see section 4.1). However, names that are used to denote instances of classes, attribute values, and relationships have to be defined using the annotation language. We need *identifiers* or *handles* to identify objects. This handle has to be globally unique and unambiguous. We chose the URL (Uniform Resource Locator, [URL]) of the WWW to denote objects, which is quite natural as we will see in the examples below. The URL defines by definition a unique way to determine which entity is referred to in the Internet.⁵ The identifiers are called object in the EBNF of Figure 4.

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

- 1) An object 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>
```

The anchor tag is usually used to define named locations within a web page and hypertext links to other locations in the WWW. Thus, it contains the attributes `name` and `href` to fulfill these purposes, respectively. For ontologically annotating a web page we added another attribute to the syntax of the anchor tag, namely the `onto` attribute. All three attributes may contain information which describes objects relevant to the Ontobroker.

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

⁵ However, it is less unique as we would wish. For example, `http://www.aifb.uni-karlsruhe.de/WBS/dfe/` and `http://www.aifb.uni-karlsruhe.de/WBS/./WBS/dfe/index.html` denote the same address. Even worse, `http://www.aifb.uni-karlsruhe.de/~dfe` is another way to point to the same address. Equalities of the former type could be handled by general equality axioms. Equalities of the latter type are introduced by web servers which are not knowable from the outside.

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.

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

This line states that the object denoted by the handle `'http://www.iii.csic.es/~richard'` has the value `'mailto:richard@iii.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 `Objn` this definition looks like this:

```
<a onto="REL(Obj1, Obj2, Obj3, ..., Objn)" > ... </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. In the following section we will provide some examples that illustrate the key ideas behind these short-cuts.

4.3.2 Short-cuts that Improve Convenience⁶

To define on a web page that an object is an instance of a class, e.g. that Richard Benjamins is a `Researcher`, one uses the following kind of annotation:

```
<a onto="HOST:CLASS" name=NAME href=HREF> ... </a>
```

The term `CLASS` has to be substituted by a concrete class as defined in the underlying ontology, e.g. `Researcher`, `Workshop`, `Article`. The object denoted by the handle `HOST` then is defined as an instance of this class. There are several possibilities to define a host object. There are special keywords which represent short-cuts for certain URLs.

- If `HOST` equals the string `page`, the URL of the currently processed HTML page is defined as the handle of the host object.

Example:

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

This tag defines the URL of the page which the tag appears on as an instance of the class `Person`. By that, the clumsy original URL can be avoided, and the ontological information becomes more readable.

⁶ While describing the annotation syntax all lower case and special characters are to be read literally (including the different quotes), all upper case terms have to be replaced by actual objects, URLs etc.

- If HOST equals the string tag or the string name, the current tag, i.e. the named region defined by the current anchor tag becomes an instance of CLASS. In this case the tag has to contain a name-attribute.

Example:

```
<a onto="tag:Article" name="Fensel at al. 97">
```

```
    Ontology Groups: Semantically enriched subnets of the WWW.</a>
```

This tag defines the named region as an object of class Article. Assuming the page of this tag is "http://www.aifb.uni-karlsruhe.de/WBS/dfe/publications97.html" this would create a new object with handle "http://www.aifb.uni-karlsruhe.de/WBS/dfe/publications97.html#Fensel at al. 97" which belongs to the class Article.

- If HOST equals the string href, the page to which the anchor tag refers to, i.e. the value of the tag's href-attribute becomes the host object. To work correctly an href-attribute has to be given within the anchor tag.

Example:

```
<a onto="href:Workshop" href="http://ksi.cpsc.ucalgary.ca/KAW">
```

```
    KAW 98, Banff, Canada </a>
```

This tag defines the page the hyperlink refers to as an object of class Workshop. In this example a global URL has been given so that this URL becomes the object handle for the newly created object. When using this variant, the part of the page embraced by <a ...> and becomes clickable.

- If none of the above equivalencies hold, HOST is assumed to contain a concrete object handle, normally an actual URL. This URL then is defined as belonging to the named class.

Example:

```
<a onto="http://ksi.cpsc.ucalgary.ca/KAW:Workshop">
```

```
    KAW 98, Banff, Canada </a>
```

This tag defines the page specified by the URL as an object of class Workshop, and thus results in exactly the same fact as the previous example. This variant should be used if the named region should not be clickable.

To improve convenience in defining attribute values and relationships similar techniques are provided. The same special keywords (page, href, name, and tag) are supported.

Example:

```
<a onto="page[href=href]" href="http://www.iiia.csic.es/">
```

```
    IIIA - Artificial Intelligence Institute.</a>
```

In this tag the already defined keyword page and href have been used. This tag defines the affiliation attribute of the object denoted by the URL of the current page. Its value is taken from the anchor-tag's href-attribute.

Because several attributes contain only plain text and no complete objects, another keyword is introduced: "body". This keyword enforces the marked portion of the web page (contained between <a ...> and) to be taken as the attribute value.

Example:

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

This annotation 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.

Finally, also in the annotation of relationships on web pages special keywords and actual URLs may be used freely. Thus an example relationship may look like this.

Example:

```
<a onto="appointment(  
  page,  
  'http://aifb.uni-karlsruhe.de/WBS/~dfe',  
  'October, 17th 1997',  
  body)" >  
  (KA)2-Initiative: The Inaugural Meeting </a>
```

In this example a relationship called `appointment` is defined. It defines a relationship between four attributes respectively, two persons, a given date, and a certain title, respectively. The first object denotes the current page, e.g. Richard Benjamins' home page, the second denotes another object denoted by an actual URL, the third is given as a literal string, and the last one refers to the text contained in the body of the tag, namely the title of the meeting.

5 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.

5.1 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:

- The user has to know the syntax of the query language. That means he has to exactly type in the syntactic expressions. However, user don't learn a query language just for one search engine. Furthermore people are making mistakes when typing text, so a good interface should lead the user and should help to avoid such mistakes.
- The user also has to know the ontology, when formulating a query. This is even a bigger hassle than the missing knowledge of the query language: without knowledge of the

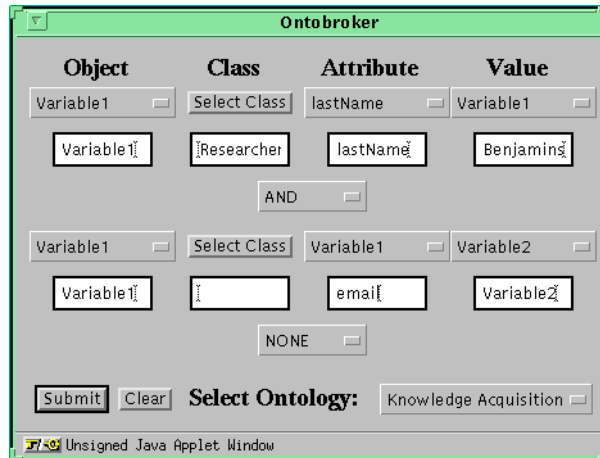


Fig. 6. The advanced query interface.

ontology one cannot formulate a useful query, because all knowledge is organized by the ontology. So an interface should present the ontology and allow an easy access to it.

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 6 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 & Rao, 1994][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 7). 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. 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>.

Based on these interfaces Ontobroker derives automatically the query in textual form and present the result of the query (see Figure 8).

5.2 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 3 gives an idea of how this translation is performed, but it does not catch the complete

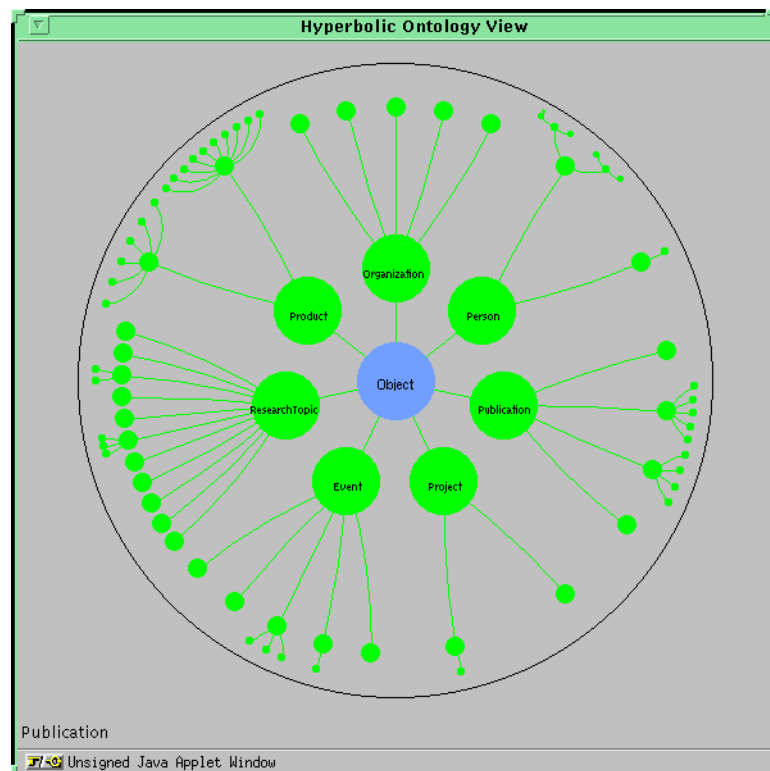


Fig. 7. The hyperbolic ontology view.



Fig. 8. The textual query interface.

Table 3. 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	$\text{part_of}(O_1, O_2)$

translation: e.g. more complex Frame-logic expressions like

$$O[A \Rightarrow V : C[AA \Rightarrow VV]]$$

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 [Lloyd, 1987] via

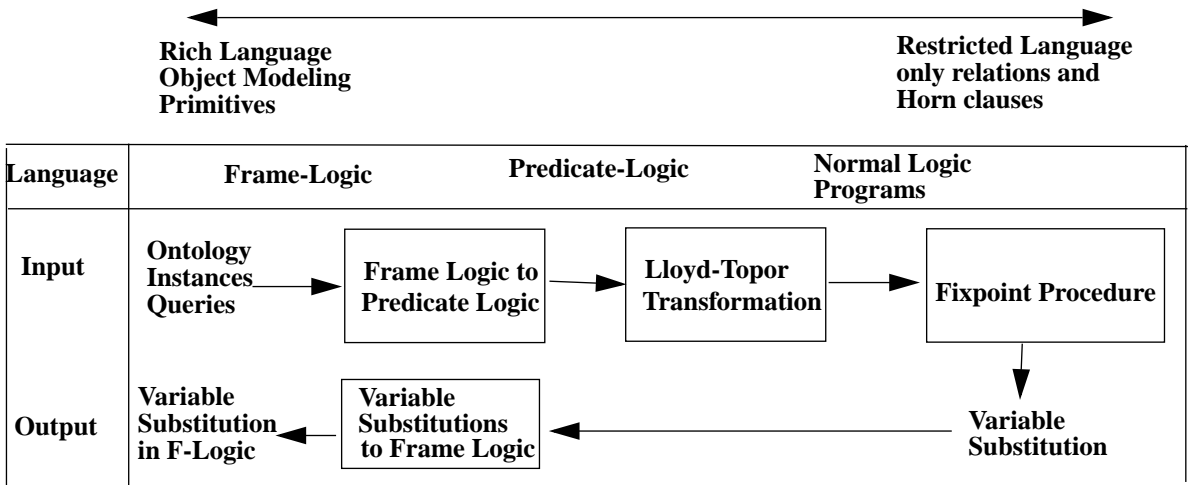


Fig. 9. Stages and Languages used in the Inference Engine

a Lloyd-Topor transformation [Lloyd & Topor, 1984]. The entire translation process is surveyed in Figure 9.

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. For an example see Figure 10. Although the clauses have a reasonable model

$$\{O_1/l; O_2/m; V/n\}$$

this can not be computed using a stratified semantics. 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].

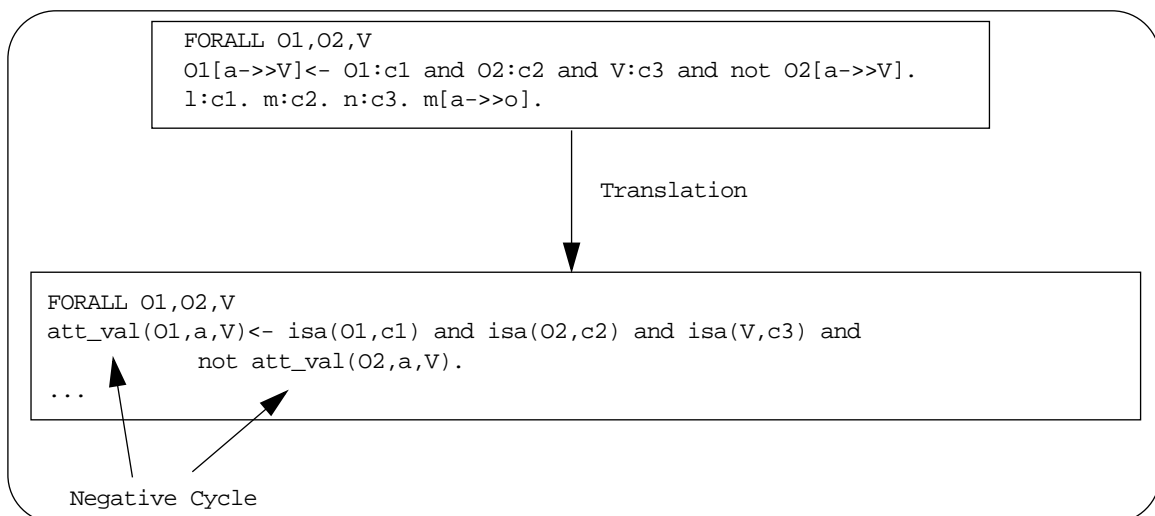


Fig. 10. Non stratified clauses due to negative cycle

The inference engine is completely written in Java [Javasoft] and therefore not bound to a particular system. However for performance reasons we generated a machine code version of it using the Toba Java-Bytecode to C compiler [Proebsting et al., 1997].

5.3 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.

6 Discussions of the Approach and Future Work

Providing information and knowledge via the Ontobroker requires two time-consuming activities: designing an ontology and annotating web documents. Both are serious bottlenecks that may hamper the success of Ontobroker. In the following, we discuss both problems.

Designing ontologies is a time consuming activity because it aims for a formal and consensual model of some aspect of reality. However, building such a model pays back in several dimensions much beyond only improving the web presentation of documents. It can be used by companies and organisations as a reference model for their internal data and informations. It can be used by standardization committees to establish standard for representing information about some area. Therefore, these ontologies found increasing popularity for supporting knowledge management in different areas. We initiated together with colleagues from other research groups the *Knowledge Annotation Initiative (KA)*² to get better insights into the merits and difficulties of establishing such ontologies ([Benjamins et al., 1998], [Benjamins & Fensel, 1998]). Part of this initiative is to establish an ontology that can be used to describe the different research groups in knowledge acquisition, their organisational informations, their products, results, and subjects. This initiative raises a couple of interesting questions at different levels: what are the necessary tools to support ontological engineering in a heterogeneous and distributed environment and how to organize the social process in establishing consensus and in attracting the critical mass of participants. Meanwhile a core ontology has been established and a broad range of research groups participate.

The creation, usage and maintainability of knowledge are the key problems that need to be solved for *knowledge management* in enterprises (cf. [Davenport, 1996]). An ontology can be used to support all of these processes. More specific, Ontobroker can be used to support usability and maintainability of these documents. One strength of Ontobroker is the tight coupling of textual, semiformal and formal knowledge which is identified as a main requirement for successful knowledge management (see e.g. [Euzenat, 1996], [Skuce, 1997]). The textual and semiformal knowledge is directly coupled with annotations that describe their formal semantics. Therefore, maintenance need not deal with problems introduced by redundancy (i.e., representing the same information at different places, one time as textual knowledge one as formalized knowledge). In addition, Ontobroker integrates these semiformal knowledge with inference rules expressed in the ontology. Automatic processing of these knowledge or coupling with automatically derived knowledge bits from other sources are

enabled. Currently we apply Ontobroker in the project *Work Oriented Design of Knowledge Systems (WORKS)* for developing a knowledge management system for industrial designers for decision-support in ergonomic decisions. Pages with ergonomic knowledge are annotated, with the following goals: first to make them retrievable for users, and second, to use the knowledge also for inferences of the system. In this case the knowledge (often numerical data) is provided as an input (and output) to problem solving methods for e.g. parametric design.

Annotating web documents with ontological information is much easier to do. A trained person with some basic HTML knowledge is able to annotate ca. five pages an hour (ca. thousand per month). Still, we would like to provide a more sophisticated tool that supports this process. Currently, annotations have to be written with text editors. However, as for the query interface one could make use of a graphical representation of the ontology and use it for a click-and-paste process in producing annotation. Another possibility for stable web sources is to replace the annotation effort by writing wrappers. [Ashish & Knoblock, 1997] mention information sources like the CIA World Fact Book or the Yahoo listing of countries. These sources use a stable format for information representation that can be used to derive wrappers that extract this information. Such a wrapper can be used to directly derive the factual knowledge that is used by the inference engine of Ontobroker. In this scenario a wrapper replaces the annotation process and the translation process from annotations to facts.

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 introduces the extensible Markup Language (XML) [XML] as a new standard for expressing the structure of web documents. XML is a language to define the syntax of structured documents and to allow the communication of several applications due to a common specification of the document syntax. To allow the annotation of XML documents the W3C 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. That approach shows a number of similarities with Ontobroker, because both approaches aim at machine-understandable content information and enable automated processing of web resources. Both use URLs to represent entities in the WWW. Both use attribute-value pairs to define properties of objects. But 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. Therefore, we will join this standard enabling other agents to read our meta information. 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. That is, this knowledge may not only be used by Ontobroker to answer direct questions of a human user but it will be available for all automated search mechanisms that can read RDF and that can make use of an ontology (cf. [Ambite & Knoblock, 1997]).

7 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. In that way, 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], [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. 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 closed to ours is SHOE that introduced the idea of using ontologies to annotate information in the WWW (cf. [Luke et al., 1996], [Luke et al., 1997]). 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 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 (see [Kandzia & Schleppehorst, 1997], [Fensel et al., to appear] for comparisons of both representation and reasoning paradigms).

One can situate Ontobroker in the general context of approaches that support the integration of *distributed* and *heterogeneous* information sources like CARNOT [Collet et al., 1991], Infomaster [Genesereth et al., 1997], Information Manifold [Levy et al., 1996], HERMES [Subrahmanian et al., 1995], SIMS [Arens et al., 1993], and TSIMMIS [Papakonstantinou et al., 1995]. 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 assumption 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.

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

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

- [Angele, 1993] J. Angele: *Operationalisierung des Modells der Expertise mit KARL*, 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.
- [Badea, 1997] L. Badea: Reifying Concepts in Description Logics. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, August 23-29, 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.
- [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.
- [Collet et al., 1991] C. Collet, M. N. Huhns, and W.-M. Shen: Resource Integration Using a Large Knowledge Base in Carnot, *IEEE Computer*, December 1991.
- [Davenport, 1996] T. Davenport: *Some Principles of Knowledge Management*, URL:<http://www.bus.utexas.edu/kman/kmprin.htm>, 1996.
- [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
- [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., to appear] D. Fensel, M.-C. Rousset, and S. Decker: Workshop on Comparing Description and Frame Logics, to appear in *Data and Knowledge Engineering*.
- [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.
- [Genesereth & Fikes, 1992] M. R. Genesereth and R. E. Fikes: *Knowledge Interchange Format, Version 3.0, Reference Manual*, Report Logic-92-1, Stanford University, June 1992.
- [Gruber, 1993] T. R. Gruber: A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*, 5(2), 1993.
- [Javasoft] Javasoft-Homepage: „<http://www.javasoft.com>“, 1997

- [Kandzia & Schlepphorst, 1997] P.-T. Kandzia and C. Schlepphorst: 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.
- [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 & Rao, 1994] L. Lamping and R. Rao: Laying Out and Visualizing Large Trees Using a Hyperbolic Space. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, November, 1994
- [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
- [Lloyd, 1987] J.W. Lloyd: *Foundations of Logic Programming*, 2nd Edition. Springer-Verlag, 1987
- [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., 1996] S. Luke, L. Spector, and D. Rager: Ontology-Based Knowledge Discovery on the World-Wide Web. In *Proceedings of the Workshop on Internet-based Information Systems at the AAAI-96*, Portland, Oregon, August 4-8, 1996.
- [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.
- [MacGregor, 1990] MacGregor: *LOOM Users Manual*, ISI/WP-22, USC/Information Sciences Institute, 1990.
- [Mauldin, 1997] M. L. Mauldin: Lycos: Design Choices in an Internet Search Engine, *IEEE Expert*, January-February 1997. <http://www.lycos.com>.
- [Nebel, 1990] B. Nebel: *Reasoning and Revision in Hybrid Representation Systems*, LNAI 422, Springer-Verlag, 1990.
- [Ontobroker] <http://www.aifb.uni-karlsruhe.de/WBS/broker>
- [Papakonstantinou et al., 1995] Y. Papakonstantinou, H. Garcia Molina, and J. Widom: Object Exchange Across Heterogeneous Information Sources. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, Taipei, Taiwan, March 1995.
- [Proebsting et al., 1997] T. A. Proebsting, G. Townsend, P. Bridges, J. H. Hartman, T. Newsham, and S. A. Watterson: Toba: Java for Applications: A Way Ahead of Time (WAT) Compiler. In *Proceedings of the Third Conference on Object-Oriented Technologies and Systems (COOTS 97)*, 1997, (also „<http://www.cs.arizona.edu/sumatra/toba/>“).
- [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>.
- [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>

- [Subrahmanian et al., 1995] V. S. Subrahmanian, S. Adali, A. Brink, J. J. Lu, A. Rajput, T. J. Rogers, R. Ross, and C. Ward: *HERMES: A Heterogeneous Reasoning and Mediator System*, Technical Report, University of Maryland, 1995.
- [Swartout et al., 1996] B. Swartout, R. Patil, K. Knight, and T. Russ: Toward Distributed Use of Large-Scale Ontologies. In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW-96)*, Banff, Alberta, Canada, November 9-14, 1996.
- [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.
- [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>.