, , 1–49 ()

OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies

EDUARDO MENA jibmenie@si.ehu.es
IIS department, University of Zaragoza, Spain

ARANTZA ILLARRAMENDI jipileca@si.ehu.es
LSI department, UPV-EHU, San Sebastián, Spain

VIPUL KASHYAP kashyap@bellcore.com
1G332R, MCC, Bellcore, 445 South St, Morrison NJ 07960

AMIT P. SHETH amit@cs.uga.edu
LSDIS Lab., University of Georgia, Athens, GA 30602

Editor: Athman Bouguettaya

Abstract. There has been an explosion in the types, availability and volume of data accessible in an information system, thanks to the World Wide Web (the Web) and related inter-networking technologies. In this environment, there is a critical need to replace or complement earlier database integration approaches and current browsing and keyword-based techniques with concept-based approaches. Ontologies are increasingly becoming accepted as an important part of any concept or semantics based solution, and there is increasing realization that any viable solution will need to support multiple ontologies that may be independently developed and managed. In particular, we consider the use of concepts from pre-existing real world domain ontologies for describing the content of the underlying data repositories. The most challenging issue in this approach is that of vocabulary sharing, which involves dealing with the use of different terms or concepts to describe similar information. In this paper, we describe the architecture, design and implementation of the OBSERVER system. Brokering across the domain ontologies is enabled by representing and utilizing interontology relationships such as (but not limited to) *synonyms, hyponyms* and *hypernyms* across terms in different ontologies. User queries are rewritten by using these relationships to obtain translations across ontologies. Well established metrics like *precision* and *recall* based on the extensions underlying the concepts are used to estimate the loss of information, if any.

Keywords: Query processing in Global Information Systems, distributed heterogeneous data access, domain ontologies

1. Introduction

Earlier database-centric approaches, including multidatabase or federated database approaches, relied on a consistent way of structuring and manipulating data. Logical integration of the schemas describing the underlying data was used to handle the structural and representational heterogeneity. This involved binding of concepts to the underlying data sets at schema definition time and establishing the relationships between concepts represented by schema objects at schema integration time. The global information infrastructure represented by the Web presents us with a

different challenge. There is no centralized or federated information management as anyone can put up a web page or make other information resources available on the Web independently. It is also impossible for users to be aware of the locations, organization/structure, query languages and semantics of the data in the various information resources because of the dynamic and open nature of such an environment.

Typical query processing approaches on the Web involve a keyword-based and a limited form of attribute-based access, as exemplified by search engines. Some also support concept based searches (e.g., [14]). However, the concept collection(s) on which they are based are not hard-coded and cannot be shared across different systems and user groups.

Several research projects on integration of heterogeneous information (e.g. SIMS [2], IM [22] and InfoSleuth [32]) go one step further and provide query processing approaches based on concepts. In such systems, ontologies are used to provide concise and declarative specification of semantic information. They are also used to describe information content in data repositories independent of the underlying syntactic representation of the data [21]. Although using a single ontology could make the task of integration and semantic interoperation easier, defining a single ontology to serve multiple users and applications in different domains is both practically impossible to create and manage, and hard to use. Furthermore, many ontologies which we refer to as ad-hoc ontologies already exist or can be developed based on classification and metadata model standards in well investigated domains (e.g., bibliography, geographic information, etc.). Exploiting such ontologies is particularly appealing.

When using multiple ontologies, a query formulated using terms in a user selected ontology needs to be translated into terms of other (target) ontologies that describe relevant information. One key impediment in this context however, is that of vocabulary sharing, especially when different domain ontologies are used to describe similar information across domains. This paper discusses the OBSERVER (*Ontology Based System Enhanced with Relationships for Vocabulary hEterogeneity Resolution*) system which addresses this key challenge. It uses multiple pre-existing ontologies to access heterogeneous, distributed and independently developed data repositories [28]. The content of each data repository is described by one or more ontologies expressed using a system based on Description Logics (DLs) [6]. Each data repository is viewed at the level of the relevant semantic concepts. Information requests to OBSERVER are specified as a DL expression based on concepts in a (user) domain ontology. By using a DL system, OBSERVER uses ontological inferences to *classify* the query and determine relevant data repositories, and translates the DL expressions to the local query languages of the data repositories. Mechanisms dealing with partial translations obtained by using *synonym relationships* and incremental enrichment of the answers by combining them were implemented and discussed in [28]. However, the substitution of a term by combinations of *hyponym and hypernym relationships* [26], which result in the change in the semantics of the query, were not considered earlier. In this paper we present a method that performs the translation using an extended set of relationships including (but not limited to)

synonyms, hyponyms and hypernyms. Furthermore, we discuss the crucial issue of estimating the possible loss of information when using relationships other than synonyms.

Our novel contributions with respect to related work (discussed in Section 2) are the following:

a) Management of several ontologies with different vocabularies (the user chooses the most appropriate ontology for her/his needs and the system manages the navigation into other ontologies).

b) Use of semantic interontology relationships.

c) Management of answers that have an associated loss of information (limited by the user).

d) Measurement of the loss of information incurred.

Subsequent to the development of a prototype system that has been tested with multiple pre-existing and new ontologies, an operational system is now in development.

The organization of this paper is as follows. A brief description of related work and a comparison with our approach appears in Section 2. In Section 3 we discuss the motivation for using ontologies and describe the various ontologies used in the OBSERVER prototype. In Section 4 we discuss the components of the OBSERVER system architecture and the broad outline of our query processing approach. In Section 5 we discuss the techniques used to map and retrieve data corresponding to a DL expression based on concepts in a particular ontology. In Section 6 we discuss techniques for translating a query across multiple related ontologies with the help of *interontology relationships*. Translation of a query across ontologies results in changes to the original semantics of the query. In Section 7 we present techniques to estimate the loss of information based on navigation of concepts in different ontologies. Finally, conclusions are discussed in Section 8.

2. Related Work

In this section we present works that are closely related to our approach for query processing in a Global Information System (GIS). We briefly describe their main features and then compare them to our own approach. We also discuss our earlier work and present the enhancements discussed in this paper.

TSIMMIS The TSIMMIS project [7] is primarily focused on the semi-automatic generation of wrappers, translators and mediators that map information in an *object exchange model* to the underlying structured or unstructured data. The DISCO [35] and HERMES [13] projects also address related issues. TSIMMIS considers query processing capabilities of data repositories in order to build the corresponding wrappers. It does not use ontologies to describe data repositories since mediators encapsulate such underlying repositories. Notice that they

take an operational approach of encapsulating the repositories. The lack of a descriptive view of the underlying data makes users depend on mediators just to browse the information contents of repositories. Although the vocabulary sharing problem is not tackled, we use similar techniques to access underlying data repositories.

Carnot The Carnot project [9] used the global (common) ontology *Cyc* to describe the whole information system. For each query expressed in SQL, a graph is generated by using information from a global dictionary. This graph is used by a semantic module which expands it by including sources with relevant information. Finally, an optimal plan is generated and executed for each graph. A key shortcoming with this approach is the difficulty and complexity of managing a large global ontology (more than 50,000 entities and relationships). This has lead researchers including us, to focus on approaches that involve the use of multiple ontologies.

Information Manifold (IM) [22]. This approach involves the development of domain ontologies expressed in a system based on Description Logics, CLASSIC, over which queries are formulated. They also work on limited query capabilities of data repositories and optimization of the number of data repositories used in answering a query. The query language used is conjunctive queries in DATALOG. Another focus on this project has been obtaining complete answers from databases where some tuples are missing in some relations (incomplete databases). Despite the support for multiple domain ontologies in their system, they do not consider the translation of queries across multiple ontologies.

InfoSleuth [32]. This project provides an agent-based infrastructure for information gathering and analysis on a global information infrastructure such as the Web. The agents communicate with each other using a special service ontology and use domain specific ontologies for capturing information. A user expresses his information request based on ontological concepts which are understood by the appropriate agents which communicate with each other to return the appropriate data to the user. Although InfoSleuth supports multiple domain ontologies, it doesn't support vocabulary sharing across domain ontologies.

SIMS [2]. In SIMS, the different information sources are accessed using a system based on Description Logics, LOOM. This work is the most closely related in terms of perspective and approach. This project tackles the vocabulary sharing problem for the cases where there is no loss of information. Although they use a similar technique for rewriting queries in order to get a complete translation of a query from one ontology into another one, they do not deal with answers that have an associated loss of information.

OBSERVER This paper represents a significant extension of our work published earlier. In [28] we deal with answers without loss of information and is comparable to the capability of the SIMS systems as discussed above. In [26] we presented the basic technique used to translate a query from one ontology into

others with an associated loss of information as an alternative to the work presented in [28]. In this paper we explain in detail the combined translation process, where translations without loss represent highly simplified cases of translations with loss. Preliminary work on the process of query translations across multiple ontologies and measuring the loss of information was presented in [27]. This paper significantly extends on the early direction presented in that paper and presents the complete framework, detailed query processing involving the use of multiple ontologies, inter-ontology relationships and estimation of loss of information. A detailed description of techniques for mapping DL expressions to the query languages supported by the underlying data repositories and the consequent data access, retrieval and correlation is also presented.

3. Ontologies

In this section we discuss the motivation for the use of ontologies and the main features of systems based on Description Logics which are used to represent the ontologies in our system. We conclude with a brief discussion on the real world ontologies used in our system that were originally designed from different perspectives and points of view.

3.1. Motivation for the use of Ontologies

In the context of knowledge sharing, the term ontology has been defined to mean a specification of a conceptualization [17]. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with, but is more general than, the usage of ontology as set-of-concept-definitions. From our point of view, an ontology is a set of terms of interest in a particular information domain and the relationships among them. Ontologies and the interontology relationships between them are created by experts in the corresponding domain. They can also represent a particular point of view of the GIS, i.e., they can describe customized domains of advanced users. In our work, ontologies are expressed as DL expressions organized in a lattice and may be considered as semantically rich metadata capturing the information content of the underlying data repositories.

The main purpose of an ontology is to make explicit the information content in a manner independent of the underlying data structures that may be used to store the information in a data repository. Ontologies are thus abstractions and can describe different types of data such as relational tables and textual and image documents. Users should be able to deal with ontologies (semantic information) instead of dealing with multiple heterogeneous data repositories. So, users formulate queries over ontologies and the system has the responsibility of managing the heterogeneity and distribution in the repositories, i.e., an ontology defines a 'language" (its set of terms) that will be used to formulate queries.

Furthermore, ontologies can be seen as commitments between information providers (who define ontologies and their mappings to the underlying data repositories[1]) and

information users. Hence, advanced users or organizations could create their own ontologies not to describe their own repositories but to describe their point of view of the GIS; the information system is thus seen as the only (and huge) data repository.

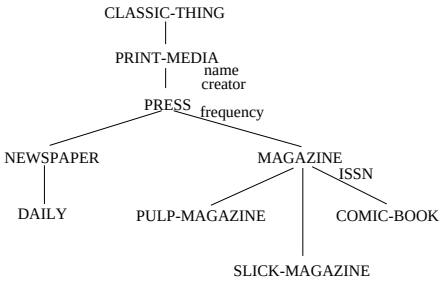
We advocate an approach for dealing with multiple ontologies because managing a global integrated ontology involves administration/maintenance, consistency and efficiency problems that are very hard to solve. A very large ontology may also be very hard for a user to navigate and comprehend. Moreover, it forces users to utilize the vocabulary of that global ontology. However, different ontologies described using different vocabularies can satisfy users' needs in a better way and problems of consistency and efficiency are alleviated. Different ontologies are however not completely orthogonal. Nor is it likely that a user's information need is satisfied by accessing the data repository accessible through mappings associated with a single ontology. To support this, ontologies are virtually linked by interontology relationships. These relationships can be used for two purposes: first, to translate user queries from an ontology into another one, and second, to indirectly support query processing that would access data described by multiple ontologies as we will explain later. Ontologies sharing many similarities or with significant overlap can be organized in clusters corresponding to general knowledge areas like "Animals", "Libraries", "Arts", etc, so that the user can browse through and select the appropriate ontology with ease. Since some clusters can be more general than others, they could be organized in hierarchies in order to make easier the selection of the most appropriate cluster.

We do not discuss in this paper the problem of design and creation of ontologies. Several works dealing with this problem can be found in the literature [31]. Our work is complementary to the above as we re-use pre-existing ontologies for specifying information requests to the GIS.

3.2. Description of Ontologies: Description Logics

In our proposal, ontologies are described using a system based on Description Logics (DL). These systems, also known as Terminological Systems, are descendants of *KL-ONE* [6]. Some systems based on Description Logics are *CLASSIC* [5], *BACK* [37], *LOOM* [23] and *KRIS* [1]. In our prototype we use CLASSIC but any other could be used because the features we use are common to all DL systems. The natural semantics of CLASSIC have been described in [4]. In this section we discuss some of the common features of the DL systems that are relevant to the OBSERVER system. A subset of one of the ontologies in our prototype, WN, whose concept hierarchy and CLASSIC descriptions are shown in Figure 1, left and right parts respectively, is used to illustrate the above mentioned features.

3.2.1. Terms: Concepts and roles. Figure 1 shows two kind of terms: *concepts* (e.g. 'print-media', 'press') and *roles* (e.g. 'name', 'creator'). Concepts represent classes of objects in the domain and roles describe binary relations between objects.



(1)

```
(define-primitive-concept 'print-media 'CLASSIC-THING)
(define-primitive-role 'name)
(define-primitive-role 'creator)
(define-primitive-concept 'press 'print-media)
(define-primitive-role 'frequency)
(define-primitive-concept 'newspaper 'press)
(define-concept 'daily '(AND newspaper (FILLS frequency "daily")))
(define-primitive-role 'ISSN)
(define-concept 'magazine '(AND press (ATLEAST 1 ISSN)))
(define-primitive-concept 'pulp-magazine 'magazine)
(define-primitive-concept 'slick-magazine 'magazine)
(define-primitive-concept 'comic-book 'magazine)
```

(2)

Figure 1. Hierarchy and descriptions of an ontology

Both kinds of terms are created via *terminological descriptions*. Concepts are built from pre-existing terms and a set of operators that allow the construction of *concept descriptions*. Although it is possible to create derived roles by providing a *role description*, we only use simple roles.

Terms can be *primitive* (if their descriptions specify only the necessary conditions) or *defined* (if their descriptions specify both the necessary and sufficient conditions)[2]. For example, as 'magazine' has been described as a defined concept, any instance of 'magazine' will be an instance of 'press' with at least one value for the role 'ISSN', and hence any instance of 'press' with at least one value for 'ISSN' will be automatically classified by the system as an instance of 'magazine'. On the contrary, as 'pulp-magazine' has been described as a primitive concept, although any instance of 'pulp-magazine' is an instance of 'magazine', there can exist instances of 'magazine' that are not instances of 'pulp-magazine'. In addition to concepts and roles, it is also possible to create *instances* (or objects) in a DL system. We do not create instances as they are obtained by retrieving data stored in data repositories underlying each ontology.

3.2.2. Subsumption Mechanism. On the left part of the figure, we can see that terms have been classified in a hierarchy. DL systems use a *subsumption mechanism*, which is based on the term definitions, to determine whether a term is more general than another. In this case the first term is said to *subsume* the other. For example, if a new term 'scientific-magazine' is described as a *defined* concept with the description '(AND press (FILLS frequency "daily") (AT-LEAST 1 ISSN))', it would be classified by the DL system as an immediate child of 'magazine' (see its description in Figure 1). This is done because 'magazine' was created as a *defined* concept and then any concept described as 'press' with at least one value for the role 'ISSN' is automatically classified as a kind of magazine, although it was not said explicitly in the creation of the new concept. The use of the subsumption mechanism supported by DL systems when dealing with ontologies provides us with the following advantages:

- The system maintains a hierarchical organization of terms which is very useful in dealing with large collections of definitions.
- *Concept incoherence and disjointness.* Given a query description it can be determined if it is incoherent with any other descriptions in the ontology. For instance, the creation of a concept which description is '*(AND magazine (AT-MOST 0 ISSN))*' will lead to an error: it is not possible to be a magazine and not have any value for role 'ISSN' at the same time. It is also possible to define two concepts as *disjoint*. This means that no instance can belong to both concepts.
- Queries, which are also descriptions, are themselves *classified* into the subsumption hierarchy. This can help first, to remove redundant constraints in the query, second, to detect incoherent queries, and third, to help identify relevant data repositories and data relevant to the query.

In every ontology there exists two distinguished concepts, *Anything* (in CLASSIC it is called 'CLASSIC-THING') and, although it is not represented explicitly, *Nothing*. The first one subsumes all the concepts in the ontology and the second one is subsumed by the rest of the concepts of the ontology (they are the top and the bottom of the ontology). Analogously, roles *Anyrole* and *Norole* are the top and bottom, respectively, of any role hierarchy.

3.2.3. Queries in DL systems. Another important feature of DL systems is that queries are also considered as concept descriptions. So queries are classified by the DL system. Thus, redundant constraints or contradictions are automatically detected by the DL system. We will use the following syntax to express queries:

[role1 role2 ...] for concept-description

where *role1*, *role2*, ..., are the role names of the instances that satisfy the constraints in the concept description whose values we want to get. *Concept-description* is a combination of DL operators that can include names of concepts, cardinality constraints on the range of roles (AT-LEAST, AT-MOST), constraints on the range of roles (ALL), values for roles (FILLS), etc. For example, the query '*[frequency] for (AND magazine (FILLS name "Life"))*' formulated (with the help of a GUI) over the ontology WN (see Figure 1) asks for the frequency (daily, monthly, ..) of the magazine "Life". Although DL systems do not provide us with a database type query language, we decided on using the more restricted DL expressions as a query language because of three reasons:

- Statistics of Web search engines and databases say that queries executed by most of the users are very simple; they usually are a conjunction of conditions that provide values for roles that objects in the answer must satisfy, which can be expressed in DL easily (operators 'AND' and 'FILLS'). The proof is that no database accessible through the Web provides SQL access but very limited forms that users have to fill in.

- Syntactically, there never exists explicit join conditions as in SQL or languages based on predicates. The different concepts are implicitly linked by DL operators. So only the "intuitive" conditions appear in the query.
- Some DL systems, although not all, support disjunctions. Lower performance is the price that we could pay if we do not use very big ontologies.

Thus, considering both the advantages and the limitations of using DL systems, we believe that DL expressions are useful as a query language for a GIS.

3.3. Ontologies in our prototype

In this section we discuss the real-world ontologies accessible in our prototype. It is important to notice the heterogeneity among the ontologies because they have been developed independently by different organizations that belong to the areas of linguistics and knowledge representation. We *re-use* these ontologies after representing them in CLASSIC. The ontologies were created using different knowledge representation languages, but we have described them in DL, trying to maintain the same semantics provided originally by their creators. We use those ontologies to describe real-world data repositories. The complete DL definitions, the semantic relationships defined between them, and their mappings to the underlying data repositories have not been included due to space limitations but can be found in [24]. We have chosen a significant set of ontologies built from three different points of view: linguistics research, knowledge representation research and the individual point of view of some research groups.

- *Linguistic point of view ontologies* Two ontologies belong to this category: *WN* which was based on WordNet 1.5 [29] and *μCosmos* which is a subset of MikroKosmos [10]. These ontologies were created from the point of view of a global ontology: they are knowledge bases used by their creators (and other users) to perform automatic recognition and translation of texts. They are very big in size because they try to represent any kind of knowledge. We would like to stress that, even though they were designed for the same purpose, they look quite different. Both ontologies, WN and μCosmos, only contain those terms from the original ontologies that are related to bibliographical references, in order to minimize the work of redefining WordNet and MikroKosmos.
- *Knowledge representation view point ontologies* Two of our other ontologies are subsets of the 'Bibliographic-Data' ontology [18] developed as a part of the ARPA Knowledge Sharing Effort. We have divided the original ontology into two pieces, both describing bibliographical references: *Stanford-I*, which corresponds to the ontology Bibliographic-Data except the subtree under 'reference', and *Stanford-II*, which corresponds to the sub-tree of the hierarchy under the concept 'reference' of the Bibliographic-Data ontology.
- *Individual point of view ontologies* We have developed two ontologies used to describe the information (and differing points of view) of two different research

groups at different universities: the *LSDIS* ontology representing the view of bibliographic data from the perspective of the personnel of the Large Scale Distributed Information Systems (LSDIS) Lab, at the University of Georgia; and the *BDI* ontology representing the point of view of people in the Interoperable Database Group (BDI, in Spanish) at the University of the Basque Country about research publications. Notice that its terms are expressed in Spanish.

Some of the corresponding concept hierarchies appear in the appendix, and all of them in [24].

4. OBSERVER: Architecture and Query Processing Approach

In this section we describe the architecture of the OBSERVER system, present an outline of the query processing approach along with a detailed discussion of the first step. But we first introduce a motivating example which illustrates the different problems that arise when querying Global Information Systems and for which we provide a solution.

4.1. A Motivating Example

The query which we will use as a running example throughout this paper is:

"Get title, number of pages, year of publication and a file containing the publication itself, if available, for books related to Mars where the only author is Carl Sagan."

The following problems need to be solved in order to answer a query such as above in a GIS:

Resource Discovery: We need to search for the repositories relevant to the query, (e.g., which repositories are likely to have information about publications related to Mars ?).

Structure/Format Heterogeneity: Different data repositories containing relevant data for a query may have different data organizations (e.g. publications are stored in databases, file systems, etc.), formats and media, (e.g., the sample query requests the publication that is stored in a non-textual format such as a Microsoft Word^{TM} document), and may support different query languages.

Modeling of Information Content: The same information may be modeled at differing levels of abstraction (e.g., "book" vs. "publication"). On the other hand only part of the information required may be modeled at an information source/ontology. For example, information about books and authors may be modeled at different information sources/ontologies.

Querying of the Information Content: If keywords do not appear in a document, such a document will not be retrieved even though it may be relevant

(e.g., the words "Carl Sagan" may not appear in the content of his publications). We need a semantic approach to access information rather than a syntactic approach. We should also define a query language expressive enough to precisely describe our information needs in an intensional manner.

The Vocabulary Problem: Current Internet tools and query processing systems are unable to support heterogeneous vocabularies used to describe the same information. In the case of keyword-based systems, if a synonym of the keyword included in the document is used as a part of the query, the document may not be retrieved. Different but related terms may be used to describe similar information at the intensional level (e.g. the term for "subject" may be modeled as "keywords" at a different ontology) as well at the extensional level (two different representations represent the same value: 'February 10, 1998' and '2/10/98').

Imprecise answers: We need to deal with imprecise answers when most of available data repositories do not model our information needs exactly. The loss of information incurred should be measured, controlled and reduced as the system enriches the answer by visiting more repositories. In the example, the system could deal with books with Carl Sagan as the only author even though there may be no way to know if those books are about Mars.

4.2. Architecture of the OBSERVER system

The design of OBSERVER reflects the need to create a system that is highly independent of the number of data repositories/ontologies and is capable of dealing with many types of heterogeneity at the structural, functional or semantic level. Each node in the GIS includes query processing capabilities and, additionally, data repositories accessible to the rest of the nodes through ontologies that describe them. In order to solve the vocabulary problem, a shared repository containing the interontology relationships is used. This repository, called the Interontology Relationship Manager (IRM) can be seen as the *catalog of semantics* of the system.

The basic elements of the architecture are illustrated in Figure 2 and introduced in the rest of this section. Ontologies were already introduced in Section 3, while the description of the mapping information will be discussed in Section 5.2. We now describe the main modules of the architecture.

4.3. The Query Processor

Figure 3 shows OBSERVER's three step query processing approach. The first step of *Query Construction* is discussed in detail in the rest of this section; the remaining two steps of *Access to Underlying Data* and *Controlled Query Expansion to New Ontologies* are discussed in the subsequent sections.

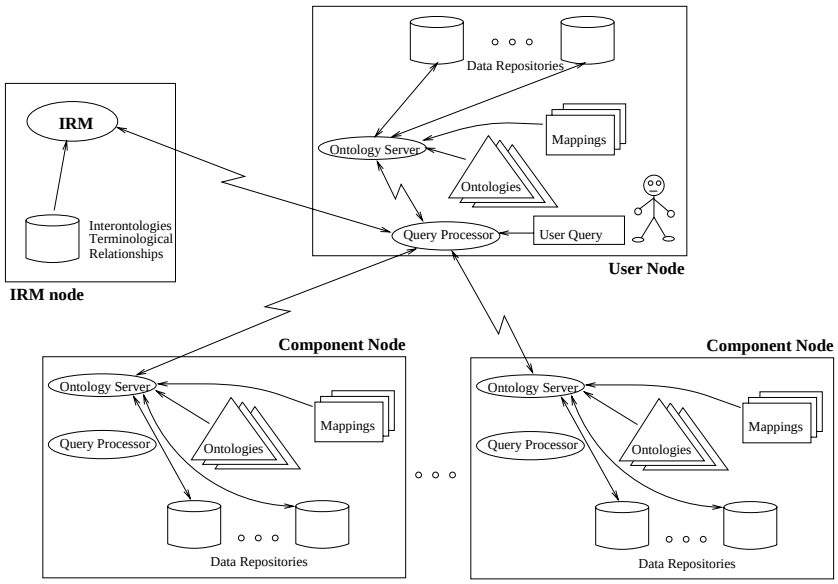


Figure 2. OBSERVER: An architecture to support query processing

4.3.1. Query Construction. This step consists of two tasks discussed below with the help of the example introduced earlier. These tasks involve direct user involvement. A GUI facilitates this user involvement.

1. Select User Ontology. For the example query, the locally developed BDI ontology is selected because it contains all the terms needed to express the semantics of the query: terms that store information about titles ('titulo'), number of pages ('paginas'), year of publication ('fecha'), the publication itself ('fichero'), for books ('libro') as well as their subjects ('temas') and authors ('autores')[3]. Other reasons could be that terms in BDI are expressed in Spanish which makes query formulation easier for that particular user who is also familiar with the structure of BDI ontology.

   Note that the important reason to choose those terms is not that they are the Spanish translations of the words in the query but that they express the exact semantics of the words in the query, independently of the natural language used. If, for example, the term 'fecha' (that literally means 'date') represents the day, month and year of the publication, it would not be what the user has asked for (only the year of publication). By navigating the ontologies, users can browse terms and its definitions both in DL and in a natural language (defined by the ontology creator) in order to choose the terms that fit the semantics of the query exactly. If incorrect terms are chosen, then the answer could contain unwanted data.

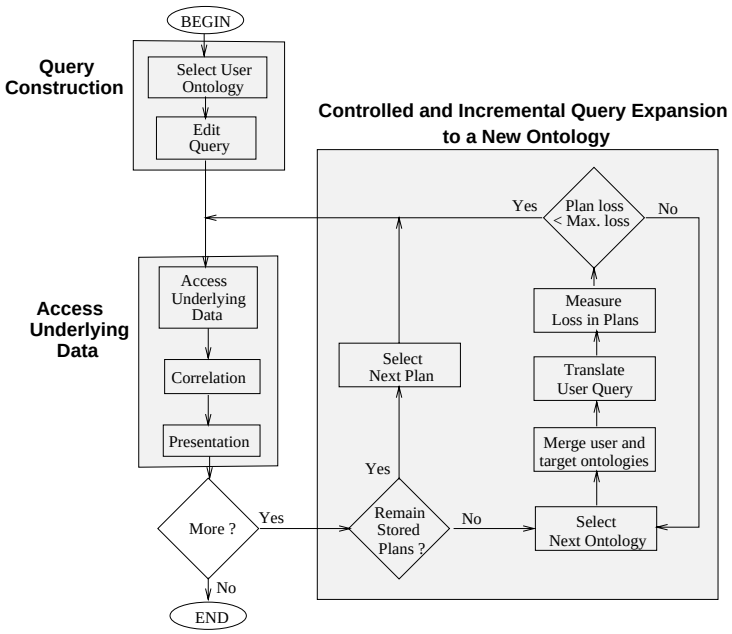


Figure 3. OBSERVER: query processing approach

2. Edit Query. After selecting the user ontology, the user chooses terms from that user ontology to build the constraints and projections that comprise the query. The DL query corresponding to our example query presented in natural language earlier is:

*[titulo paginas fecha fichero] for (***AND** *libro (***FILLS** *temas "Mars") (***EXACTLY** *1 autores) (***FILLS** *autores "Carl Sagan"))*[4]

A *defined term* (see Section 3.2) $\mathcal{Q}$ corresponding to the constraints in the query is created by the Query Processor in the user ontology. That is, the DL concept $\mathcal{Q}$ represents the objects satisfying the user query.

$\mathcal{Q}$:=[5] *(***AND** *libro (***FILLS** *temas "Mars") (***EXACTLY** *1 autores) (***FILLS** *autores "Carl Sagan"))*
'*[titulo paginas fecha fichero] for* $\mathcal{Q}$ ' is the new query

$\mathcal{Q}$ is removed by the Query Processor after the processing corresponding to that query completes.

4.3.2. Accessing the Underlying Data. The Query Processor invokes the *Ontology Server* to retrieve data corresponding to the query by accessing data repositories

associated with the user ontology. For each ontology there exists some *Mapping Information* that links that ontology with the underlying data elements. This information is used by the Ontology Server to translate the user query (expressed in terms of the user ontology) into different queries to the underlying data repositories (each subquery expressed in the local query language of the repository where it is going to be executed). Thus, the Ontology Server retrieves the information corresponding to the query. That information is retrieved from the different repositories in an common format so that the partial answers can be easily combined, i.e., *correlated*. The answer is returned to the user by the Query Processor. The query processing ends if the user is satisfied with the answer.

4.3.3. Controlled Query Expansion to New Ontologies. If the user wants to obtain more relevant data, then other ontologies that have related terms are visited. In this case, the original query is translated from terms of the user ontology into terms of another component ontology, also referred to as a target ontology. The translation cannot always be exact because all the abstractions represented in the user ontology may not appear in the component ontologies. Hence the user can define a limit for the *Loss of Information* allowed. For example, if the user defines a limit of 20% the system guarantees that the amount of unwanted (loss in *precision*) or missed data (loss in *recall*) in the future answers presented to the user is kept always below 20% of the information showed. Of course the user can define a limit of 0% to ensure that the answer always matches exactly the semantics of her/his query.

For the task of translating the query from terms of the user ontology into terms of the target ontology, the user and the target ontologies are integrated automatically by the Query Processor. The Query Processor requests the semantic interontology relationships that were defined between the user and target ontologies from the IRM. In that process of integration the user query is rewritten into terms of one or more target ontologies. If, because of the rewriting, the user query is completely expressed in terms of the target ontology, the translation is called a *full translation*. The Query Processor invokes the Ontology Server corresponding to the target ontology which will retrieve the underlying data that corresponds to the translated query.

A *partial translation* is obtained if some terms from the user ontology cannot be rewritten in terms of the target ontology. The query processor tries to substitute the non-translated terms in order to obtain a complete translation. This task can generate several alternative translations; for each one, the Query Processor estimates the associated loss of information. It rejects those that are beyond the threshold and chooses the one with the least loss of information. The new answer is correlated with the previous one and the result (and the associated loss) is presented to the user who can choose again between finishing the process or enriching the answer by visiting new ontologies. In the latter case, a new target ontology is chosen and the same iteration is repeated again until the user is satisfied.

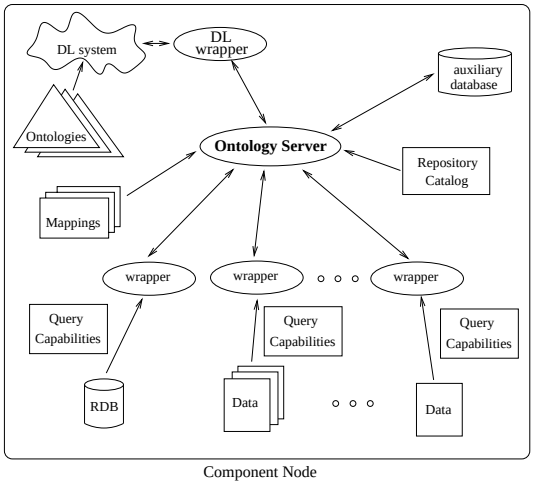


Figure 4. The Ontology Server and its associated components

4.4. The Ontology Server

The Ontology Server is the module that provides information about ontologies residing on its node as well as about their underlying data repositories. There is only one Ontology Server per node containing ontologies. The Ontology Server maintains a catalog of its ontologies, the *Repository Catalog*, which contains information about the data repositories underlying each ontology. We explain the rest of the associated components (see Figure 4) in this section.

The goal of the Ontology Server is twofold: 1) to encapsulate any direct interaction with ontologies and the DL system, and 2) to encapsulate any access to data repositories. Both kinds of encapsulation make it possible to develop a Query Processor that never deals with the details/heterogeneity existing in the underlying repositories. This allows for an approach for handling the vocabulary sharing at two levels: differences across ontologies and differences across underlying repositories. The services provided by the Ontology Server are classified according to the information requested:

Services that provide intensional information of an ontology. The Ontology Server can be invoked by other system modules in order to obtain information about the structure of any ontology on its node. The following are the main services of this kind, illustrated with examples:

- Enumeration of terms. For example, Get-concepts(WN) $\rightarrow$ { print-media, dictionary, book, ... }.
- *Obtaining term definitions.* For example,
  Get-definition(dictionary, WN) $\rightarrow$ (**AND** print-media (**FILLS** content "d")).

- *Immediate subsumers/subsumees of a term.* For example,
  Get-immediate-subsumers(periodical,WN) → { publication }
  Get-immediate-subsumees(periodical,WN) → { pictorial, series, journal }.

- *Verification of defined terms.* For instance, Is-defined(dictionary, WN) → true.

- *Size of extension of a term.* This information is calculated based on the *Mapping Information* of the term (see Section 5.2) and stored information about the size of the underlying data structures (this stored information is updated periodically and automatically).
  For example, Size-of(book,WN) → 1105.

- *Graphical representation of the ontology.* For example, get-VRML(WN) → VRML description, it returns a three-dimensional representation of the concept hierarchy of ontology WN in Virtual Reality Modeling Language (VRML) [38].

Most of these services are invoked by the Query Processor during the process of translating the user query into the language of another component ontology. This is described in detail in Section 6.

Answering queries formulated over an ontology. Given a query in DL and an ontology name the Ontology Server returns the corresponding data stored in the repositories underlying such an ontology. The answer is returned in a structure similar to a relation of the Relational Model independently of the underlying data structures. For example:
Get-extension('[pages] for dictionary',WN) → <tuple1, tuple2, ... >
The Ontology Server utilizes mappings between terms in the ontology and data structures in the underlying repositories as well as the appropriate *wrappers* in order to achieve this task. This service is the core of the Ontology Server and will be explained in detail in Section 5. In fact, this complex process encapsulates the heterogeneity and distribution of the data repositories underlying the ontologies. The Ontology Server consults the *query capabilities* of each data repository to construct a plan and then invokes different *wrappers* specialized in specific data organizations to access the data.

4.5. The Interontology Relationships Manager (IRM)

Any kind of semantic property that can be defined across component ontologies of the system is stored and managed by the IRM in an independent repository. This enables a solution to the *vocabulary problem.* The IRM is the critical component which supports ontology-based interoperation. The interontology relationships make explicit the semantic overlapping between the ontologies or the domains they represent. It also enhances the scalability of the query processing strategy by avoiding the need for: (a) designing a common global ontology containing all the relevant terms in the GIS; and (b) investing time and energy for the development of an ontology specific to a user's needs when similar ontologies are available. The main assumption behind the IRM is that the number of relationships between terms across

ontologies is an order of magnitude smaller than the number of all the terms relevant to the system. The information needed to build a huge global ontology describing the whole GIS is distributed in our system between the component ontologies and the interontology relationships stored in the IRM repository.

Hammer and McLeod [20] have suggested a set of relationship descriptors to capture relationships between terms across different (locally developed) ontologies. In our context, as relationships should help to translate queries from the "language" of an ontology into the language of another ontology, OBSERVER deals with the following kinds of interontology relationships, which are a subset of the defined in the above work:

- *Synonym relationships*, when two terms in different ontologies have the same semantics. Note that it does not mean that they have the same extension.

- *Hyponym relationships*, when a term is less general than other in another ontology, i.e., a term in one ontology subsumes the abstraction represented by another term in a different ontology. Note that this does not mean that the extension of the term is a superset of the extension of the abstraction.

- *Hypernym relationships*, when a term is more general than other in another ontology, i.e., a term in one ontology is subsumed by another term in a different ontology. Note that this does not mean that the extension of the general term is the superset of the extension of the specific term.

- *Overlap relationships*, when there exists an intersection between the abstractions represented by two given terms. Two numbers can be associated with these relationships to represent the approximate percentage of the two terms participating in the relationship that belong to the intersection. If the first term subsumes the second, the second percentage will be 100% (all the objects that belong to the second term belong to the first too). These kind of relationships are very common when dealing with abstractions created from different points of view (synonymy is very infrequent, hypernymy and hyponymy only occur between very general or very specialized abstractions). We provide here an illustrative example of overlap between terms of the BDI ontology:

  < publicadas, 90% > overlap < articulo-libro, 20% >; it asserts that the 90% of the publications of the group BDI already accepted ('publicadas') represent around the 20% of the publications classified in the ontology BDI as articles in a book ('articulo-libro'). The two terms participating in the relationship are not related by any hypernym or hyponym relationship.

  We observe that these relationships, in spite of representing semantic information, are based on extensions. The percentages help to estimate the loss of information when substituting a term by another one in a query.

- *Disjoint relationships*, when there exists no intersection between the abstractions represented by two terms.

- *Covering relationships*, when the abstraction represented by a term in one ontology is the same that the abstraction represented by the union of other given abstractions which are subsumed individually by the term, i.e., there does not exist an object represented by the parent term which is not represented by the union of the given children terms.

The assertion of these properties does not mean that they are also true with respect to the extensions of the terms involved. We are mainly interested in semantics. These relationships will be consulted by the Query Processor to solve the vocabulary problem at the intensional level. In order to solve the vocabulary problem at the extensional level, *transformer functions* between roles of different ontologies can also be defined in the IRM. These functions are necessary to transform constants from the user ontology into some target ontology (to access the data underlying the target ontology) and vice versa from the target ontology into the user ontology (to correlate and present the answer to the user). If the IRM repository becomes so huge that the efficiency of the GIS is affected, its independence with respect to the system enables its partitioning or mirroring without affecting the rest of the system. Distributed database management techniques can be used to handle this issue.

Services provided by the IRM. The IRM makes available to Query Processors the semantic relationships across all the component ontologies in the GIS. The following are some of the IRM services that can be used by the Query Processors:

- **Get-clusters()** returns the name of all the clusters (or knowledge areas) defined.
- **Get-ontologies(cluster)** returns the name of all the component ontologies of the GIS related to the indicated cluster.
- **Get-node(ont)** returns the node where that ontology and its Ontology Server are located.
- **Related-terms(ont1, ont2)** returns five lists (synonym, hyponym, overlap, disjoint and cover terms) that include the pair of terms of ontologies *ont1* and *ont2* related by the previous relationships.
- **Transform-value(val,role1,ont1,role2,ont2)** returns the equivalent value of *val* stored in *role1* (ontology *ont1*) but for *role2* in the ontology *ont2*. For example:

  transform-value("d", content, WN, type-of-work, Stanford-II) → "dictionary"

  In the previous example, in ontology WN, the value "d" in role "content" indicates that the corresponding publication is a dictionary; the same idea is represented in ontology Stanford-II by including the value "dictionary" in role "type-of-work".

- **Transform-table(table,list-of-roles1,ont1,list-of-roles2,ont2)**, given a relational *table* containing a list of values for the roles in *list-of-roles1* of ontology *ont1*, returns another table. In the resulting table, if there exists a transformer function between $role1_i \in$ *list-of-roles1* and $role2_i \in$ *list-of-roles2*, all the values in $column_i$ are substituted by the result of Transform-value(*value*,$role1_i$,*ont1*, $role2_i$,*ont2*). In other words, this service translates the values included in *table* from the representation of *ont1* into the representation of *ont2*. For example:

```
table = { < 'Webster Dictionary', 'd', '04/14/96' >,
          < 'Painting in XIX Century', 'e', '11/01/75' > }
```

Transform-table(table, <name, content, date>, WN, <title, type-of-work, date>, Stanford-II) performs the following transformation:

```
table = { < 'Webster Dictionary', 'dictionary', 'Apr. 14, 1996' >,
          < 'Painting in XIX Century', 'encyclopedia', 'Nov. 1, 1975' > }
```

5. Accessing underlying Data Repositories

In this section we first present the definitions used to denote different components of data repositories and a formal description of the mapping information. Then we give a detailed presentation of the methods for accessing underlying data repositories. Finally, we discuss several interesting issues related to the correlation of data from multiple repositories.

5.1. Logical schemas, data repositories and data sources

From the point of view of mapping expressions, data repositories are seen as a set of entity types and attributes. Thus each repository has an associated logical schema. Mappings act as an intermediate language between DL expressions and the query languages of the local repositories. Each data repository has a specific data organization and may or may not have a data manager. It can be composed of several **data sources** which actually store the data. The different data sources of a repository can be distributed. The simplest data source is a system file. Almost everything can be a data repository: a set of files of different formats, an HTML page, a database, and their combinations. Different types of repositories may exist under an ontology of our prototype. This is illustrated in Figure 5. Table 1 further summarizes the logical schemas, the real world data sources, the organization and the size of the repositories associated with the various ontologies used in our prototype.

A *wrapper* is a module which understands a specific data organization. It knows how to retrieve data from the underlying repository and hide the specific data organization to the rest of the GIS. Extensive work has been performed on generating/designing and using wrappers (e.g., [19]), so we do not discuss this topic in detail.

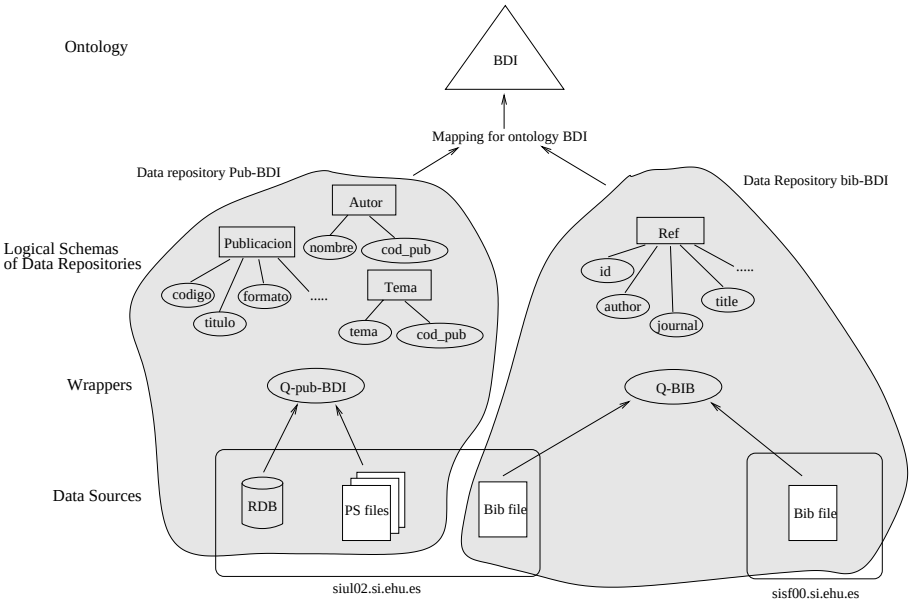


Figure 5. Data repositories, wrappers and data sources for ontology BDI

Table 1. Overview of the underlying repositories in the prototype

Ontology	Data repository	Data sources	Data Org.	Records
WN	uga-library	UGA Main Library (subset)	Files containing MARC records	1.5K
μCosmos	amazon	Amazon.com	Unknown	2.5M
Stanford-I	monterrey	Library at Monterrey (subset)	ORDB storing MARC records	25K
Stanford-II	loc	Library of Congress	Unknown	3.8M
LSDIS	lsdis	LSDIS Lab Publications	A Text file	70
		publications in HTML	an HTML page	70
		LSDIS postscripts	Postscript files	43
BDI	pub-BDI	BDI group publications	MSQL RDB	44
		BDI postscripts	Postscript files	27
	bib-BDI	BDI references on siul02	a BIB file	395
		BIB references on sisf00	a BIB file	354

The data organizations of Amazon.com and the Library of Congress (LOC) are unknown to us (and we do not care about them). In such cases, a wrapper for accessing such as site emulates the behavior of a user accessing it from the Web. In such cases, we take an *operational view* of these data sources.

5.2. Mappings: The Key to Repository Heterogeneity Encapsulation

Mapping Information relates terms in the ontology (concepts, roles) with data elements in data repositories (e.g., tables and columns in the case of a relational database, entities and attributes of the underlying logical schemas in general). For each term in an ontology the mapping information is represented as a tuple which involves the use of Extended Relational Algebra (ERA) expressions [12]. These mappings play a key role in encapsulating the heterogeneity due to different formats and organization of the data in the various repositories. The advantages of these mappings are as follows.

- The mappings subscribe to the idea of viewing a data repository as a set of entities and attributes (or relations and attributes in ERA), independently of the specific organization of the data in the repository. This gives an homogeneous view of the description of the data repositories without capturing any characteristic specific to the individual data repositories.
- They are expressive enough to capture the complex associations of concepts and roles with entities and attributes.
- They act as an intermediate language between the DL expressions and the concrete query languages of the local repositories.

The mapping information for a term is a list of *basic mappings*. The description of the basic mappings for concepts and roles is as follows. For a concept, a basic mapping is defined as a 3-tuple:

$$\textbf{<Rel, } (a_1 \ldots a_n)\textbf{, T>}$$

where *Rel* is a (basic or derived) relation in ERA; $a_1 \ldots a_n$ are attributes of *Rel* that identify its instances; and T is a list containing the types of those attributes. An example using terms from BDI ontology is as follows:

```
CONCEPT libro:
    [UNION [SELECTION pub-BDI.publicacion
                [= pub-BDI.publicacion.formato "Book"]]
           [SELECTION bib-BDI.ref
                [= bib-BDI.ref.type "book"]]]
    pub-BDI.publicacion.codigo
    string
```

For a role, a basic mapping is defined as a 6-tuple:

$$<\mathbf{Rel},\ (a_1 \ldots a_n),\ \mathbf{T},\ (an_1 \ldots an_m),\ \mathbf{T}_{rl},\ \mathbf{f}_{rl}>$$

where *Rel* is a (basic or derived) relation in ERA; $a_1 \ldots a_n$ are attributes of *Rel* that identify its instances; T is a list containing the types of those attributes; $an_1 \ldots an_m$ are attributes of *Rel* that define the role values (or contain the role values) corresponding to the concept instances; T_{rl} is the range of the role; and $f_{rl} : D_1$ X ... X $D_m \rightarrow T_{rl}$ where D_i is the domain of attribute an_i for $1 \leq m$. f_{rl} allows the transformation of the stored attribute values into the final values of the role; f_{rl}^{-1} allows the transformation of role values into stored attribute values. E.g., if underlying data is stored in Euros but we have defined a role whose values are in Dollars, in the mappings of such a role we have to specify a function f_{rl} that transforms Euros into dollars; f_{rl}^{-1} allows the transformation from Dollars into Euros. An example of a basic mapping for a role is as follows:

```
ROLE titulo:
  [UNION [PROJECTION pub-BDI.publicacion [pub-BDI.publication.codigo
                                          pub-BDI.publication.titulo]]
         [PROJECTION bib-BDI.ref [bib-BDI.ref.id bib-BDI.ref.title]]]
   pub-BDI.publicacion.codigo
   string
   pub-BDI.publicacion.titulo
   string
   none
```

The formal definition of the mapping information used and its application to relational databases can be found in [15]. The mappings defined for ontologies in our prototype can be found in [24]. Since mappings are defined for terms, in order to obtain the mapping information corresponding to a DL query (which is a DL expression) the Ontology Server also uses a mechanism to combine mappings of terms to obtain the mapping of the whole query, as explained in the next section.

5.3. Main Steps in Accessing Underlying Data Repositories

In this section we explain in detail the method followed by the Ontology Server to access data corresponding to a DL query formulated over an ontology. The main steps followed are presented in Figure 6. In the following we explain each step in detail with the help of the example query presented in previous sections.

Step 1: Translation from DL into Mapping. The DL query formulated over an ontology has to be rewritten in terms of the underlying data repositories in order to retrieve the corresponding data. The first step to achieve that goal is to translate the DL query into an equivalent mapping expression. This translation is always possible as mappings for each term in the ontology will have been defined when the ontology was made available to the GIS.

The Ontology Server builds the mapping expression corresponding to the DL query in a recursive manner, constraint by constraint. This involves combining

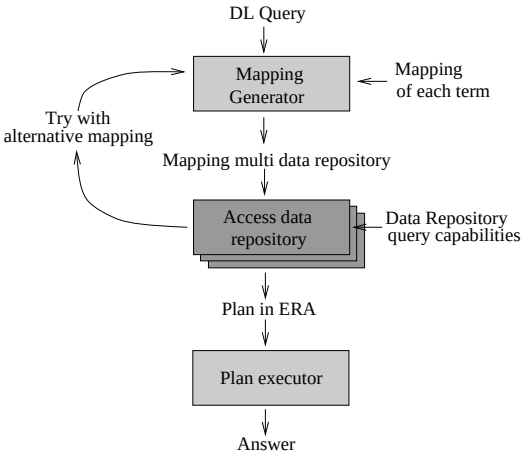


Figure 6. Steps needed to answer a DL query over an ontology

the mappings of the terms appearing in a constraint in a such a manner that the resulting mapping is a representation in ERA of the semantics of the DL query. A detailed description of each case and some possible optimizations have been presented in [15] so we will only show here the result of this step for our example.

```
PROJECTIONS: [pub-BDI.publicacion.titulo bib-BDI.ref.pages
              pub-BDI.publicacion.fecha pub-BDI.publication.fichero]

RELATION:
        [UNION [PROJECTION [JOIN [AGGR-FUNCTION [JOIN [JOIN [SELECTION pub-BDI.publicacion
                                                              [= pub-BDI.publicacion.formato "book"]]
                                                    [SELECTION pub-BDI.temas
                                                        [= pub-BDI.temas.tema "Mars"]]
                                                ]
                                                pub-BDI.autor
                                            ]
                                            [pub-BDI.publicacion.titulo pub-BDI.publicacion.fecha
                                             pub-BDI.publicacion.fichero]
                                            [[COUNT = 1] pub-BDI.autor.nombre]
                                ]
                                [JOIN pub-BDI.publicacion
                                      [SELECTION pub-BDI.autor [= pub-BDI.autor.nombre "Carl Sagan"]]
                                ]
                           ]
                           [pub-BDI.publicacion.titulo NULL
                            pub-BDI.publicacion.fecha pub-BDI.publicacion.fichero]
            ]
            [PROJECTION [JOIN [AGGR-FUNCTION [SELECTION bib-BDI.ref [AND [= bib-BDI.ref.type "book"]
                                                                         [= bib-BDI.ref.key "Mars"]
                                                                     ]
                                            ]
                                            [bib-BDI.ref.title bib-BDI.ref.pages bib-BDI.ref.year]
                                            [[COUNT = 1] bib-BDI.ref.author]
                                ]
                                [SELECTION bib-BDI.ref [= bib-BDI.ref.author "Carl Sagan"]]
                           ]
                           [bib-BDI.ref.title bib-BDI.ref.pages bib-BDI.ref.year NULL]
            ]
    ]

ATRC: pub-BDI.publication.codigo

DOMC: string
```

Notice that the mapping expression involves two repositories, 'pub-BDI' and 'bib-BDI'. The mapping of the terms is based on the entities and attributes of the *logical*

schemas describing the underlying data repositories. These logical schemas are the views defined for each data repository. Of course, a term in an ontology can have a mapping involving entities or attributes from different schemas, i.e., a term in an ontology can be related to several data repositories. We call this a *multi-repository mapping expression*. Alternatively, a *mono-repository mapping expression* involves only one repository/local schema.

Step 2: Accessing the Underlying Data Repositories. This step is one of the most important in the entire process of obtaining the answer for a DL query formulated over an ontology. Two main tasks are involved in accessing the data corresponding to the mapping expression obtained in the previous step: creation of mono-repository subexpressions; followed by their translation into the local query languages of the repositories and data access. These tasks are discussed next.

Task 1: Obtaining the main plan by dividing the multi-repository mapping expression into several mono-repository subexpressions combined by relational operators. The multi-repository mapping expression is analyzed and a tree representing the main plan is generated. Mono-repository mapping subexpressions are the leaves of the tree and the intermediate nodes are relational operators that will operate on data coming from different data repositories. Figure 7 shows the main plan obtained for our example. These operations (in the example, the union) will be performed by the Ontology Server with the help of an auxiliary database as explained later in the Step 3.

The subqueries at the leaves of the tree must be replaced by the name of a relation in the auxiliary database that contains the corresponding data. This is achieved in the second step by repeating it for each leaf in the main plan.

Task 2: Translation from mono-repository mappings into LQL and data access. Each mono-repository subexpression is translated into the query language understood by the component data repository referred to as the *local query language* (LQL). Subsequently, the data is accessed using the corresponding *wrappers* and stored in an auxiliary database. Finally, the plan is re-written according to the operations that could not be executed by data repository managers.

The retrieval of data corresponding to a mapping expression from the corresponding data repository may not be easy or straight forward. In some cases, the manager of a component repository may have limited query capabilities. Some operations appearing in the mapping expression may have to be postponed in such cases and executed later by the Ontology Server with the help of an auxiliary relational database.

Each data repository has an associated grammar that represents its query capabilities. ERA is used as a common way to express the query capabilities of the repositories independently of the LQL. The following technique is used to avoid the generation of LQL expressions that cannot be executed in the underlying data repository[6]. The mapping expression is analyzed using a context free grammar that recognizes any valid mapping expression. For each mapping expression recognized in a rule, if there exists a translation into the LQL the corresponding sequence of

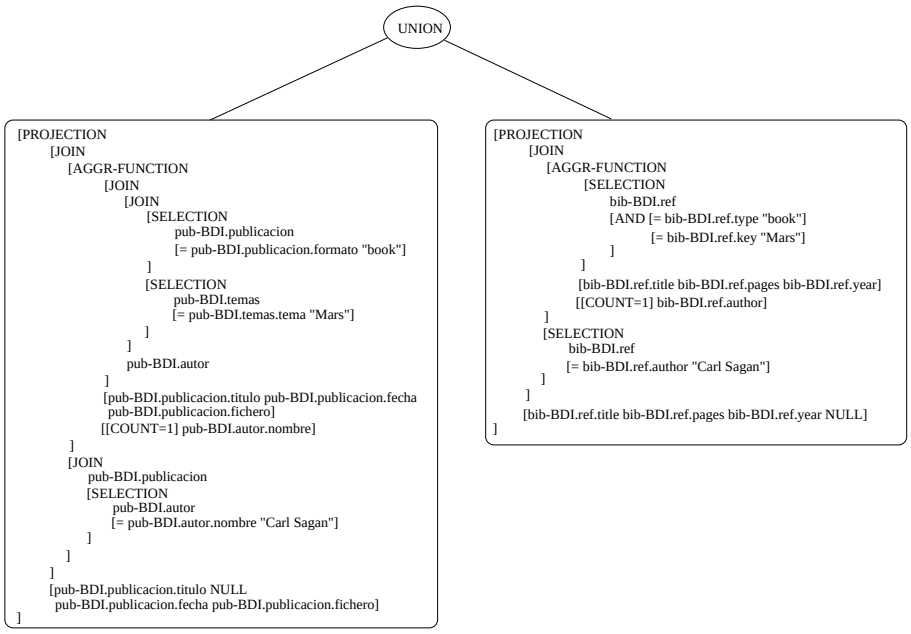


Figure 7. Main plan for the example

sentences in LQL is returned; in the other case, one subplan is generated for that mapping expression. This grammar accepts any mapping expression but, depending on the concrete query capabilities, the result will be a trivial plan (only one LQL expression) or a more complex plan combining several LQL expressions.

Continuing with our example, the translation of the first subplan (left branch in Figure 7) is shown in Figure 8. In the case of 'pub-BDI' repository whose main[7] data source is a relational database any mapping expression can be translated into the LQL (i.e., SQL). The technique to translate mapping expressions (based on ERA) into SQL is well known and is not discussed here. The result is a trivial case where the only node is a dashed box containing an expression in LQL.

Figure 9 shows the translation from mappings into the LQL of the repository 'bib-BDI'. As there is no manager for this repository, the LQL expression used is the name of the entity involved, the projections ('*' denotes the projection of all the attributes) and a list of conditions (conjunction between them is implicit). The grammar describing query capabilities is also included. Notice that not all ERA operators in the mapping expression have a translation into this LQL.

The operations represented by the inner nodes will be performed by the Ontology Server later. The result of the substitution of each mapping subexpression by the two resulting plans is shown in Figure 10.

When the whole plan is returned by the analyzer, the Ontology Server parses the tree and for each leaf (which is an expression in LQL represented as dashed squares

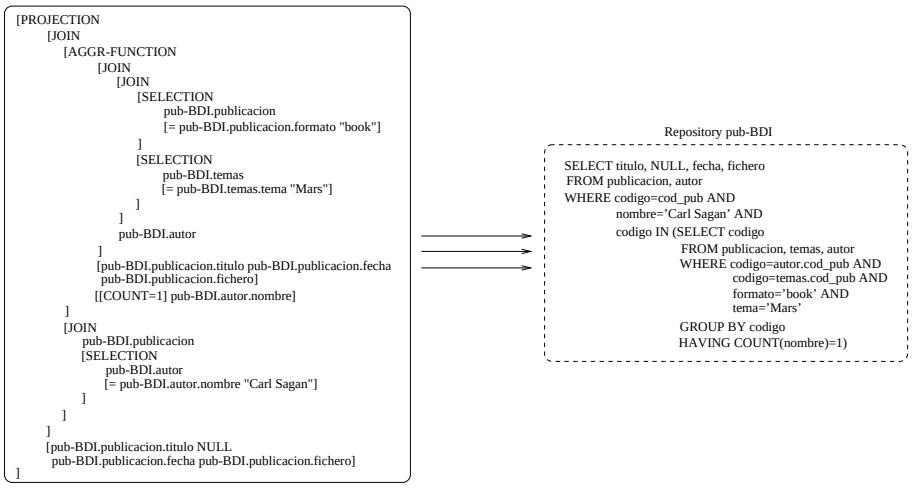


Figure 8. Translation of left branch of main plan into LQL of 'pub-BDI'

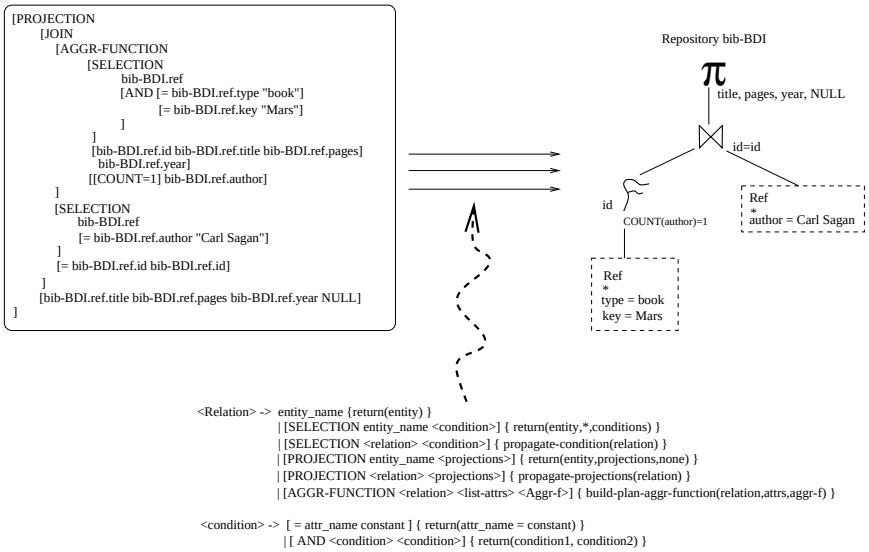


Figure 9. Translation of right branch of main plan into LQL of 'bib-BDI'

in figures), the corresponding data is accessed using a *wrapper*, then it stores the data in an auxiliary database (also with the help of a wrapper) and finally each LQL expression in the tree is substituted by the name of the relation of the auxiliary database that contains the data corresponding to such an LQL expression. This is done for all the leaves and at the end a tree containing only ERA operations and relation names is returned. This means that the mono-repository mapping

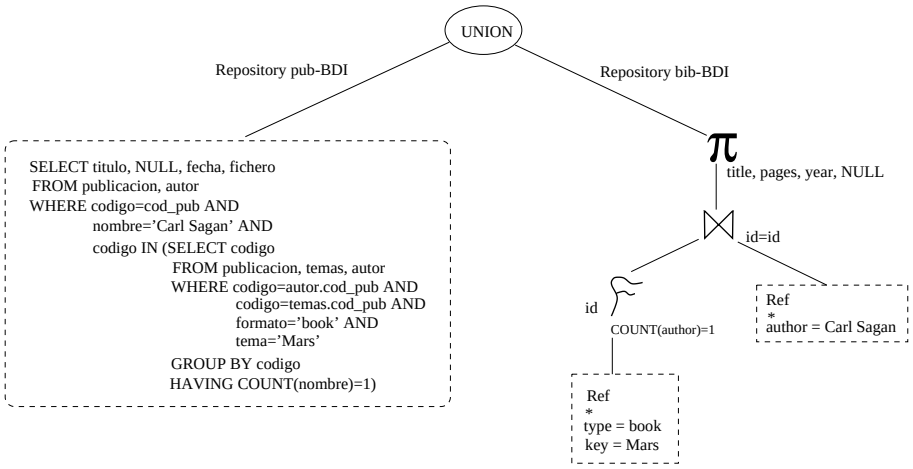


Figure 10. Updating the main plan after translating into LQL

expression has been substituted by a plan in ERA that represents the operations that cannot be performed by the underlying data repository. As a side effect, the result of accessing the repository is stored in an auxiliary database. The process described here is applied for the rest of the leaves in the main plan.

The above process is illustrated using our example in Figure 11 as: (1) the main plan after accessing the 'pub-BDI' repository; 'AUX-1' is a table of the auxiliary database that stores the data retrieved; (2) the main plan after accessing 'bib-BDI'; 'AUX2' is the table that stores the data retrieved; and (3) the main plan after accessing again 'bib-BDI' and storing the retrieved data in table 'AUX-3'. Observe that the main plan in (3) only references the auxiliary database.

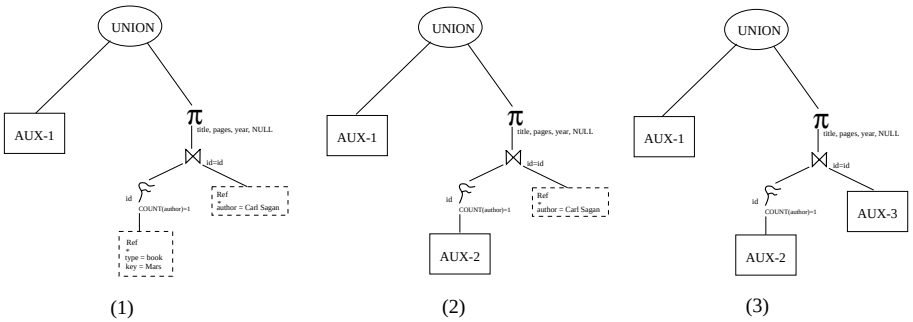


Figure 11. Updating the main plan after accessing data repositories

Step 3: Plan execution and retrieval of the final answer. For the execution of the main plan the Ontology Server utilizes a relational DBMS. The transformation of the leaves in the main plan from mono-repository mapping expressions into subplans whose leaves are names of relations in the auxiliary database results in the main plan that is a mono-repository plan (the repository is the auxiliary database). The main plan is now translated into a list of SQL sentences as shown next.

```
(SELECT *
 FROM AUX-1)
    UNION
(SELECT title, pages, year, NULL
 FROM AUX-3
 WHERE id IN (SELECT id
              FROM AUX-2
              GROUP BY author
              HAVING COUNT(author)=1))
```

The only remaining task is to execute such SQL sentences on the auxiliary database. The result will be the answer corresponding to the DL query formulated over the ontology that will be returned by the Ontology Server to the Query Processor.

6. Incremental Query Expansion to Multiple Ontologies

In this section we present a mechanism for incremental and controlled query expansion using additional ontologies in the GIS. If the user's need for information is not satisfied, the user has a choice to expand the scope of query processing to additional ontologies, which may lead to incremental access to more information.

The query expansion process starts with the selection of a new component ontology referred to as the *target ontology*. Ontologies in the same cluster as the user ontology will be chosen first as they are supposed to describe similar information. There does not exist an easy way to find out which ontology is the best before applying the method explained below.

Once a target ontology is selected, the user query needs to be expressed using terms of that target ontology; this means that all the terms in the user query (that belong to the user ontology) have to be substituted by equivalent or related terms of the target ontology. For this purpose, first the user and target ontologies are integrated by using the interontology relationships defined between them. In this process the user query $\mathcal{Q}$, which was introduced as a new term of the user ontology, is rewritten/translated into this new integrated ontology. If there still exist terms from the user ontology in the query, then the translation is called a *partial translation*. Partial translations can be dealt with in two ways:

- A partial translation can be combined with the partial translations obtained when visiting other target ontologies, such that the untranslated terms may be translated across multiple ontologies.
- Each conflicting term (for which no synonym is available) is substituted by the intersection of its immediate parents or by the union of its immediate children until the whole query is completely expressed in terms of the target ontology.

The new query can now be used to retrieve the data underlying the target ontology. The obtained data will be correlated with the previous answer and the user will decide again if s/he is satisfied with the new answer or, on the contrary, the system has to repeat the same process with another target ontology. In principle, the open and dynamic environments of the Web have a very large number of repositories, so this incremental enrichment of the answer could continue until all the ontologies (and therefore all the data repositories) in the GIS are visited. However, we discuss a strategy to control this expansion process so far as the results lead to a quantifiable improvement.

We now describe the main steps to get a translation of the user query into the "language" of the target ontology. Based on the example, we choose the Stanford-II ontology to demonstrate the process.

6.1. Integration of the user and target ontologies

Two types of relationships are considered in order to integrate the user and the target ontology.

1. Synonym, hyponym and hypernym relationships between terms in the user and target ontologies.

2. Synonyms, hyponyms and hypernyms within the user and the target ontologies.

The first three types of relationships are those stored in the IRM repository; they are defined when a new ontology is made available to the GIS. For example, if the target ontology is already registered as a component ontology and the user ontology is added to the system, then the relationships between the terms in these two ontologies are added to the IRM. The second three types are relationships between terms in the same ontology; synonyms are equivalent terms, hyponyms are terms subsumed by a given term and hypernyms are terms that subsume a given term, i.e., they can be obtained by a DL system after consulting the ontology definitions. The Query Processor requests the first kind of relationships from the IRM to integrate the user and target ontologies by using the deductive power of the DL system. These ontology descriptions will be provided by the Ontology Servers corresponding to each ontology involved in the process. Thus, no user intervention is required. The integration process is now discussed in detail.

Initial definitions:

$\mathcal{T} = Ont_{user} \bigcup Ont_{target}$

$\mathcal{R}$ = { semantic relationships between Ont_{user} and Ont_{target} stored in the IRM }

t^{descr} = Description of term t

Redescribe-Term(t, new-descr) substitutes the description of t by the new one

Delete-Term(t) removes t from all descriptions in which it appears and then removes the term t

1. Renaming of terms in user and target ontology:

   $\forall t \in Ont_{user} \Longrightarrow$ Rename-term(t, $Ont_{user}\#t$)

   $\forall t \in Ont_{target} \Longrightarrow$ Rename-term(t, $Ont_{target}\#t$)

2. Rewriting of term descriptions based on semantic relationships:
   $\forall r \in \mathcal{R}$
   (a) If $r \equiv t_1$ synonym t_2 $(t_1, t_2 \in \mathcal{T}) \Longrightarrow$
   $\forall t_1/t_1 \in t^{descr} \Longrightarrow$ Redescribe-Term (t, (**AND** t^{descr} t_2))
   Delete-Term(t_1)[8]
   (b) If $r \equiv t_1$ hyponym t_2 $(t_1, t_2 \in \mathcal{T}) \Longrightarrow$ Redescribe-Term (t_1, (**AND** t_1^{descr} t_2))
   (c) If $r \equiv t_1$ hypernym t_2 $(t_1, t_2 \in \mathcal{T}) \Longrightarrow$ Redescribe-Term (t_2, (**AND** t_2^{descr} t_1))
3. Assertion of updated terms from user and target ontologies in the DL system

Although some of the semantic relationships may be redundant, the DL system will classify the terms at the right place in the integrated ontology. As we use the deductive features of DL systems we avoid defining new *costly deductive algorithms* to determine the immediate hyponyms and hypernyms of a term. Rules described in [3] are applied to determine whether the resulting terms of the integrated ontology are *primitive* or *defined*. Apart from the complexity of the third step, which depends on the specific DL system, the complexity of this algorithm is $\mathcal{O}(kn)$ where $k = \mid \mathcal{R} \mid$ and $n = \mid \mathcal{T} \mid$.

Studies about the performance of the DL systems [34] show that it is possible to integrate two ontologies of around a thousand terms in less than a minute. In our example, the process of integrating in CLASSIC [5] (Lisp version) the ontologies BDI (24 terms) and Stanford-II (51 terms) taking into account 9 interontology relationships takes less than a second. Ontologies describing specific domains, as opposed to a global ontology are not expected to be huge since knowledge is distributed among several ontologies and combined when needed by our system. Thus, the integration can be performed at run time and lends favorably to the scalability of our query processing strategy.

The result of integrating the BDI and Stanford-II ontologies by applying the relationships defined between them is shown in Figure 12. Terms from user ontology (BDI) are in uppercase and those from the target ontology (Stanford-II) are in lowercase. Terms without parents in the figure are actually the immediate children of *Anything* (the top term in any ontology expressed in DL). Notice that the user query has also been rewritten ($\mathcal{Q}$ ='(**AND** *book-ref* (**FILLS** *author "Carl Sagan")* (**EXACTLY** *1 author)* (**FILLS** *keywords "Mars"))*') and classified in the right place by the DL system. In the case of synonyms (reference, REFERENCIAS), (book-ref, LIBRO), (technical-report, INFORME-INTERNO), (journal-article-ref, ARTICULO-REVISTA), (keywords, TEMAS), (year, FECHA), (document, FICHERO), (pages, PAGINAS) and (author, AUTORES), only the terms from Stanford-II (those in lower case) appear because we are translating from the BDI (user ontology) into the Stanford-II (target) ontology. Hereafter, the Query Processor will only deal with the integrated ontology since it contains all the needed information.

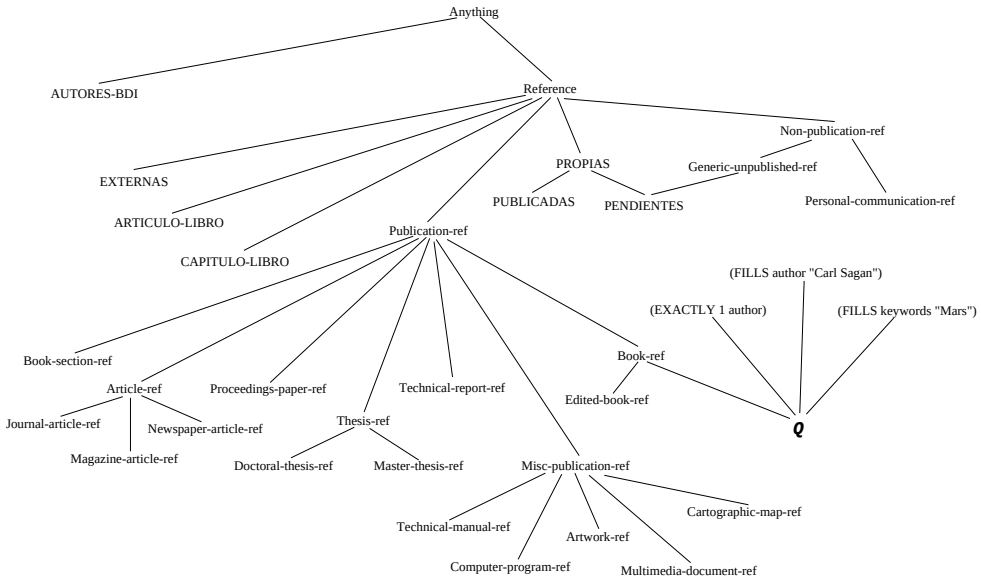


Figure 12. Integration of BDI and Stanford-II ontologies

6.2. Plans with no loss of information

A complete translation of the user query into the target ontology can be achieved in one of the two ways– a direct translation using synonym relationships or by combining partial translations.

6.2.1. Direct Translation Using Synonym Relationships. When the user and target ontologies are integrated, all terms in the user query may be translated by their corresponding synonyms. In our example, let us consider the first target ontology chosen to enrich the answer is Stanford-II. After integrating BDI and Stanford-II, the user query $\mathcal{Q}$ initially expressed in terms of the BDI ontology:

*[titulo paginas fecha fichero] for (***AND** *libro (***FILLS** *temas "Mars") (***EXACTLY** *1 autores) (***FILLS** *autores "Carl Sagan"))*

has been translated into Stanford-II ontology and rewritten as:

[title pages year document] for (AND book-ref (FILLS keywords "Mars") (EXACTLY 1 author) (FILLS author "Carl Sagan"))

All terms in $\mathcal{Q}$ belong to the target ontology so this case involves a *full translation*. This is because each term in the query expressed in the ontology BDI has a synonym (defined in the IRM) in the Stanford-II ontology. The underlying data is then accessed by the Ontology Server that corresponds to ontology Stanford-II and

correlated with data obtained from the BDI ontology and a new answer (containing more relevant data) is presented to the user.

6.2.2. Combining Partial Translations. After integration, the user query may not be completely translated (some terms in the query do not have synonyms in the target ontology). In this case, the partial translation obtained could be combined with partial translations into other target ontologies in order to get a new plan with no loss of information. Consider the case in which after visiting BDI (user ontology) and Stanford-II (first target ontology), the user wants more relevant data. Then ontology LSDIS is selected as the new target ontology. After integration of BDI and LSDIS ontologies, the user query $\mathcal{Q}$ is rewritten as follows:

Translation into LSDIS:
[title NULL date location-document] for (AND **libro** *(FILLS subject "Mars") (EXACTLY 1 authors) (FILLS authors "Carl Sagan"))*
non-translated constraints = { libro }

One term in $\mathcal{Q}$, 'libro' is not in the target ontology, so this is a partial translation. As the user does not allow imprecise answers, the only way to enrich the answer is to visit more ontologies looking for a new full translation. We now present an interesting theorem which enables us to determine when a combination of partial translations is logically equivalent to a query. The theorem has been rigorously proved in [27].

Theorem: *Given a user query $\mathcal{Q}$ and a set of partial translations of that query, if the intersection of the non-translated parts is empty then the intersection of the objects of the translated parts will satisfy all the constraints in $\mathcal{Q}$.*

In the [28] we presented an algorithm which, given a new partial translation, tries to determine whether it can be combined with any combination of the previously obtained partial translations. If the number of constraints of a given user query is k, the previous algorithm will never construct combinations of more than k-1 elements/partial translations since only non-redundant combinations[9] are considered. This reduces the explosion of the search space. So the algorithm returns a list of minimal combinations which are equivalent to full translations (the combinations returned satisfy the theorem).

If k is the number of constraints in the user query and i is the number of partial translations found (remember that the system obtains a new (partial) translation each time it translates the user query into a new target ontology), the complexity of the algorithm is the following:

$$complexity = \begin{cases} O(k2^i) & 1 \leq i \leq k \\ O(ki2^k) & i > k \end{cases}$$

This means that the algorithm is exponential only in the first k ontologies visited. Notice that while the number of ontologies available in a GIS can be large, the number of constraints in a query is usually a small number (less than ten). Continuing with our example, the ontology WN is taken as the third target ontology and the result of integrating BDI and WN ontology is:

Translation into WN:
[name pages NULL NULL] for (AND **libro** *(FILLS general-topics "Mars") (EXACTLY 1 creator) (FILLS creator "Carl Sagan"))*
non-translated constraints = { libro }

As it happened with LSDIS, the term 'libro' has no translation so the system stores it as a new partial translation. Let us see if the two partial translations satisfy the theorem:

INTERSECTION(non-translated constraints in WN, non-translated constraints in LSDIS) = libro

They do not constitute a full translation since the intersection of their non-translated parts are not empty. In other words if we retrieve the data corresponding to both partial translations and perform the intersection between them, we are not sure that the result corresponds to publications that are books due to constraint 'libro' has not been verified by either of the two translations. So, the system continues visiting a new target ontology. Stanford-I is chosen and the user query, after integration of BDI and Stanford-I, is rewritten in the following manner:

Translation into Stanford-I:
[title number-of-pages NULL NULL] for (AND book **(FILLS temas "Mars")** *(EXACTLY 1 doc-author-name) (FILLS doc-author-name "Carl Sagan"))*
non-translated constraints = { (FILLS temas "Mars") }

Another new partial translation. Let us see what happens with the combinations:

- INTERSECTION(non-translated constraints in Stanford-I,non-translated constraints in LSDIS)= { }. They constitute a new full translation!

  Objects(Stanford-I, '(AND book (EXACTLY 1 doc-author-name) (FILLS doc-author-name "Carl Sagan"))')

  $\bigcap$

  Objects(LSDIS, '(AND (FILLS subject "Mars") (EXACTLY 1 authors) (FILLS authors "Carl Sagan"))')

  satisfies all the constraints in the user query

- INTERSECTION(non-translated constraints in Stanford-I, non-translated constraints in WN) = { }. They constitute a new full translation !

  Objects(Stanford-I, '(AND book (EXACTLY 1 doc-author-name) (FILLS doc-author-name "Carl Sagan"))')

  $\bigcap$

  Objects(WN, '(AND (FILLS general-topics "Mars") (EXACTLY 1 creator) (FILLS creator "Carl Sagan"))')

  satisfies all the constraints in the user query

Here the system obtains two combinations that constitute two new full translations. All the constraints in the original user query are now verified in one of the two partial translations that constitute each combination. Thus, the intersection of the data corresponding to each partial translation in each of the two combinations satisfies all the constraints of the user query, as per the theorem discussed above. This checking can be performed in parallel with a translation process (explained later) that uses hyponym and hypernym relationships and incurs a loss of information as these relationships result in changing the semantics of the query. The reason to deal with loss of information is that sometimes there may not exist synonym relationships between terms in independently developed ontologies. In that case, it is better to return some information with an estimate of information loss rather than no information at all.

6.3. Plans with loss of information

After translation of the user query into the integrated ontology, there may be terms of the user ontology for which there did not exist synonyms into the target ontology. Each conflicting term in the user query is then replaced by the intersection of its immediate parents or by the union of its immediate children. This method is applied recursively until a translation of the conflicting term is obtained using only the terms of the target ontology. Note that it is always possible to get at least one full translation of any conflicting term in both the directions since the terms *Anything* and *Nothing* exist in any ontology. This procedure changes the semantics of the query resulting in loss of information in the answer returned to the user. Techniques for measuring the loss of information are discussed in the next section. For each visited term, the system stores its plans/translations so that each term is explored only once. The complexity of this algorithm is $\mathcal{O}(e)$ where e is the number of edges in the integrated ontology where the translation is performed.

Traversing hyponym and hypernym relationships as described above can result in several possible translations for each conflicting term. All the possibilities are explored and the result is a list of tuples in the format $< Plan, Loss >$, where *Plan* is a DL expression using only terms from the target ontology, and *Loss* is a number between 0 and 100 representing the percentage loss of information of *Plan* with respect to the original user query. All possible plans are evaluated and the loss of information incurred by each plan is estimated [16]. Redundant plans are removed using the following rule:

$$< Plan1, Loss1 > \subset < Plan2, Loss2 > \Leftrightarrow Plan1 \subset Plan2 \wedge Loss1 > Loss2$$

Based on the above rule, $< Plan1, Loss1 >$ will be eliminated. If the first condition is not satisfied, *Plan1* could bring new relevant objects and hence needs to be considered. If the second condition is not satisfied, *Plan1* will be chosen before *Plan2* as it has less loss. After the removal of redundant plans, the one with less loss is chosen to access new relevant data.

6.3.1. Example: Generation of Plans with Loss of Information. To demonstrate the different cases that can arise, let us consider a query by another user– "*Retrieve title and number of pages of all the books written by Carl Sagan*". This user chooses WN as user ontology and allows the system to provide imprecise answers and limits the maximum loss to 50%, i.e., using the qyery editor the user asserts that at least fifty percent of the objects retrieved should satisfy the constraints in the user query. This is the query in DL:

$\mathcal{Q}$ = *[NAME PAGES] for (***AND** *BOOK (***FILLS** *CREATOR "Carl Sagan"))*

After accessing the data underlying WN the user wants more data and the system chooses Stanford-I as target ontology. The user query has to be translated into terms of the Stanford-I ontology. After the process of integrating the WN and Stanford-I ontologies (Figure 13), $\mathcal{Q}$ was redefined as follows:

$\mathcal{Q}$ = *[title number-of-pages] for (***AND** *BOOK (***FILLS** *doc-author-name "Carl Sagan"))*

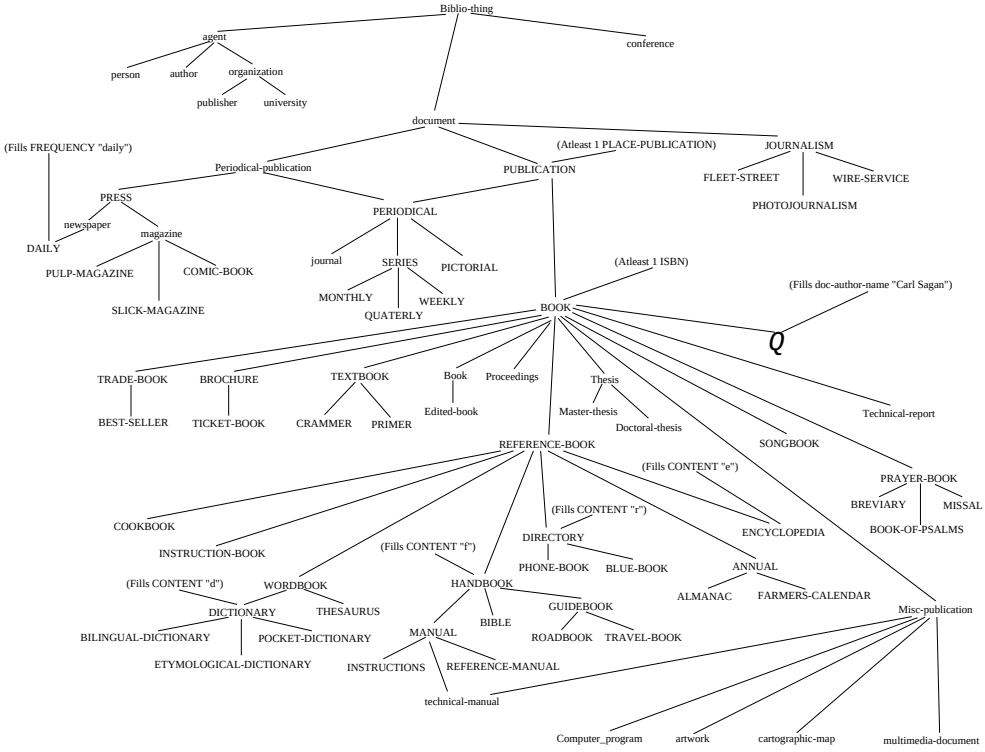


Figure 13. Integration of WN and Stanford-I ontologies

The roles to be projected were renamed in the integration process as 'title' and 'number-of-pages', since they are synonyms of 'NAME' and 'PAGES', respectively.

As $\mathcal{Q}$ has no children, the only way is to substitute it by the intersection of its immediate parents, i.e., 'BOOK' and '(**FILLS** doc-author-name "Carl Sagan")'. Notice that, as 'CREATOR' has a synonym into Stanford-I ('doc-author-name'), it was renamed and the user query $\mathcal{Q}$ was described in the integrated ontology as (**AND** BOOK (**FILLS** doc-author-name "Carl Sagan"). The only remaining conflicting term is 'BOOK' for which the system applies the translation process:

1. Intersection of parents. 'PUBLICATION' and '(**ATLEAST** 1 ISBN)') are the parents of 'BOOK'. The role 'ISBN' is not from the Stanford-I ontology, hence that constraint can be ignored. The concept 'PUBLICATION' is not in the target ontology and has to be translated. Two possibilities are:

   (A) Intersection of parents. '(**ATLEAST** 1 PLACE-PUBLICATION)' and 'document' are the parents of 'PUBLICATION'. The ATLEAST constraint is ignored (no translation for 'PLACE-PUBLICATION') so 'document', a term of the target ontology, is identified as a plan for 'PUBLICATION'.

   (B) Union of children. 'PERIODICAL' and 'BOOK' are the children of 'PUBLICATION'. Note that 'BOOK' is not taken as it is the term we were trying to translate initially. The two possible translations of 'PERIODICAL' are:

      i. Intersection of parents. 'periodical-publication' and 'PUBLICATION' are the parents of 'PERIODICAL'. 'PERIODICAL' is ignored as it is the term we were trying to translate initially and 'periodical-publication', a term in the target ontology, is identified as a plan for 'PERIODICAL'.

      ii. Union of children. 'journal', 'SERIES', and 'PICTORIAL'are the children of 'PERIODICAL'. 'SERIES' and 'PICTORIAL' lead to the bottom concept *Nothing*, so they are ignored by the union operation. The union of children is 'journal', a term of the target ontology.

   plans(PUBLICATION) = { document, periodical-publication, journal }.

2. Union of children. 'TRADE-BOOK', 'BROCHURE', 'TEXTBOOK', 'Book', 'Proceedings', 'REFERENCE-BOOK', 'Thesis', 'Misc-publication', 'SONGBOOK', 'PRAYER-BOOK' and 'Technical-report' are the children of 'BOOK'. Terms 'TRADE-BOOK', 'BROCHURE', 'TEXTBOOK', 'SONGBOOK' and 'PRAYER-BOOK' lead to *Nothing* and are ignored. The only term that needs to be translated is 'REFERENCE-BOOK'. We do not take the intersection of its parents as 'BOOK' is its only parent, and that is what we were trying to translate initially. The only option is:

   (A) Union of the children. 'COOKBOOK', 'INSTRUCTION-BOOK', 'WORD-BOOK', 'HANDBOOK', 'DIRECTORY', 'ANNUAL' and 'ENCYCLOPEDIA' are the children of 'REFERENCE-BOOK' that lead to *Nothing*[10]. 'HANDBOOK' is substituted by 'technical-manual', following a similar procedure.

plans(BOOK) = { document, periodical-publication, journal,
UNION(book, proceedings, thesis, misc-publication[11], technical-report) }

The possible plans for the query '(**AND** BOOK (**FILLS** doc-author-name "Carl Sagan"))' are:

1. < (**AND** document (**FILLS** doc-author-name "Carl Sagan")) >
2. < (**AND** periodical-publication (**FILLS** doc-author-name "Carl Sagan")) >
3. < (**AND** journal (**FILLS** doc-author-name "Carl Sagan")) >
4. < (**AND** UNION(book, proceedings, thesis, misc-publication, technical-report)
(**FILLS** doc-author-name "Carl Sagan")) >

The computation of the loss of information for each plan is illustrated in Section 7.

7. Estimating the Loss of Information

In this section, we describe the mechanism used to measure the change in semantics when a term in a query is replaced by an expression from another ontology (in an attempt to obtain a full translation of the user query). The loss of information associated with this substitution can be limited by the user by defining the maximum loss allowed (as a percentage). The system guarantees that the answer it builds incrementally is always under such a limit; the user can change the limit when s/he wants.

There have been approaches in the research literature for approximating query answering in situations where multiple answers may be obtained from multiple information sources. Most approaches are typically accompanied by an attempt to estimate some measure of divergence from the true answer and are based on modeling uncertainty. In the Multiplex project [30], the soundness and completeness of the results are estimated based on the intersections and unions of the candidate results. In our approach, the information retrieval analogs of soundness (precision) and completeness (recall) are estimated based on the sizes of the extensions of the terms. We combine these two measures to compute a composite measure in terms of a numerical value. This can then be used to choose the answers with the least loss of information. Numerical probabilistic (possibilistic) measures are on the other hand used in [36, 11], but are based on *ad hoc* estimates of the initial probability (possibility) values. In our approach we provide a set theoretic basis for the estimation of information loss measures.

In our case, the change in semantics caused by the use of hyponym and hypernym relationships are measured not only to decide which substitution minimizes the loss of information but also to present to the user some kind of "level of confidence" in the new answer. This would enable the system to navigate those ontologies which contain more relevant information for the user needs. In this section, we define and illustrate with examples, measures for estimating the loss of information. First, we present a way of measuring the change in semantics based on intensional

information, and second, a technique that measures the change in semantics based on extensional information. Both measures are presented to the user whenever a new answer is obtained.

7.1. Loss of Information measurement based on intensional information

In our context, and due to the use of DL systems, loss of information can be expressed as the terminological difference between two expressions, the user query and its translation. The terminological difference between two expressions is those constraints of the first expression that are not subsumed by the second expression. The DL system is able to calculate the difference automatically[12]. Let us show an example based on the plans obtained in Section 6.3.1:

Original query: $\mathcal{Q}$ =*[NAME PAGES] for (***AND** *BOOK (***FILLS** *CREATOR "Carl Sagan"))*

Plan 1: $\mathcal{Q}$ =*[title number-of-pages] for (***AND** *document (***FILLS** *doc-author-name "Carl Sagan"))*

Taking into account the following term definitions[13]:

*BOOK = (***AND** *PUBLICATION (***ATLEAST** *1 ISBN))*
*PUBLICATION = (***AND** *document (***ATLEAST** *1 PLACE-PUBLICATION))*

The terminological difference is, in this case, the constraints ignored in the translation process:

(**AND** (**ATLEAST** 1 ISBN) (**ATLEAST** 1 PLACE-PUBLICATION))

A special problem arises when computing loss based on intensional information due to the vocabulary differences. As the loss is expressed using terms of two different ontologies, the explanation might make no sense to the user as it mixes two "vocabularies". Suppose an example in which the term "book" in one user ontology is a hypernym of "book" in another ontology restricted to medical domain ('book' in the user ontology is a hypernym of 'book' in the medical ontology). The explanation "Only 'book' is retrieved instead of 'book' (original query)" does not make any sense because both terms would be seen as homonyms by the user. The problem could be even worse if both ontologies were expressed in different natural languages. Thus intensional information can help to understand the loss only in some cases. We re-visit the plans found in the example of Section 6.3.1 and enumerate the loss (based on intensional information) incurred in plan 1 and plan 4. Cases 2 and 3 are similar to the one for plan 1.

1. Plan 1 = [title number-of-pages] for (**AND** document (**FILLS** doc-author-name "Carl Sagan"))

   Loss="Instead of books written by Carl Sagan, OBSERVER provides all the documents (even if they do not have an ISBN and place of publication)[14] written by Carl Sagan."

2. Plan 4 = [title for number-of-pages] for (**AND** UNION(book, proceedings, thesis, misc-publication, technical-report) (**FILLS** doc-author-name "Carl Sagan"))

   Loss="Instead of books written by Carl Sagan, OBSERVER provides the books[15], proceedings, theses, misc-publication and technical manuals written by Carl Sagan. Any book not included in this group is not retrieved."

In addition to the vocabulary problem, an intensional measure of the loss of information can make it hard for the system to decide between two alternatives, such that the plan with less loss is executed first. Thus, a numeric measure of the loss of information is highly desirable.

7.2. Loss of Information measurement based on extensional information

We also measure the loss of information based on the number of instances of terms involved in the substitutions performed on the query. Since the measure depends on the sizes of the term extensions, we first discuss techniques to estimate the extensions of complex expressions based on set theoretic operations such as unions and intersections. Second, we briefly describe a composite measure combining measures like *precision* and *recall* [33] that is used to estimate the information loss when a term is substituted by an expression. The composite measure defined takes into account the bias of the user as to whether precision is more important than recall. And third, we present our proposal for adapting these measures based on semantic relationships between the various expressions. We give priority to semantic relationships before resorting to extensional information because, for instance, it can happen that a term in one ontology is semantically more general than another term in another ontology; at the same time the subsumer term can have less instances than the subsumed term because they belong to different ontologies and take instances from different data repositories that are not necessarily related. In Section 7.3 real examples of these cases are shown and above techniques are illustrated by evaluating the loss of information for the plans generated in Section 6.3.1.

7.2.1. Estimating the size of the extension of an expression. Given an expression, let us say $Expr$, considered as a translation of a conflicting term, we need to approximate the size of its extension, denoted by $|\text{Ext}(Expr)|$, in order to calculate a numeric difference between retrieving the objects that belong to $Expr$ and retrieving the objects corresponding to the conflicting term. The expression is a combination of unions and intersections of terms in the target ontology since at each translation step, the system substitutes conflicting terms by the intersection of its parents or by the union of its children. The estimate is an interval with an upper (|Ext(Expr)|.high) and lower (|Ext(Expr)|.low) bound. The computation details are presented next.

- The intersection of two sets can be empty at the least (no overlap). At the most, the intersection of two sets can only be the smaller of the two sets (maximum overlap).

|Ext(Subexpr1) ∩ Ext(Subexpr2)|.low = 0
|Ext(Subexpr1) ∩ Ext(Subexpr2)|.high = min[|Ext(Subexpr1)|.high, |Ext(Subexpr2)|.high]

- The union of two sets can be at the least the bigger of the two sets (maximum overlap). At the most the size of the union can be the sum of the sizes of the two sets (no overlap).
  |Ext(Subexpr1) ∪ Ext(Subexpr2)|.low = max[|Ext(Subexpr1)|.low, |Ext(Subexpr2)|.low]
  |Ext(Subexpr1) ∪ |Ext(Subexpr2)|.high = |Ext(Subexpr1)|.high + |Ext(Subexpr2)|.high

As a trivial case, when an expresion *Expr* is the name of a term T, both bounds are equal to the size of the extension of such a term, i.e., the number of objects residing in the underlying data repositories that satisfy the constraints that constitute T.

$$\text{if } Expr = T \Longrightarrow |\text{Ext(Expr)}| = |\text{Ext(T)}|$$

The information about the extension of single terms is retrieved, stored and updated periodically by the system, by consulting the underlying repositories. Intervals when calculating the extension lead to intervals for the resulting precision, recall and loss of information. If overlap relationships were available in the IRM between terms involved in expressions for which the system is estimating the size, they will be used to get a more exact approximation of the bounds of the intervals.

7.2.2. A Composite Measure combining Precision and Recall. Precision and Recall have been very widely used in the information retrieval literature to measure loss of information incurred when the answer to a query issued to the information retrieval system contains some proportion of *irrelevant* data [33]. These measures are defined, and adapted to our context, as follows:
C-Term = conflicting term to be translated into the target ontology
Ext(C-Term) = extension underlying C-Term = relevant objects[16] (RelevantSet)
Expression = translation with loss of the term
Ext(Expression) = extension underlying Expression = retrieved objects (RetrievedSet)

$$\textbf{Precision} = \textit{proportion of the retrieved objects that are relevant} = \text{Probability(Relevant|Retrieved)} = \frac{|RetrievedSet \cap RelevantSet|}{|RetrievedSet|} = \frac{|Ext(C-Term) \cap Ext(Expression)|}{|Ext(Expression)|}$$

$$\textbf{Recall} = \textit{proportion of relevant objects that are retrieved} = \text{Probability(Retrieved|Relevant)} = \frac{|RetrievedSet \cap RelevantSet|}{|RelevantSet|} = \frac{|Ext(C-Term) \cap Ext(Expression)|}{|Ext(C-Term)|}$$

Based on the above we use a composite measure [8] which combines the precision and recall to estimate the loss of information. We seek to measure the extent to which the two sets do not match. This is denoted by the shaded area in Figure 14. The area is, in fact, the symmetric difference:
RelevantSet Δ *RetrievedSet* = *RelevantSet* $\cup$ *RetrievedSet* - *RelevantSet* $\cap$ *RetrievedSet*
We are interested in the proportion (rather than the absolute number) of relevant and non-relevant objects retrieved, so a normalization of the measure gives:

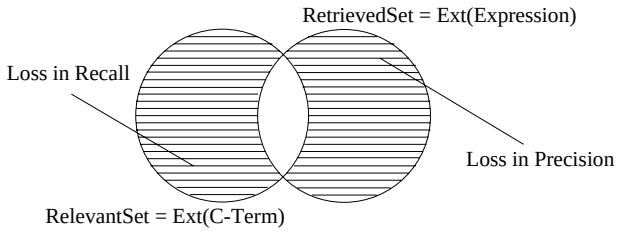


Figure 14. The mismatch between the RetrievedSet and RelevantSet

Loss = $\frac{|RelevantSet \Delta RetrievedSet|}{|RelevantSet|+|RetrievedSet|}$
In terms of precision and recall we have: Loss = 1 - $\frac{1}{\frac{1}{2}(\frac{1}{Precision})+\frac{1}{2}(\frac{1}{Recall})}$
In an open and dynamic environment, it is important to satisfy the information needs of a widely varying cross-section of users. The users may have widely varying preferences when it is necessary to choose between precision and recall. We introduce a parameter α $(0 \leq \alpha \leq 1)$ to capture the preference of the user where α denotes the importance attached by a user to precision. The modified composite measure may now be given as: Loss = 1 - $\frac{1}{\alpha(\frac{1}{Precision})+(1-\alpha)(\frac{1}{Recall})}$

7.2.3. Extensional Information vs. Semantic Relationships: Semantic adaptation for precision and recall measures. Techniques for estimating precision appear in the information retrieval literature, but our work differs in the following important aspect: we give higher priority to semantic relationships than those suggested by the underlying extensions. Only when the semantics are not available, the system resorts to the use of extensional information. Since the system has translated a term from one ontology into an expression with terms from another ontology with different underlying repositories, the extensional relationships may not reflect the semantic relationships. For instance, a term in a user ontology which semantically[17] subsumes a term in the target ontology may have a smaller extension than the child term. This is reflected in the proposed measures.

We now enumerate the various cases that arise depending on the relationship between the conflicting term and its translation and present measures for estimating the information loss. We assume that a **Term** is translated into an **Expression** in the integrated ontology. The critical step here is to estimate the extension of **Expression** based on the extensions of the terms in the target ontology. Precision and recall are adapted as follows:

1. Precision and recall measures for the case where a term subsumes its translation: Semantically, we do not provide an answer irrelevant to the term, as Ext(Expression) $\subseteq$ Ext(Term) (by definition of subsumption). Thus, as Term subsumes Expression $\Rightarrow$ Ext(Term) $\cap$ Ext(Expression) = Ext(Expression). Therefore:

Precision = 1,
Recall = $\frac{|Ext(Term) \cap Ext(Expression)|}{|Ext(Term)|} = \frac{|Ext(Expression)|}{|Ext(Term)|}$

Since terms in *Expression* and *Term* are from a different ontology, the extension of *Expression* can be bigger than the extension of *Term*, although *Term* subsumes *Expression* semantically. In this case we consider the extension of *Term* to be: |Ext(Term)| = |Ext(Term) ∪ Ext(Expression)|. Thus recall can be defined as:
Recall.low = $\frac{|Ext(Expression)|.low}{|Ext(Expression)|.low + |Ext(Term)|}$,
Recall.high = $\frac{|Ext(Expression)|.high}{max[|Ext(Expression)|.high, |Ext(Term)|]}$

2. Precision and recall measures for the case where a term is subsumed by its translation: Semantically, all elements of the term extension are returned, as Ext(Term) ⊆ Ext(Expression) (by definition of subsumption). Thus, as Expression subsumes Term ⇒ Ext(Term) ∩ Ext(Expression) = Ext(Term), The calculus of precision is like the one for recall in the previous case. Therefore:
Recall = 1,
Precision.low = $\frac{|Ext(Term)|}{|Ext(Expression)|.high + |Ext(Term)|}$,
Precision.high = $\frac{|Ext(Term)|}{max[|Ext(Expression)|.low, |Ext(Term)|]}$

3. *Term* and *Expression* are not related by any subsumption relationship: The general case is applied directly since intersection cannot be simplified. In this case the interval describing the possible loss will be wider as *Term* and the *Expression* are not related semantically[18].
Precision.low = 0,
Precision.high = max[$\frac{min[|Ext(Term)|, |Ext(Expression)|.high]}{|Ext(Expression)|.high}$, $\frac{min[|Ext(Term)|, |Ext(Expression)|.low]}{|Ext(Expression)|.low}$]
Recall.low = 0, Recall.high = $\frac{min[|Ext(Term)|, |Ext(Expression)|.low]}{|Ext(Term)|}$

Two special cases can arise in which the substitution of a term by an expression does not imply any loss:

1. Substituting a term by the intersection of its immediate parents implies no loss of information if it was defined as *exactly* its definition[19], i.e., the term and the intersection of its parents are semantically equivalent. For instance, in the example 'BOOK' was defined as exactly '(**AND** PUBLICATION (**ATLEAST** 1 ISBN))' and therefore the substitution of 'BOOK' by its immediate parents implies no loss.

2. Substituting a term by the union of its children implies no loss of information if there exists an extensional relationship saying that the term is covered extensionally by its children (total generalization).

7.3. Example of translation and measurement of the extensional loss

We now illustrate the computation of precision, recall and loss of information for each plan presented in Section 6.3.1. As 'BOOK' is the only conflicting term in the

translation (i.e., the only one with no synonym), we explore the different translations for this term (no loss was incurred until replacing 'BOOK'). For the sake of discussion, we assume α=0.5 (equal importance to precision and recall)[20] and the maximum loss allowed is 50%. The extensional values have been obtained from the underlying data repositories.

1. The loss of information incurred on substitution of 'BOOK' by 'document' is as follows; it is an example of case 2 explained in Section 7.2.3 since 'BOOK' is subsumed by 'document' [25].

   |Ext(BOOK)|=1105, |Ext(document)|=24570
   Precision.low=$\frac{|Ext(BOOK)|}{|Ext(BOOK)|+|Ext(document)|}$=0.043,
   Precision.high=$\frac{|Ext(BOOK)|}{max[|Ext(BOOK)|,|Ext(document)|]}$=0.044,
   Recall=1,
   Loss.low=1-$\frac{1}{\frac{\alpha}{Precision.high}+\frac{(1-\alpha)}{Recall.high}}$=0.91571,
   Loss.high=1-$\frac{1}{\frac{\alpha}{Precision.low}+\frac{(1-\alpha)}{Recall.low}}$=0.91755

2. The loss of information incurred on substitution of 'BOOK' by 'periodical-publication' is as follows; it is an example of case 3 in Section 7.2.3 since 'BOOK' and 'periodical-publication' are not related (none of them subsumes each other).

   |Ext(BOOK)|=1105, |Ext(periodical-publication)|=34
   Precision.low = 0,
   Precision.high = max [$\frac{min[|Ext(Term)|,|Ext(Expression)|.high]}{|Ext(Expression)|.high}$, $\frac{min[|Ext(Term)|,|Ext(Expression)|.low]}{|Ext(Expression)|.low}$] = 1
   Recall.low = 0, Recall.high = $\frac{min[|Ext(Term)|,|Ext(Expression)|.low]}{|Ext(Term)|}$ = 0.03076
   Loss.low=1-$\frac{1}{\frac{\alpha}{Precision.high}+\frac{(1-\alpha)}{Recall.high}}$=0.94031, Loss.high=1-$\frac{1}{\frac{\alpha}{Precision.low}+\frac{(1-\alpha)}{Recall.low}}$=1

3. The loss of information incurred on substitution of 'BOOK' by 'journal' is as follows; it is another example of case 3 in Section 7.2.3.

   |Ext(BOOK)|=1105, |Ext(journal)|=8
   Precision.low = 0,
   Precision.high = max [$\frac{min[|Ext(Term)|,|Ext(Expression)|.high]}{|Ext(Expression)|.high}$, $\frac{min[|Ext(Term)|,|Ext(Expression)|.low]}{|Ext(Expression)|.low}$] = 1
   Recall.low = 0, Recall.high = $\frac{min[|Ext(Term)|,|Ext(Expression)|.low]}{|Ext(Term)|}$ = 0.00723
   Loss.low=1-$\frac{1}{\frac{\alpha}{Precision.high}+\frac{(1-\alpha)}{Recall.high}}$=0.98564, Loss.high=1-$\frac{1}{\frac{\alpha}{Precision.low}+\frac{(1-\alpha)}{Recall.low}}$=1

4. The loss of information incurred by considering the children of 'BOOK' in the integrated ontology is as follows; 'BOOK' subsumes the union since it subsumes each of them separately, although the extension of 'BOOK' (1105) is smaller than the extension of the union (between 14199 and 14237). It is an example of case 1 in Section 7.2.3.

   |Ext(BOOK)|=1105, |Ext(book)|=14199, |Ext(proceedings)|=6, |Ext(thesis)|=0,
   |Ext(misc-publication)|=31, |Ext(technical-report)|=1
   Ext-union.low=max[|Ext(book)|, |Ext(proceedings)|, ...]=14199,

Ext-union.high=sum[|Ext(book)|, |Ext(proceedings)|, ...]=14237
Ext-expr.low=$\frac{Ext-union.low}{|Ext(BOOK)|+Ext-union.low}$=0.92780,
Ext-expr.high=$\frac{Ext-union.high}{|Ext(BOOK)|+Ext-union.high}$=0.92798,
Precision=1
Recall.low=$\frac{Ext-expr.low}{Ext-expr.low+|Ext(BOOK)|}$=0.92780,
Recall.high=$\frac{Ext-expr.high}{max[|Ext(BOOK)|,Expr-ext.high]}$=[21] 1
Loss.low=1-$\frac{1}{\frac{\alpha}{Precision.high}+\frac{(1-\alpha)}{Recall.high}}$=0, Loss.high=1-$\frac{1}{\frac{\alpha}{Precision.low}+\frac{(1-\alpha)}{Recall.low}}$=0.07220

Table 2. The various plans and the respective Loss Of Information

Plan	**Loss Of Information**
(**AND** document (**FILLS** doc-author-name "Carl Sagan"))	91.57%≤loss≤91.75%
(**AND** periodical-publication (**FILLS** doc-author-name "Carl Sagan"))	94.03%≤loss≤100%
(**AND** journal (**FILLS** doc-author-name "Carl Sagan"))	98.56%≤loss≤100%
(**AND** UNION(book, proceedings, thesis, misc-publication, technical-report) (**FILLS** doc-author-name "Carl Sagan"))	0%≤loss≤7.22%

The four possible plans and the respective losses for the user query '(**AND** BOOK (**FILLS** doc-author-name "Carl Sagan"))' are illustrated in Table 2. Only the fourth case results in the loss below the *user-max-loss* (50%) and is hence chosen. This means that the chosen translation into Stanford-I of the original user query '*[NAME PAGES] for (***AND** *BOOK (***FILLS** *CREATOR "Carl Sagan"))*' is '*[title number-of-pages] for (***AND** *UNION(book, proceedings, thesis, misc-publication, technical-report) (***FILLS** *doc-author-name "Carl Sagan"))*'. The answer does not contain incorrect data in the best case (which is possible) but around a 7% of the ideal answer may be missed or substituted by irrelevant data in the worst case.

8. Conclusions

As the Web becomes the predominant environment for more and more people to create applications and export information, syntactic approaches for navigation and keyword based searches are becoming increasingly inadequate. We have presented an approach which attempts to enhance the scalability of query processing in a global information system and at the same time enables the user to specify an information request in terms of semantic concepts. The novel contributions of our approach are:

- Use of pre-existing domain ontologies to describe information in the underlying data repositories. This enables the user to view the repositories at the level of its relevant semantic concepts. Thus an information request can now be expressed using these concepts and the user can now browse multiple domain ontologies as opposed to individual heterogeneous repositories or concepts based on statistical information.

- Use of brokering across domain ontologies to enhance the scalability of the query processing approach. Semantic relationships across terms in ontologies are stored and utilized to perform the brokering, thus avoiding the need to have a global schema or collection of concepts.
- Management of answers that have an associated loss of information as limited by the user and measurement of such a loss to provide answers with the least loss.

In this paper, we discuss how a query expressed using terms in one ontology can be translated into a query involving terms from other ontologies. We also characterize the cases where such a translation would involve a modification in the semantics of the query and propose measures to estimate the resulting loss of information. Based on this estimate we choose that translation which minimizes the loss of information. These techniques enable query processing without the need of a common or global collection of concepts thus enhancing the scalability of the process. The answer is upgraded in an incremental manner. We propose an architecture that supports scalable concept-based query processing as:

- The extensions of terms in different ontologies based on the data repositories underlying the respective ontologies can be combined according to semantic relationships between them. This is more scalable than computing the mappings of the terms in all the ontologies to all the data repositories, an approach which applies to the global ontology approach.
- The number of semantic relationships between terms in different ontologies is an order of magnitude less than the all the terms and their interrelationships in the global ontologies.

In our work we have used real-world ontologies (developed independently) to describe real-world repositories from the domain of bibliographic information. We have synthesized technologies such as information brokerage and domain ontologies with internet computing to design and implement a prototype, thus demonstrating modest but concrete progress in developing semantics-based information access on the Web. The performance of the prototype developed has demostrated that the bottle-neck of the system is the access to the distributed data repositories.

Finally, OBSERVER is being used as a part of a project funded by the Government of the Basque Country, whose goal is to create and manage a GIS of Eusko Ikaskuntza, an organization that compiles and manages data about the basque culture.

Appendix

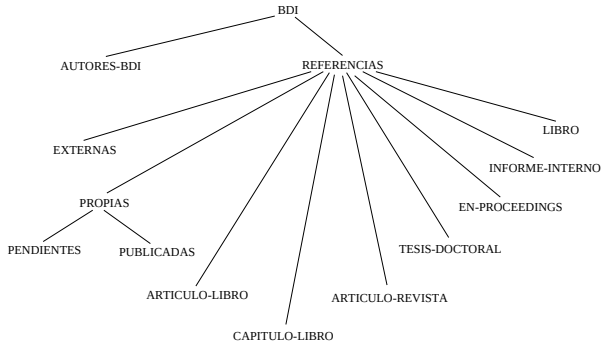


Figure A.1. BDI: the BDI database group ontology

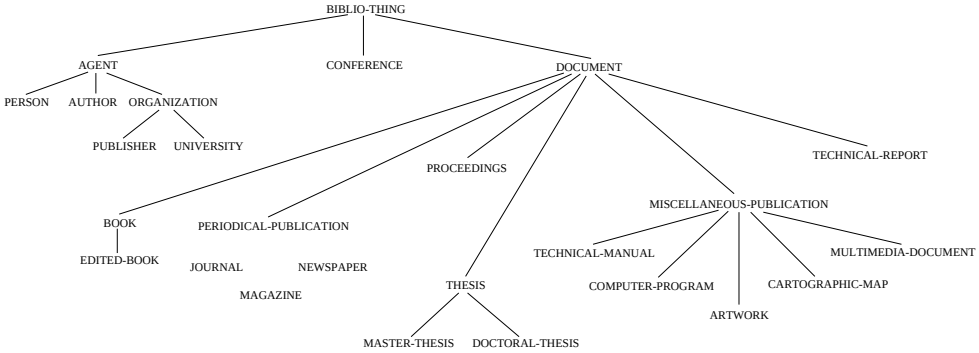


Figure A.2. Stanford-I: A subset of the Bibliographic-data ontology

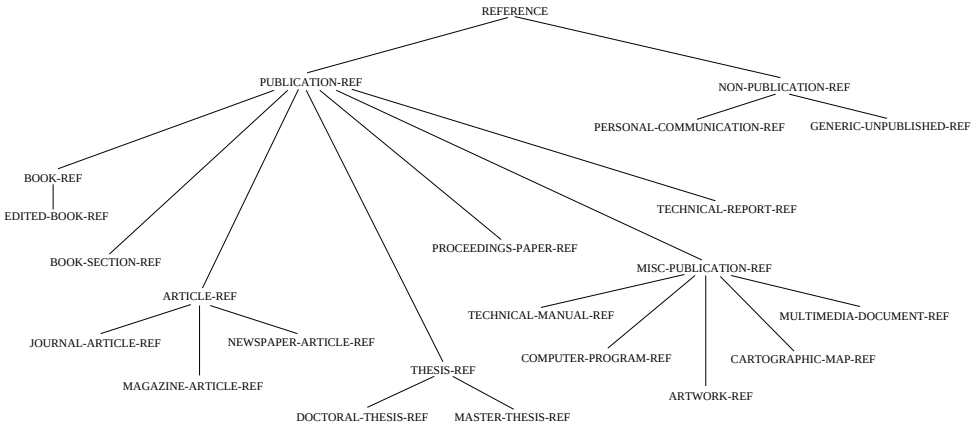


Figure A.3. Stanford-II: A subset of the Bibliographic-data ontology

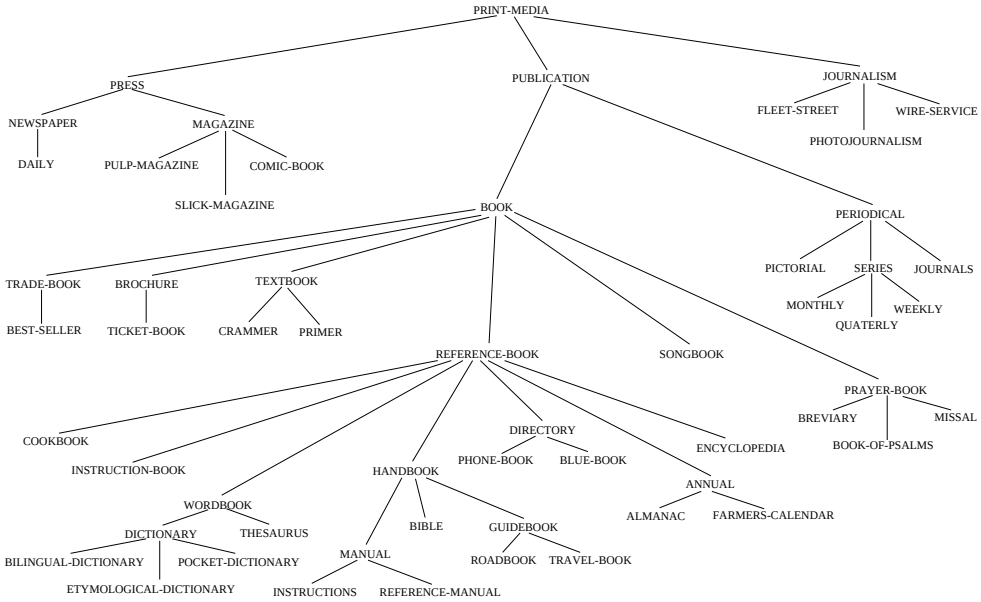


Figure A.4. WN: A subset of the WordNet ontology

Notes

1. This task must be performed by ontology administrators and repository administrators, i.e., by those who have complete knowledge of the semantics of ontologies and repositories.
2. CLASSIC does not support *defined roles* but other DL systems, like BACK, do.
3. Just to make the understanding of the process easier, the spanish words 'título', 'páginas', 'fecha', 'fichero', 'libro', 'temas' and 'autores' mean 'title', 'pages', 'date', 'file', 'book', 'subjects' and 'authors', respectively.
4. The DL operator 'EXACTLY' restricts the cardinality of the role to the indicated. 'FILLS' guarantees that the indicated role has at least the indicated values.
5. Due to syntactic clarity, we borrow the syntax from BACK to express the creation of a defined concept (new-concept := definition).
6. If the underlying data repository is a relational database, any mapping expression can be translated into a list of SQL sentences.
7. Data corresponding to attribute 'fichero', which are not stored in the database but in Postscript files, are obtained by the wrapper after accessing the database.
8. We are only interested in terms in the target ontology as we are looking for substituting conflicting terms of the user ontology by terms in the target ontology.
9. Each partial translation of a combination translates at least one constraint of the user query that the others in the same combination do not.
10. 'WORDBOOK', 'HANDBOOK', 'DIRECTORY' and 'ENCYCLOPEDIA' are ignored although they lead to new constraints based on role 'CONTENT' because this role has no translation into Stanford-I.
11. 'Misc-publication' subsumes 'technical-manual' so 'technical-manual' is ignored.
12. If the DL system used lacks this feature, the terminological difference could be calculated with the help of its *subsumption mechanism* (see [6]).
13. The terminological difference is calculated between the extended definitions.

14. '(ATLEAST 1 ISBN)' and '(ATLEAST 1 PLACE-OF-PUBLICATION)' are constraints in the description of 'BOOK' with no translation into Stanford-I and they were ignored in all the plans; see example in Section 6.3.1.
15. This is the problem mentioned at the beginning of this section. The sentence makes no sense for the user since they are homonyms.
16. This extensional information will be retrieved, stored and updated periodically by the system.
17. The interontology relationships used in integration of the ontologies are semantic and not extensional relationships.
18. As we change in numerator and denominator we do not know which option is greater.
19. In DL systems they are called *defined terms*.
20. Calculation of loss is measured as a fraction but presented to the user as a percentage value.
21. If the higher bound is 1 or the lower bound is 0, no new information has been obtained.

References

1. E. Achilles, B. Hollunder, A. Laux, and J. Mohren. KRIS: Knowledge Representation and Inference System. Technical Report D-91-14, DFKI Kaiserslautern-Saarbrucken, 1991.
2. Y. Arens, C.A. Knoblock, and W. Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6(2-3):99–130, 1996.
3. J.M. Blanco, A. Illarramendi, and A. Goñi. Building a Federated Database System: An approach using a Knowledge Based System. *International Journal on Intelligent and Cooperative Information Systems*, 3(4):415–455, December 1994.
4. A. Borgida. From type systems to knowledge representation: Natural semantics specifications for description logics. *International Journal on Intelligent and Cooperative Information Systems*, 1(1), March 1992.
5. A. Borgida, R.J. Brachman, D.L. McGuinness, and L.A. Resnick. CLASSIC: A structural data model for objects. In *Proceedings ACM SIGMOD-89, Portland, Oregon*, 1989.
6. R. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9:171–216, Feb 1985.
7. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of 10th IPSJ conference, Tokyo, Japan*, October 1994.
8. C.J. van Rijsbergen. Information retrieval. http://dcs.glasgow.ac.uk/Keith/Chapter.7/Ch.7.html.
9. C. Collet, M. N. Huhns, and W. Shen. Resource integration using a large knowledge base in CARNOT. *IEEE Computer*, 24(12):55–62, December 1991.
10. Computing Research Laboratory (CRL) at New Mexico State University. http://crl.nmsu.edu/Research/Projects/mikro/index.html.
11. D. Dubois, J. Lang, and H. Prade. Automated Reasoning using Possibilistic Logic: Semantics, Belief Revision, and Variable Certainty Weights. *IEEE Transactions on Knowledge and Data Engineering*, 6(1), February 1994.
12. R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, 2nd edition, 1994.
13. V. Subrahmanian et al. Hermes: Heterogeneous reasoning and mediator system. http://www.cs.umd.edu/projects/hermes/overview/paper/index.html.
14. Excite. http://www.excite.com.
15. A. Goñi, J.M. Blanco, and A. Illarramendi. Connecting knowledge bases with databases: a complete mapping relation. In *Proc. of the 8th ERCIM Workshop. Trondheim, Norway*, 1995.
16. A. Goñi, E. Mena, and A. Illarramendi. Querying heterogeneous and distributed data repositories using ontologies. In *Proceedings of the 7th European-Japanese Conference on Information Modelling and Knowledge Bases (IMKB'97), Toulouse (France)*, May 1997.
17. T. Gruber. http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.
18. T. Gruber. Theory BIBLIOGRAPHIC-DATA, September 1994. http://www-ksl.stanford.edu/knowledge-sharing/ontologies/html/bibliographic-data/index.html.

19. J. Hammer, M. Breunig, H. Garcia-Molina, S. Nestorov, V. Vassalos, and R. Yerneni. Template-based wrappers in the tsimmis system. In *Proceedings of the Twenty-Sixth SIGMOD International Conference on Management of Data, Tucson, Arizona*, May 1997.
20. J. Hammer and D. McLeod. An approach to resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous, Database Systems. *International Journal on Intelligent and Cooperative Information Systems*, 2(1), March 1993.
21. V. Kashyap and A. Sheth. Semantic and Schematic Simililarities between Databases Objects: A Context-based approach. *The VLDB Journal*, 5(4), December 1996.
22. A.Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, 5(2):121–143, September 1995.
23. R. MacGregor. A deductive pattern matcher. In *Proceedings of AAAI-88, St. Paul, Minnesota*, August 1987.
24. E. Mena. http://siul02.si.ehu.es/~jirgbdat/OBSERVER/ontologies.html.
25. E. Mena, A. Illarramendi, and J.M. Blanco. Discovering relationships among ontologies describing data repositories contents. In *International Conference on Information, Systems, Analysis and Synthesis (ISAS'96), Orlando (Florida), USA*, July 1996.
26. E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth. Managing multiple information sources through ontologies: Relationship between vocabulary heterogeneity and loss of information. In *Proceedings of Knowledge Representation Meets Databases (KRDB'96), ECAI'96 conference, Budapest, Hungary, August 1996, pp. 50-52*, 1996.
27. E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth. Domain specific ontologies for semantic information brokering on the global information infrastructure. In *Proc. of the International Conference on Formal Ontologies in Information Systems (FOIS'98). Trento (Italy)*, June 1998.
28. E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. In *Proc. of the First IFCIS International Conference on Cooperative Information Systems (CoopIS'96), Brussels (Belgium), June*. IEEE Computer Society Press, 1996.
29. G. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11), November 1995.
30. A. Motro. Multiplex: A formal model of multidatabases and its implementations. Technical report, Technical Report ISSE-TR-95-103, Department of Information and Software Systems Engineering, George Mason University, Fairfax, Virginia, March 1995.
31. M. Papazoglou and S. Milliner. Subject-based organization of the information space in multi-database networks. In *International Conference on Advanced Information Systems Engineering (CaiSe), Pisa, Italy*, June 1998.
32. R. Bayardo, W. Bohrer, R. Brice, A. Cichocki, G. Fowler, A. Helai, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. Infosleuth: Agent-based semantic integration of information in open and dynamic environments. In *Proceedings of the 1997 ACM International Conference on the Management of Data (SIGMOD), Tucson, Arizona.*, May 1997.
33. G. Salton. *Automatic text processing.* Addison-Wesley, 1989.
34. P.H. Speel. *Selecting Knowledge Representation Systems.* PhD thesis, University of Twente, Enschede, the Netherlands, 1995.
35. A. Tomasic, L. Raschid, and P. Valduriez. Scaling heterogeneous distributed databases and the design of disco. In *Proceedings of the 16th International Conference on Distributed Computing Systems, Hong Kong*, 1995.
36. Pauray S. M. Tsai and Arbee L. P. Chen. Querying Uncertain Data in Heterogeneous Databases. In *Third International Workshop on Research Issues in Data Engineering: Interoperability in Multidatabase Systems, Vienna, Austria*, April 1993.
37. K. von Luck, B. Nebel, C. Peltason, and A. Schmiedel. The anatomy of the BACK system. Technical Report KIT Report 41, Technical University of Berlin, Berlin, F.R.G., 1987.
38. VRML Architecture Group. http://kether.vrml.org/vrml10c.html.

Contributing Authors

Eduardo Mena Eduardo Mena is an Assistant Professor at the University of Zaragoza. He got his B.S. in Computer Science from the University of the Basque Country and he is a PhD. student at the University of Zaragoza. For a year he was a visiting researcher in the Large Scale Distributed Information Systems Laboraotry at the University of Georgia. His research interest areas include interoperable and heterogeneous data systems, Internet computing and mobile computing. His main contribution is the OBSERVER project. He has several publications related to that project in international journals and conferences. He has also served as Program Commitee of the Intelligent Information Integration workshop.

Arantza Illarramendi Dr. Arantza Illarramendi is a professor of Computer science in the Department of Languages and Systems Information at the University of the Basque Country, Spain, where she heads the Interoperable Database Group. She has been a visiting researcher at the Toronto University. Her main research interest include interoperable and heterogeneous data systems and mobile computing. She has published several papers in refereed journals and in the proceedings of international conferences and workshops. She has also served as Program Committee member of international conferences/workshops.

Vipul Kashyap Dr. Vipul Kashyap is a Research Scientist at Bellcore's Applied Research Laboratory at Morristown, NJ. He has earlier done research at R&D Labs such as MCC and the Large Scale Distributed Information Systems Laboratory at University of Georgia. He has done his Ph.D. in Computer Science from Rutgers University. His research interests are Information Brokering over heterogeneous digital data. His work involves developing and evaluating architectures for information brokering over the global information infrastructure. He is conducting research into domain specific techniques for capturing information content and the use of real world pre-existing domain ontologies for describing multimedia data. He is also looking into issues of interoperation across ontologies describing different information domains. His works have been published in prestigious journals and conferences and he is writing a book on metadata based information brokering. He has participated in panels and served on the Program Committee of prestigious conferences.

Amit P. Sheth Dr. Amit Sheth directs the Large Scale Distributed Information Systems (LSDIS) Lab (http://lsdis.cs.uga.edu), is an Associate Professor of Computer Science at the University of Georgia, and the President of Infocosm, Inc (http://infocosm.com). Earlier he worked in the R&D labs at Bellcore, Unisys, and Honeywell. His current research interests include multiparadigm transactional workflow (project METEOR), management of heterogeneous digital data and semantic issues in global information systems (projects InfoQuilt and VisualHarness), telemedicine/teleconsulting, and electronic commerce. Besides numerous publications, his research has lead to two commercial products (InfoHarness and METEOR). He is the steering committee chair of the Joint ACM Conference Work Activity Coordination and Collaboration, a founding member and executive committee member Intl. Foundation on Cooperative Information Systems, and has served as Program/General chair of four international conferences/workshops. He has edited books on metadata for digital media, workflow management, and multidatabase systems.