UNIVERSITY OF CRETE
DEPARTMENT OF COMPUTER SCIENCE

Collaborative Ontology-based Information Indexing and Retrieval

Yannis Tzitzikas

Doctoral Dissertation

Heraklion, June 2002

UNIVERSITY OF CRETE

DEPARTMENT OF COMPUTER SCIENCE

Collaborative Ontology-based Information Indexing and Retrieval

Dissertation submitted by

Yannis Tzitzikas

in partial fulfillment of the requirements for

the PhD degree in Computer Science

Author:

Yannis Tzitzikas, Department of Computer Science

Panos Constantopoulos, Professor, Dept. of Computer Science, University of Crete, Advisor

---

Nicolas Spyratos, Professor, Dept. of Computer Science, Universite de Paris-Sud

---

Dimitris Plexousakis, Assistant Professor, Dept. of Computer Science, University of Crete

---

Vassilis Christophides, Assistant Professor, Dept. of Computer Science, University of Crete

---

Manolis Koubarakis, Associate Professor, Dept. of Electronic and Computer Engineering, Technical University of Crete

---

Yannis Ioannidis, Professor, Dept. of Informatics, National and Kapodistrian University of Athens

---

Timos Sellis, Professor, Dept. of Electrical and Computer Engineering, National Technical University of Athens

Approved by:

Panos Constantopoulos

Chairman of Graduate Studies

Heraklion, June 2002

Abstract

An information system like the Web is a continuously evolving system consisting of multiple heterogeneous information sources, covering a wide domain of discourse, and a huge number of users (human or software) with diverse characteristics and needs, that *produce* and *consume* information. The challenge nowadays is to build a scalable information infrastructure enabling the effective, accurate, content-based retrieval of information, in a way that adapts to the characteristics and interests of the users.

The aim of this work is to propose formally sound methods for building such an information network based on ontologies which are widely used and are easy to grasp by ordinary Web users.

The main results of this work are:

- A novel scheme for indexing and retrieving objects according to multiple aspects or *facets*. The proposed scheme is a faceted scheme enriched with a method for specifying the combinations of terms that are *valid*. We give a model-theoretic interpretation to this model and we provide mechanisms for *inferring* the *valid combinations* of terms. This inference service can be exploited for preventing errors during the indexing process, which is very important especially in the case where the indexing is done collaboratively by many users, and for deriving "complete" navigation trees suitable for browsing through the Web. The proposed scheme has several advantages over the hierarchical classification schemes currently employed by Web catalogs, namely, *conceptual clarity* (it is easier to understand), *compactness* (it takes less space), and *scalability* (the update operations can be formulated more easily and be performed more efficiently).
- A flexible and efficient model for building *mediators* over ontology-based information sources. The proposed mediators support several modes of *query translation* and evaluation which can accommodate various application needs and *levels of answer quality*. The proposed model can be used for providing users with customized *views* of Web catalogs. It can also complement the techniques for building mediators over relational sources

so as to support *approximate translation* of partially ordered domain values.

- A data-driven method for *articulating* ontologies. This method can be used for the automatic, or semi-automatic, construction of an articulation between two or more *materialized* ontologies. A distinctive feature of this method is that it is *independent* of the nature of the objects, i.e. the objects may be images, audio, video, etc, and that it can be implemented efficiently by a *communication protocol*, thus the involved ontologies (sources) can be *distant*.
- A novel method for *fusing* the results of sources which return ordered sets of objects. The proposed method is based solely on the actual results which are returned by each source for each query. The final (fused) ordering of objects is derived by aggregating the orderings of each source by a *voting* process. In addition, the fused ordering is accompanied by a *level of agreement* (alternatively construed as the level of confidence). The proposed method does not require any prior knowledge about the underlying sources, therefore it is appropriate for environments where the underlying sources are heterogeneous and autonomous, thus it is appropriate for the Web, for example, for building mediators over search engines.

These results allow creating a complex *network of sources* consisting of primary and secondary sources, where a primary source can be ontology-based, retrieval or hybrid (i.e. a source that is both ontology-based and retrieval).

Supervisor:
Panos Constanpoulos,
Professor of Computer Science
University of Crete

Some inspiring words..

There are four sorts of men:
He who knows not and knows not he knows: he is a fool; *shun him*
He who knows not and knows he knows not: he is simple; *teach him*
He who knows and knows not he knows: he is asleep; *wake him*
He who knows and knows he knows: he is wise; *follow him*

Arabian proverb

"Never regard your study as a duty, but as the enviable opportunity to learn to know the liberating influence of beauty in the realm of the spirit for your own personal joy and to the profit of the community to which your later work belongs."

"Everything should be made as simple as possible, but not simpler."

"If we knew what it was we were doing, it would not be called research, would it?"

Albert Einstein

to my family

Acknowledgments

First of all I would like to thank my advisor, Professor Panos Constantopoulos, for supervising and supporting my research efforts, both at MSc and the PhD level.

I am really very grateful to my advisor and now close friend Professor Nicolas Spyratos, for giving me the opportunity to work with him, for the endless discussions which often escape the narrow framework of this thesis and was touching almost all aspects of life. He has a remarkably clear way of thinking and his is full of energy. I will never forget him, he is a professor of life! I wish to him and his family the very best.

I would also like to thank the members of my dissertation committee from the University of Crete and ICS-FORTH, Professors Dimitris Plexousakis, and Vassilis Christophides as well as the external members, Professors Manolis Koubarakis, Yannis Ioannidis, and Timos Sellis for their helpful comments and questions.

Many thanks to all former and current members of the Information Systems Laboratory at the Institute of Computer Science (ICS-FORTH), and particular to Vassilis Christophides (for many fruitful discussions and for providing me with fresh bibliographic references), Manos Theodorakis (for helping me prepare the slide presentation) and Anastasia Analyti (for her willingness to help me).

I would like to thank the secretaries of the Department of Computer Science, especially Rena Kalaitzakh, and of the Institute of Computer Science (ICS-FORTH).

I would like to acknowledge the support, both financial and in facilities, by the Institute of Computer Science (ICS-FORTH). I also want to thank the University of Crete for granting me the Maria Michael Manasaki legacy's fellowship.

This dissertation would not have been possible without the support and encouragement from my parents, Titos and Stella, and my brother Nikos. Last by certainly not least, I would like to thank my beloved Tonia Dellaporta for her support, patience, and endurance to all the frustration and long working hours that have been part of this dissertation, and for being there

to inspire me and remind me of the many joys of life.

Finally, I wish to thank the following: Manos Theodorakis, Polivios Klimathianakis, George Georgiannakis and Athena Trapsioti (for being really good friends and for encouraging me at the first years of this work); Dimitris, Polykarpos, Katerina, Nikos, Carola (for all the good times we had together); U2 (and they know why); *and* Loulou (because she asked me to).

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

We are witnesses of a fundamental paradigm shift. Following the periods of the art of writing (2500 b.c.), the art of printing (1400) and now the art of communication (2000). The rapid growth and expansion of the Web the recent years, proves this realization. The Web is a continuously evolving system consisting of multiple heterogeneous information sources, covering a wide domain of discourse, and a huge number of users (human or software) with diverse characteristics and needs, who *produce* **and** *consume* information. Currently the Web consists of an estimated 1.5 billion HTML pages and 250 millions users. In addition, large volumes of volatile, redundant, semi- or unstructured heterogeneous data are available in connected legacy databases, file systems, multimedia database systems and software applications. This includes, for example, bibliographic entries, images, speech, text, and video data.

In general, users have two ways to find the pages they are looking for: they can *search* and they can *browse*. The various search engines index large numbers of pages and allow users to enter keywords and retrieve pages that contain those keywords. Browsing is usually done by clicking through a hierarchy of subject terms (connected in a *taxonomy* or *ontology*) until the area of interest has been reached. The corresponding node then provides the user with links to related pages.

Every Web user knows how time consuming and error-prone it is to locate specific information. The main problem is that there is too much information available, and that keywords are rarely an appropriate means for locating the information in which a user is interested. It has been recognized long ago [113] that structured and controlled vocabularies (taxonomies or *ontologies*)

promise more effective and efficient information retrieval (ensuring indexing consistency and supporting reasoning). Indeed, Web catalogs (like Yahoo![1] and Open Directory[2]) turn out to be very useful for browsing and querying. Although they index only a fraction of the pages that are indexed by search engines they are hand-crafted by domain experts and are therefore of high quality.

However, these catalogs have three important drawbacks: First, their vocabularies are very big. For example, the hierarchy of Yahoo! consists of 20 thousands terms, while the hierarchy of Open Directory consists of 300 thousands terms! Due to their size these catalogs exhibit *inconsistencies* in the structure of their terminologies. Moreover, the indexing of the pages, which is done by humans, is a *laborious* task often resulting in incomplete or inconsistent object indexing. Second, it is unlikely that the 250 million Web users have knowledge backgrounds and interests similar enough, so that one taxonomy fits all needs. Third, users *cannot* update the taxonomy or the indexing of the objects so as to customize them to their interests and needs.

It would be very useful if users could use their *own* terminology and taxonomy in order to *access*, *search/query* and *index* the pages of the Web.

The general objective of this dissertation is to study the arising issues and to propose formally sound methods and tools for building such an information network. The dissertation does not focus on a particular application (or implementation), nor on the Web in particular.

1.2 Background - The Thesis of this Dissertation

Information retrieval deals with the representation, storage, organization of, and access to information items. The representation and organization of the information items should provide the user with easy access to the information in which he/she is interested.

Every information retrieval system can be described as consisting of a set of information items, or *objects* (Obj), a set of requests (Q), and some mechanism (Sim), for determining which, if any, of the objects meet the requirements of, or is relevant to, the requests. Sim actually represents an operator that maps specific queries to particular objects included in the stored object set. Figure 1.1 shows the relationship of these components.

In practice, the relevance of specific objects to particular requests is not determined directly. Rather the objects are first converted to a specific form using a classification or *indexing language*

[1] http://www.yahoo.com
[2] http://dmoz.org

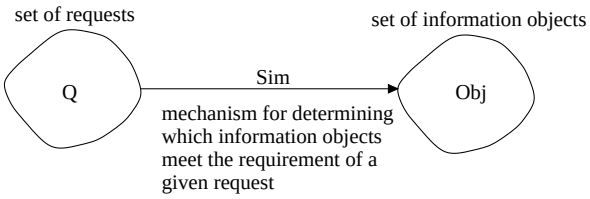


Figure 1.1: Information retrieval system environment

(L). The requests are also converted into a representation consisting of elements of this language. Figure 1.2 [3] exhibits the processing of the objects and the requests into L.

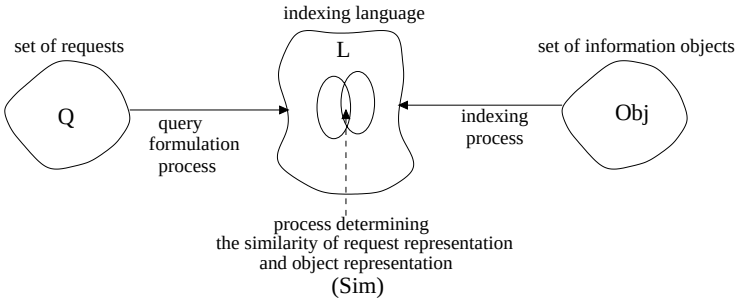


Figure 1.2: Functional overview of information retrieval

The mapping of the objects to the indexing language is known as the *indexing process* and it has three primary purposes: (a) to permit the easy location of objects by topic, (b) to define topic areas, and hence relate one object to another, and (c) to predict the relevance of a given object to a specified information need. The mapping of the information requests to the indexing language is known as the *query formulation process*. The procedures for determining which objects should be retrieved in response to a query are based on representations of the requests and objects consisting of elements of the indexing language. In some cases, the system places the retrieved objects in order of probable relevance to the request.

The set of indexing terms may be *controlled*, that is, limited to a predefined set of index terms, or *uncontrolled*, that is, allowing use of any term that fits some broad criteria.

Indexing may be carried out either *manually* or *automatically*, or by a combination of the two processes. Automatic indexing is usually applicable in the case where the objects are texts and is based on text analysis (e.g. see [113]). Manual indexing involves some intellectual effort to identify and describe the content of an object. If the indexing language is controlled, then the indexing of the objects is usually done manually. However there are techniques which allow the

[3] This Figure is taken from [113].

automatic indexing of objects under a controlled vocabulary (e.g. see [105, 88, 19, 143, 84]).

There are several models for computing the similarity, or *relevance*, between the user's query and an object. For instance, in the case where the objects are texts and the indexing language is uncontrolled, we have the boolean model, the vector space model [114], the probabilistic model [108], the inference network model [125], the belief network model [107], etc. In fact, relevance is not a formally and clearly defined notion; what relevance is, in other words, is defined by the user from time to time and from experiment to experiment, and is then heavily dependent on judgments where highly subjective and hardly reproducible factors are brought to bear.

The evaluation of an information retrieval system commonly concerns retrieval *effectiveness* and *efficiency*. Retrieval efficiency concerns issues like the user effort and the time and cost needed for retrieving the desired information objects. The evaluation of retrieval effectiveness is based on a test reference collection and on an evaluation measure. The test reference collection consists of a collection of documents, a set of example requests, and a set of relevant documents (provided by specialists) for each example request. Given a system S, the evaluation measure quantifies (for each example request) the similarity between the set of documents retrieved by S and the set of relevant documents provided by the specialists. This provides an estimation of the goodness of the system S. The most widely used evaluation measures are the *recall* and *precision*. Let R be the set of relevant documents of a given information request and assume that the system which is being evaluated processes the information request and generates a document answer set A. The recall and precision are defined as follows

$$Recall = \frac{|A \cap R|}{|R|} \text{ and } Precision = \frac{|A \cap R|}{|A|}$$

Certainly, the effectiveness and efficiency of information retrieval is in a significant degree determined by the indexing language employed. Uncontrolled indexing languages introduce many opportunities for ambiguity and error, while controlled vocabularies permit the control of spelling and elimination of synonyms. Controlled indexing languages may be structured (e.g. thesauri, semantic network-based ontologies), capturing an adequate body of real-world (domain) knowledge. This knowledge is exploited (through a form of reasoning) for improving the effectiveness of retrieval. The adoption of thesauri has already been recognized to improve the effectiveness of retrieval ([100], [83], [85], [57]) and assist the query formulation process by expanding queries with synonyms, hyponyms and related terms.

However, controlled indexing languages guarantee retrieval of appropriately marked items only when the correct search terms are known. Moreover, the gains of controlled indexing languages are dependent on the quality, accuracy and consistency of the indexing process: the indexers

must be aware of the available terms and must classify similar information items to comparable indexing entries. In practice, accuracy and consistency are difficult to maintain.

The controlled indexing languages that are used for information retrieval usually consist of a set of terms structured by a small number of relations such as subsumption and equivalence. However, the indexing of the objects can also be done (especially in the case of a manual indexing process) with respect to more expressive conceptual models representing domain knowledge in a more detailed and precise manner. Such conceptual models can be represented using logic-based languages, and the corresponding reasoning mechanisms can be exploited for retrieving objects. There are several works that take this conceptual modeling and reasoning approach to information retrieval (e.g. relevance terminological logics [86], four-valued logics [110]). This conceptual modeling approach is useful and effective if the domain is narrow. If the domain is too wide (e.g. the set of all Web pages) then the problem is that it is hard to conceptualize the domain; actually there are many different ways to conceptualize it, meaning that it is hard to reach a conceptual model of wide acceptance. For this purpose, even today, ontologies that have simple structure are usually employed for retrieving objects from large collections of objects ([112]).

As mentioned earlier, the indexing of objects under a controlled and structured vocabulary is usually done manually. By consequence, if the collection of objects is big, this task is laborious and costly. If the scope of the intended application is small (e.g. within a small enterprise) then this cost is forbidding, therefore, automatic indexing techniques are usually preferred in such applications.

However, if the scope of the intended application is large (hence the expected profits from good indexing are big), then manual indexing can be carried out. Yahoo! is an example of this case. Although the objective of Yahoo! is to index a huge collection of objects (the set of all Web pages), the indexing of the pages is done mainly manually by a team of 20 indexers. The indexing cost is paid off by the profits coming from the numerous clients of Yahoo!: probably every Web user has accessed, at least once, the catalog of Yahoo!.

The manual indexing of the pages of the Web is not actually an unrealistic assumption because there are many users (250 millions) all having access to the *same* set of pages (1.5 billion pages). This means that the indexing cost can be divided. The case of Open Directory supports this claim. Open Directory is probably the biggest and most popular catalog of the Web. It is used daily by thousands of users. In addition, this catalog is exploited by several search engines

such as Google[4], Netscape[5], Lycos[6] and HotBot[7]. The indexing of the pages in Open Directory is done manually by 20 thousands volunteer editors. This approach, although collaborative, results to a system that has some important drawbacks: First its vocabulary is very big (300 thousands terms). Due to this size the catalog exhibits inconsistencies in the structure of its terminology. Moreover, the indexing of the pages is a laborious task often resulting in incomplete or inconsistent object indexing. In addition, it is unlikely that the 250 million Web users are similar enough in their knowledge background and interests so that one taxonomy fits all needs. Furthermore, the users cannot update the taxonomy or the indexing of the objects. A volunteer editor can only update a specific part of the catalog, the part of the catalog which concerns a topic on which the editor has declared himself an expert.

This functionality is very far from the functionality that we envision. We would like each user to be able to use a terminology that is familiar to him, structured according to his preferences, in order to locate the useful information on the Web. In addition, we would like each user to be able to use this terminology in order to index, and store in a "personal" database, the objects that are of interest to him. The objects in this database should also be accessible by other users using their own vocabularies.

We envisage a *network* of sources. The sources may be primary or secondary. Roughly, we can distinguish three kinds of primary sources: *ontology-based*, *retrieval* and *hybrid*.

- *Ontology-based* sources.

  We use the term ontology-based source to refer to a source which indexes the objects of interest using terms taken from a *controlled*, possibly *structured*, vocabulary. These sources accept queries expressed over their vocabulary. In the environment of the Web, general purpose catalogs, such as Yahoo! or Open Directory, as well as domain specific catalogs/gateways (e.g. for medicine, physics, tourism), can be considered as examples of such sources.

- *Retrieval* sources.

  We use the term retrieval source to refer to a source that indexes the objects of interest using an *uncontrolled* vocabulary. These sources usually accept natural language queries and return *ordered sets* of objects. Text retrieval systems (the typical case of "Information Retrieval systems"), as well as the search engines of the Web, fall into this category.

[4] http://www.google.com
[5] http://www.netscape.com
[6] http://www.lycos.com
[7] http://www.hotbot.com

- *Hybrid* sources.

  We use the term hybrid source to refer to a source that is both ontology-based and retrieval source. A hybrid source accepts *two* kinds of queries: queries over a controlled vocabulary and natural language queries. A source whose functionality moves towards this direction is Google. Using Google, one can first select a category, e.g. `Sciences/CS/DataStructures`, from the ontology of Open Directory and then submit a natural language query, e.g. "`Tree`". The search engine will compute the degree of relevance with respect to the natural language query, "`Tree`", only of those pages that fall in the category `Sciences/CS/DataStructures` in the catalog of Open Directory. Clearly, this enhances the precision of the retrieval and is computationally more economical.

Figure 1.3 sketches graphically each kind of source.

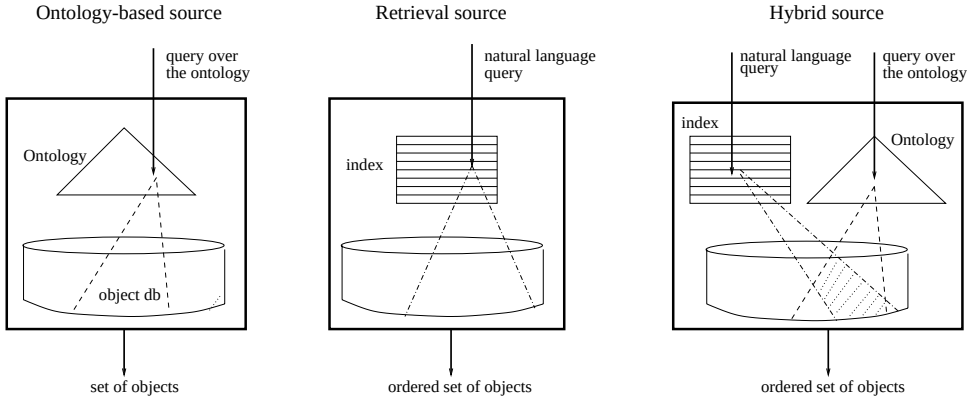


Figure 1.3: Kinds of sources

The network that we envision should also have secondary sources, or *mediators*, which integrate and provide a unified access to several other sources which may be primary or secondary. Figure 1.4 shows an example of a network of this kind, for the environment of the Web.

The contribution of this thesis towards the realization of this network lies in

- A model describing formally the *ontology-based sources* and their functionality. An essential feature of these sources is that they can provide two types of answer to a given query, namely, a *sure* answer or a *possible* answer. The first type of answer is appropriate for users that focus on precision, while the second for users that focus on recall. Ontology-based sources incorporate and extend the functionality of current Web catalogs.
- The introduction of *extended faceted ontologies*, a novel scheme for indexing and retrieving

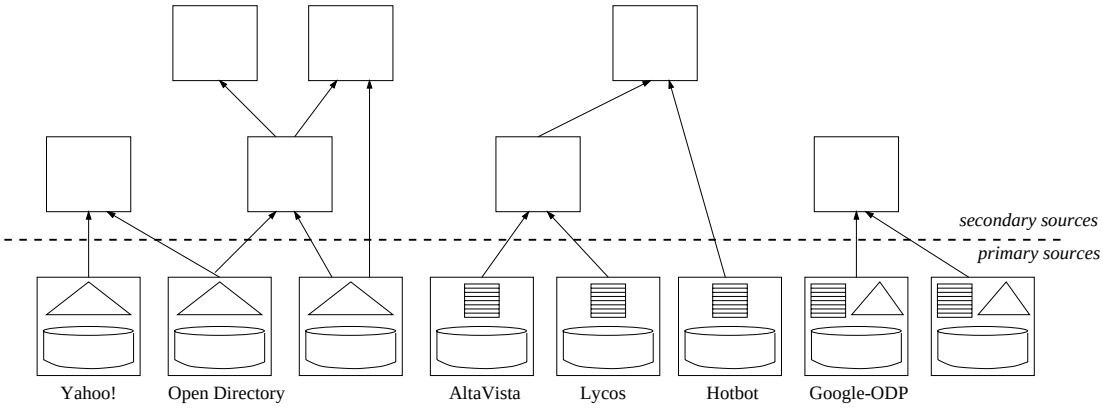


Figure 1.4: A network of Web sources

objects according to multiple aspects, or *facets*, which has several advantages by comparison to the hierarchical taxonomies (hierarchical classification schemes) that are currently employed by Web catalogs, namely:

- *conceptual clarity* (they are easier to understand),
- *compactness* (they take less space), and
- *scalability* (the update operations can be formulated more easily and be performed more efficiently).

A faceted scheme is proposed, the originality of which lies in a mechanism for specifying and inferring valid combinations of terms. This inference mechanism can be exploited for:

- preventing errors during the indexing process, which is especially important in case where indexing is performed collaboratively by many users, and
- for *dynamically* deriving "complete" hierarchical taxonomies which are suitable for browsing through the Web.

This scheme can aid the development of enhanced ontology-based sources. Parts of this work have been published in [128], [129] and [134].

- A flexible and efficient model for building *mediators* over ontology-based information sources. The model is based on *articulations*, i.e. on relations that bridge the gap between different ontologies. Ontology articulation has many advantages compared to ontology merging, because merging would introduce storage and performance overheads, as the ontologies employed by Web catalogs contain very large numbers of terms. In addition, full merging is a laborious task which in many cases does not pay-off because the integrated

ontology becomes obsolete when the ontologies involved change. Another problem with full merging is that it usually requires full consistency, which may be hard to achieve in practice, while articulation can work on locally consistent parts of the ontologies involved.

The proposed mediators support several modes of *query translation* and evaluation which can accommodate various application needs and *levels of answer quality*. The proposed model can be used for providing users with customized views of Web catalogs. It can also complement the techniques for building mediators over relational sources so as to support *approximate translation* of partially ordered values. Parts of this work have been published in [130], [133] and [132].

- A data-driven method for *articulating* ontologies. Finding semantic mappings among ontologies is a crucial need in many application areas and it is also a key challenge in building the *Semantic Web* [13]. Given the de-centralized nature of development of the Semantic Web, there will be an explosion in the number of ontologies (see for example [26]). Many of these ontologies will describe similar domains, but using different terminologies, and others will have overlapping domains. To integrate data from disparate ontologies, we must know the semantic correspondences, or *articulations*, between them. The proposed method can be used for the automatic, or semi-automatic, construction of an articulation between the ontologies of two or more sources. The method is based on the objects that are indexed under both ontologies and a distinctive feature of this method is that: (a) it is independent of the nature of the objects, i.e. the objects may be images, audio, video, etc., and (b) it can be implemented efficiently by a communication protocol, thus the involved sources can be *distant*.

- A novel method for *fusing* the results of sources that return ordered sets of objects. The proposed method can be used for building mediators over retrieval sources. It is based solely on the actual results which are returned by each source for each query. The final (fused) ordering of objects is derived by aggregating the orderings of each source by a voting process. In this way the objects of the fused answer are ordered according to a measure of aggregate relevance. Since the proposed method does not require any prior knowledge about the underlying sources, it is appropriate for environments where the underlying sources are heterogeneous and autonomous, thus it is appropriate for the Web. Furthermore, the proposed method can be exploited for building mediators over ontology-based and hybrid sources. Parts of this work have been published in [127].

1.3 Outline of this Thesis

This thesis consists of five parts. Each part has its own role and contribution. In addition, these parts in conjunction drive towards the network of sources that we envisage. Each of the five parts is described in a separate chapter. The review of the bibliography and the novelty/contribution of each part is given in the corresponding chapter. Specifically, the remaining of this thesis is organized as follows:

- Chapter 2 describes the *ontology-based sources* and their functionality.
- Chapter 3 introduces the *extended faceted ontologies*.
- Chapter 4 describes our model for building *mediators* over ontology-based sources.
- Chapter 5 presents a data-driven method for *articulating* ontologies.
- Chapter 6 describes our method for *fusing* the results of retrieval sources.
- Chapter 7 concludes the thesis and discusses further research.

Figure 1.5 illustrates the overall structure of this thesis and the dependencies between chapters. An arrow from Chapter I to Chapter J means that that J depends on I. The broken arrows indicate a weak dependency.

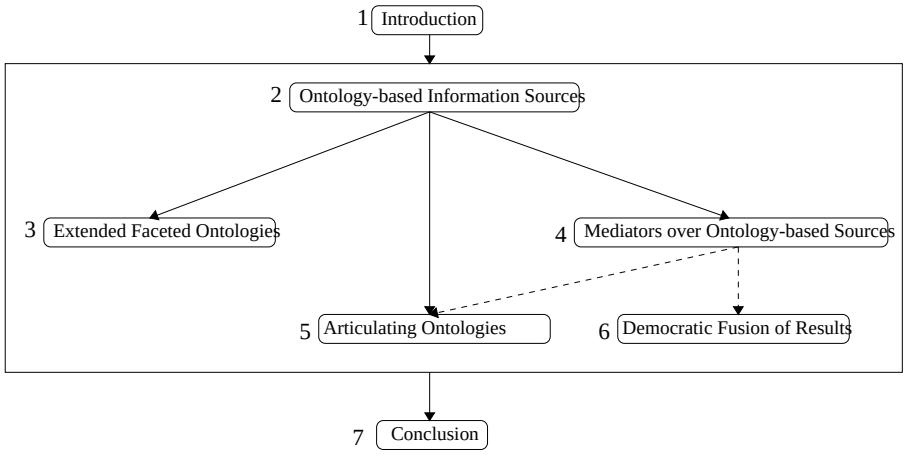


Figure 1.5: The structure and the chapter dependencies of the thesis

Chapter 2

Ontology-based Information Sources

This chapter presents what we call *ontology-based sources*. We consider a domain consisting of a denumerable set of objects, for example, in the environment of the Web, the domain could be the set of all Web pages, specifically, the set of all pointers to Web pages. An ontology-based source consists of an *ontology*, i.e. a set of names, or *terms*, structured by a *subsumption* relation, and a *database* storing objects that are of interest to its users. Specifically, each object in the database of a source is indexed under one or more terms of the ontology of that source. In quest for objects of interest, a user can browse the source ontology until he reaches the desired terms, or he can query the source by submitting a boolean expression of terms. The source will then return the appropriate set of objects. Moreover, each object in the answer is accompanied by its description.

In the environment of the Web, personal bookmarks, general purpose catalogs, such as Yahoo! or Open Directory, and domain specific catalogs/gateways (e.g. for medicine, physics, tourism) can be considered as examples of such sources. These catalogs turn out to be very useful for browsing and querying. Although they index only a fraction of the pages that are indexed by search engines using statistical methods (e.g. AltaVista[1], Google[2]), they are hand-crafted by domain experts and are therefore of high quality. Recently, also search engines start to exploit these catalogs in order to enhance the quality of retrieval. Specifically, the search engines now employ catalogs for achieving "better" degrees of relevance, and for determining (and presenting to the user) a set of relevant pages for each page in the answer set. In addition, some search engines (e.g. Google) now employ ontologies in order to enable limiting the scope (i.e. defining the context) of searches. According to the division of primary sources introduced in Section 1.2,

[1] www.altavista.com
[2] www.google.com

these can be regarded as *hybrid* sources.

An essential feature that distinguishes the sources that we will present is that they can provide two types of answer to a given query, namely, a *sure* answer or a *possible* answer (the first type of answer being appropriate for users that focus on precision, while the second for users that focus on recall). Moreover each object of the answer can be accompanied by its description, i.e. by all terms under which the object is indexed. These descriptions can aid the user (a) in selecting the objects of the answer that are most relevant to his information need, and (b) in getting acquainted with the ontology of the source.

The remaining of this chapter is organized as follows: Section 2.1 reviews ontologies and Section 2.2 describes the ontologies considered in this thesis. Section 2.3 defines formally the ontology-based sources. Section 2.4 discusses the query answering process and Section 2.5 enhances answers with object descriptions. Section 2.6 discusses storage issues and Section 2.7 discusses query evaluation. Section 2.8 provides some basic inference mechanisms which enable the reasoning over the ontology of a source. Finally, Section 2.9 summarizes and concludes the chapter.

2.1 Brief Review Of Ontologies

Research on ontologies is becoming increasingly widespread in the computer science community and its importance is being recognized in many diverse research fields and application areas, such as conceptual analysis, conceptual modeling, information integration, agent communication, semantic annotation (see [54], [55] for a review).

Ontology is a branch of Philosophy, which gives a systematic account of existence, i.e. a system of categories for a certain vision of the world (e.g. Aristotle's ontology [1]). These ontologies not only tell us what exists, according to a certain philosophical viewpoint, but also how we would classify and describe those things and can thus be thought of as an intellectual "lens" through which to view reality. Recent work by [95] suggests that conceptual modelling languages have an inherent ontology which is assumed by the modeling constructs provided by the language. Specifically, this work tries to characterize the conceptual modeling languages according to four ontologies, namely *Static*, *Dynamic*, *Intentional* and *Social*. Static ontologies describe what things exist, their attributes and interrelationships. Dynamic ontologies encompasses dynamic aspects of an application in terms of states, state transitions and processes. Intentional ontologies encompasses the world of agents, and things agents believe in, want, prove or disprove and argue

about. Such ontologies include concepts such as agent, issue, goal, supports, denies, subgoalOf, etc. Social ontologies covers social settings, permanent organizational structures or shifting networks of alliances and inter-dependencies. They are characterized in terms of concepts such as actor, position, role, authority, commitment, etc.

A similar work that tries to characterize the conceptual modeling languages according to categories of Chisholm's Ontology, shown in Figure 2.1, can be found in [89].

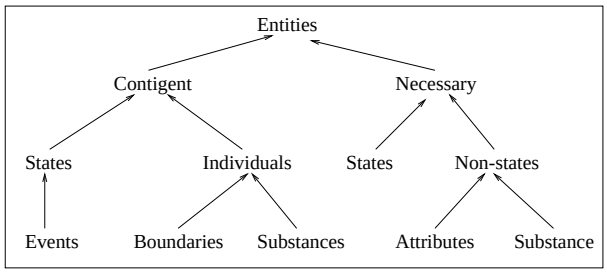


Figure 2.1: The categories of Chisholm's Ontology

The term ontology has also been employed in the field of *Language Engineering* which focuses on applications such as building natural language interfaces. The ontologies of this area are often called *lexical ontologies* or word-based thesauri. Examples of such ontologies include WordNet [29] and Roget's thesaurus. They consist of terms and relationships between terms, where terms denote lexical entries (words), while relations are intended as lexical or semantic relations between terms. For instance in WordNet [29], terms are grouped into equivalence classes, called synsets. Each synset is assigned to a lexical category i.e. noun, verb, adverb, adjective, and synsets are linked by hypernymy/hyponymy and antonymy relations. The former is the subsumption relation, while the latter links together opposite or mutually inverse terms such as tall/short, child/parent. Apart from this kind of thesauri, there are information retrieval-oriented thesauri like INSPEC [2], LCSH (Library of Congress Subject Headings [4]), and MeSH (Medical Subject Headings [5]). Each such thesaurus consists of a set of terms and a set of relations such as BT (Broader Term), NT (Narrow Term), UF (Used For), and RT (Related To) (for more see the standard [120]).

In *Artificial Intelligence* there are numerous definitions of what an ontology is, revolving around the basic idea that "an ontology is a consensual and formal specification of a vocabulary used to describe a specific domain" (see [54] for a review of the definitions that have been given). Below we present a few of them:

> An ontology is an explicit specification of a conceptualization [53].
>
> An ontology is considered as an engineering artifact constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the "intended meaning" of the vocabulary words [55]. These assumptions usually have the form of a first-order logical theory where vocabulary words appear as unary or binary predicate names, respectivelly called *concepts* and *relations*. In the simplest case an ontology describes a hierarchy of concepts related by subsumption relationships, while in more sophisticated cases, suitable axioms are added in order to express other relationships between concepts and to constrain their intended interpretation.
>
> Ontologies are consensual and formal specifications of a vocabulary used to describe a specific domain [33].

In general, an ontology specifies a conceptualization of a domain in terms of concepts, attributes, and relations. The *concepts* model the entities of interest in the domain. They are typically organized into a *taxonomy* where each node represents a concept and each concept is a specialization of its parent. Figure 2.2 shows a sample ontology for the Computer Science department domain. Each concept is associated with a set of *instances*. By the taxonomy's definition, the instances of a concept are also instances of an ancestor concept. For example, instances of `Student` are also instances of `People`. Each concept is associated with a set of *attributes*. For example the concept `Person` may have the attributes `name` and `birth`. An ontology may also define a set of *relations* among its concepts. For example a relation `AdvisedBy(Student, Professor)` might list all instance pairs of `Student` and `Professor` such that the former is advised by the latter. Furthermore, an ontology may also define a set of *axioms*. For example an axiom may state that the sets of instances of the concepts `Courses` and `People` have to be disjoint.

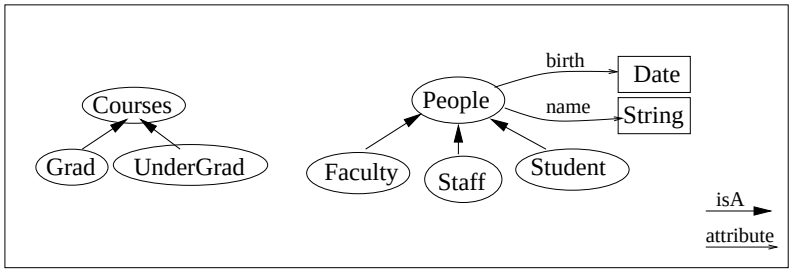


Figure 2.2: A sample ontology for the Computer Science department

Quite a number of ontologies have already been developed for structuring and reusing large bodies of knowledge (e.g. see CYC [75], KIF/Ontolingua [3]). These ontologies can be categorized according to various criteria (for example see [123], [55], [56]). For instance, according to [123] ontologies are distinguished to *theory ontologies* and *domain ontologies*. The former contain terms for concepts for representing some aspect of the world (e.g. time, space, causality, plans), while the latter contain terms for describing some domain (e.g. medicine, computer maintenance). Theory ontologies are usually abtract and small, while domain ontologies are usually very large containing thousands of terms. According to [55] ontologies are distinguished to *Top-Level*, *Domain*, *Task* and *Application*. Top-Level ontologies describe concepts which are independent of a particular problem or domain (e.g. space, time, matter, object, event, action, etc). Domain ontologies describe the vocabulary related to a generic domain (e.g. medicine). Their terms may specializate terms introduced in the top-level ontology. Task ontologies describe the vocabulary related to a generic task or activity (like diagnosing or selling). Their terms may specializate terms introduced in the top-level ontology. Finally, Application ontologies describe concepts depending both on a particular domain and task. These concepts often correspond to roles played by domain entities while performing a certain activity (e.g. replaceable unit, or space component).

In the *Web*, the term ontology commonly refers to the content-based organizational structures employed by site providers, in order to organize their contents and to provide browsing and retrieval services, i.e. Yahoo!'s subject hierarchy. Recently, more structured ontologies (including attributes and relations) are employed for meta-tagging ([79], [136]). Describing Web resources using formal knowledge is the essence of the next evolution step of the Web, termed the *Semantic Web* [13] and several formal languages to specify ontologies have been proposed, such as OIL, DAML+OIL, SHOE [79, 58] and RDF/S [18]. Though these languages differ in their terminologies and expresiveness, the ontologies that they model essentially share the same features (concepts, attributes and relations) we described above. For a list of ontologies for the Semantic Web see [80].

The term ontology is currently used to denote quite different things in different application settings. Several interesting research issues concern the applications of ontologies, for example, ontologies and *reuse* in conceptual modeling (e.g. see [53], [123], [75], [101], [141], [64]), ontologies and enhanced *information retrieval* (e.g. see [100], [83], [57] [85]), ontologies and information *integration* (e.g. see [52], [53], [55]), ontologies and the Semantic Web (e.g. see [94], [65]) ontologies and *agents* (e.g. see [69], [124] [79]), collaborative development and evolution of

ontologies (e.g. see [58]).

2.2 Ontologies in the Context of this Dissertation

The ontology structure we consider in this thesis is quite simple: ontologies consist of a set of terms structured by a subsumption relation. We do not consider attributes, relations or axioms. There are several reasons behind this choice:

- There are many ontologies of this kind since a lot of human resources have been allocated for developing this kind of structured vocabularies for several domains and disciplines.
- We wanted the ontology structure to be easy to grasp by ordinary Web users with minimal effort. Moreover, the queries submitted by ordinary users are bags of words and not structured queries.
- In a very broad domain such as the set of all Web pages, it is not easy to identify the classes of the domain because the domain is too wide and different users, or application needs, conceptualize it differently, e.g. one class of the conceptual model according to one user may correspond to a value of an attribute of a class of the conceptual model according to another user.

  For example, Figure 2.3 shows two different ways to conceptualize the same domain. Our example shows only two objects of the domain which are denoted by the natural numbers 1 and 2.

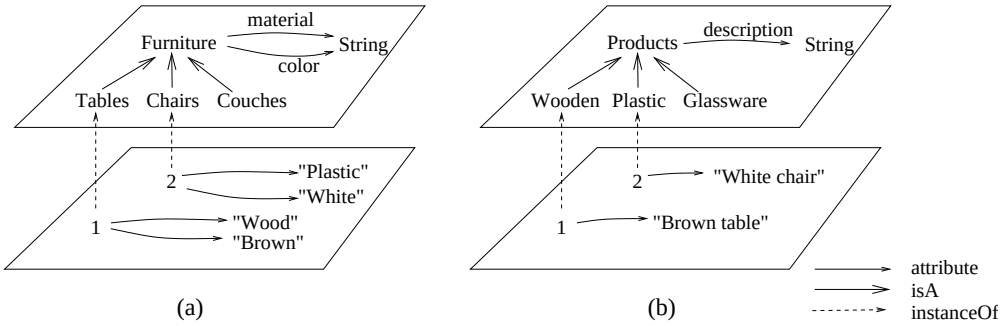


Figure 2.3: Two different ontologies (conceptual models) for the same domain

The conceptual model of Figure 2.3.(a) seems appropriate for building an information system for a store that sells furniture, while the conceptual model of Figure 2.3.(b) seems

appropriate for building an information system for a supermarket.

We can say that the classes of the ontology (a), i.e. the classes `Tables`, `Chairs` and `Couches`, have been defined in order to distinguish the objects of the domain according to their *use*. On the other hand, we can say that the classes of the ontology (b), i.e. the classes `Wooden`, `Plastic` and `Glassware`, have been defined in order to distinguish the objects of the domain according to their *material*. This kind of distinction is useful for a supermarket, as it determines (in a degree) the placement of the objects in the various departments of the supermarket.

Figure 2.4 shows an ontology for the same domain which consists of terms and subsumption links only. This ontology seems to be more application independent. All criteria (characteristics) for distinguishing the objects are equally "honoured".

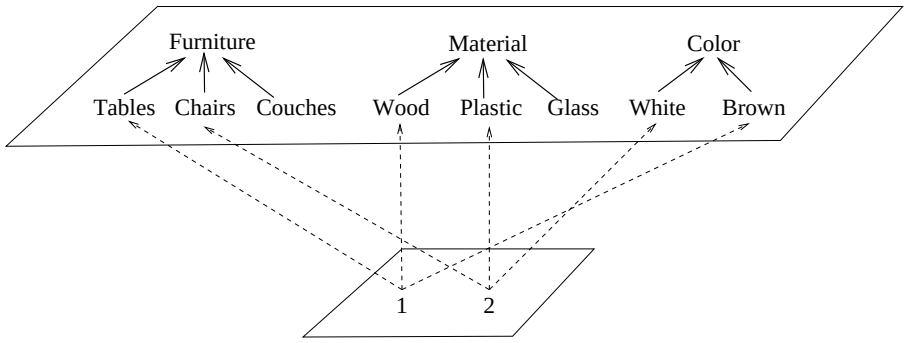


Figure 2.4: An ontology that consists of terms and subsumption links only

Specifically, we consider ontologies defined as follows:

Def 2.1 An *ontology* is a pair $(T, \preceq)$ where:

- T is a *terminology*, i.e. a set of names, or *terms*, and
- $\preceq$ is a *subsumption* relation over T, i.e. a reflexive and transitive relation over T.

If a and b are terms of T, we say that a is *subsumed* by b if $a \preceq b$; we also say that b *subsumes* a; for example, `Databases` $\preceq$ `Informatics`, `Canaries` $\preceq$ `Birds`. We say that two terms a and b are *equivalent*, and write $a \sim b$, if both $a \preceq b$ and $b \preceq a$ hold, e.g., `Computer Science` $\sim$ `Informatics`. Note that the subsumption relation is a preorder over T and that $\sim$ is an equivalence relation over the terms T. Moreover $\preceq$ is a partial order over the equivalence classes of terms, i.e. a reflexive, transitive and anti-symmetric relation over the set of equivalence classes.

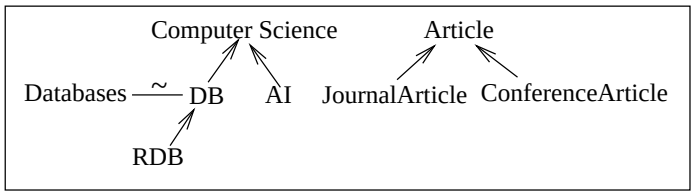


Figure 2.5: Graphical representation of an ontology

Figure 2.5 shows graphically an example of an ontology. Subsumption of terms is indicated by a continuous-line arrow from the subsumed term to the subsuming term. For example, the term `RDB` in Figure 2.5 is subsumed by `DB` as there is a continous-line arrow going from `RDB` to `DB`; this arrow indicates that $\texttt{RDB} \preceq \texttt{DB}$.

Note that we do not represent the entire subsumption relation but a subset of it sufficient to generate the entire relation. In particular, we do not represent the reflexive nor the transitive arrows of the subsumption relation.

Equivalence of terms is indicated by a continuous non-oriented line connecting the terms that are equivalent. For example, the term `Databases` is equivalent with the term `DB` since these two terms are connected by a continuous non-oriented line. Note that equivalence captures the notion of synonymy, and that each equivalence class simply contains alternative terms for naming a given set of objects.

For technical reasons that will become clear shortly, we assume that every terminology T contains two special terms, the *top term*, denoted by $\top$, and the *bottom term*, denoted by $\bot$. The top term subsumes every other term t, i.e. $t \preceq \top$. The bottom term is strictly subsumed[3] by every other term t different than top and bottom, i.e. $\bot \prec t$, for every t such that $t \neq \top$ and $t \neq \bot$.

The above definition captures ontologies of various kinds, e.g.:

- Ontologies which conceptualize the domain as a denumerable set of objects, and consist of terms that denote sets of objects and relationships that denote extensional subsumption[4]. The subject hierarchies of Web catalogs fit in this case. For example, Figure 2.8 shows the subject hierarchy of a catalog that provides access to hotel home pages according to the *location* of the hotels and the *sport facilities* they offer.

- Ontologies which consist of terms that denote geographical areas structured by a spatial

[3] A term a is strictly subsumed by a term b, denoted $a \prec b$, if $a \preceq b$ and $b \not\preceq a$.

[4] In contrast to the intensional meaning of terms (i.e. see [68], [20]) and to intensional subsumption.

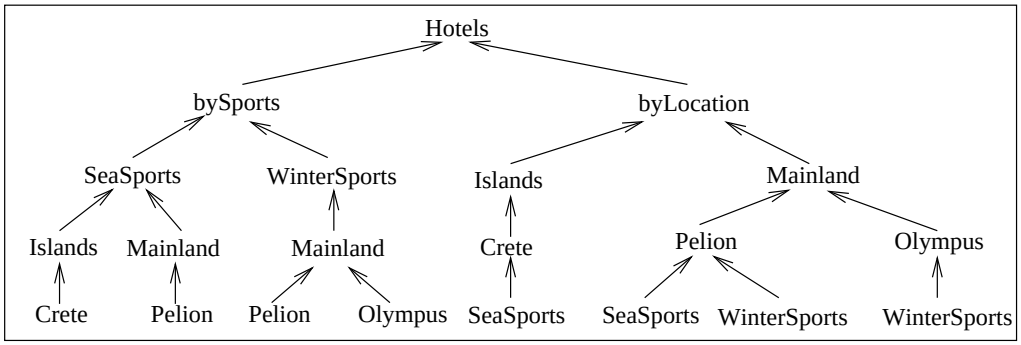


Figure 2.6: The subject hierarchy of a Web catalog

inclusion relation (e.g. `Crete` $\preceq$ `Greece`). For example, the ontology shown in Figure 2.7, as well as, the Thesaurus of Geographical Names (TGN [63]), fit in this case.

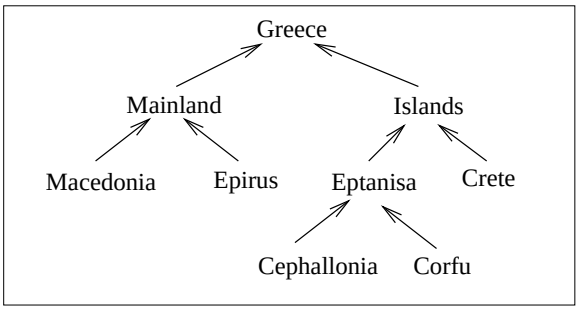


Figure 2.7: An ontology of geographical names

- Ontologies which consist of terms that denote time intervals (or events) structured by the temporal relation *during* (e.g. `Battle Of Crete` $\preceq$ `World War II`)
- Taxonomies (for products, living species, fields of knowledge) in which the terms correspond to categories structured by a subclass relation (e.g. `Canaries` $\preceq$ `Birds`). For example, the ontology shown in Figure 2.8, and many existing thesauri fit in this case.

2.3 Ontology-based Sources: Definition

Let Obj denote the set of all objects of the domain. A typical example of such a domain is the set of all pointers to Web pages.

Def 2.2 Given a terminology T, we call *interpretation* of T over Obj any function $I : T \rightarrow 2^{Obj}$.

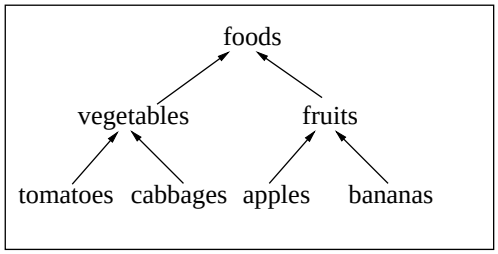


Figure 2.8: A taxonomy of foods

Here we use the symbol 2^{Obj} to denote the power set of Obj. Thus each term t denotes a set of objects in Obj and its interpretation $I(t)$ is the set of objects to which the term t correctly applies. In our discussion the set Obj will usually be understood from the context. So, we shall often say "an interpretation" instead of "an interpretation over Obj". Interpretation, as defined above, assigns a denotational or extensional meaning to a term [142].

A source has an ontology $(T, \preceq)$ and a stored *interpretation* I of its terminology. Figure 2.9 shows an example of a source.

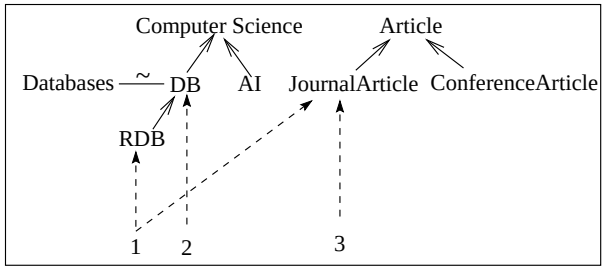


Figure 2.9: Graphical representation of a source

In this and subsequent figures the objects are represented by natural numbers and membership of objects to the interpretation of a term is indicated by a dotted arrow from the object to that term. For example, the objects 1 and 3 in Figure 2.9 are members of the interpretation of the term `JournalArticle` as these objects are connected to `JournalArticle` by dotted arrows. As these are the only objects connected to `JournalArticle` by dotted arrows, they make up the interpretation of `JournalArticle`, i.e. $I(\texttt{JournalArticle}) = \{1, 3\}$.

Moreover, we assume that every interpretation I of T satisfies the condition $I(\bot) = \emptyset$.

2.4 Query Answering: The Sure and the Possible Answer

A source can respond to queries over its own terminology. A query is either a term or a combination of terms obtained using the connectives $\wedge$, $\vee$, $\neg$ and (). For technical reasons that will become clear shortly we shall also use the concept of *empty query* denoted by ϵ. More formally, a query is defined as follows:

Def 2.3 Let T be a terminology. A *query* over T is any string derived by the following grammar, where t is a term of T:

$$q ::= \ t \mid q \wedge q' \mid q \vee q' \mid q \wedge \neg q' \mid (q) \mid \epsilon$$

Note that our use of negation corresponds to domain restricted negation.

The set of interpretations of a given terminology T can be ordered using pointwise set inclusion.

Def 2.4 Given two interpretations I, I' of T, we call I less than or equal to I', and we write $I \sqsubseteq I'$, if $I(t) \subseteq I'(t)$ for each term $t \in T$.

Note that $\sqsubseteq$ is a partial order over interpretations.

A source answers queries based on the stored interpretation of its terminology. However, in order for query answering to make sense, the interpretation that a source uses for answering queries must respect the structure of the source's ontology (i.e. the relation $\preceq$) in the following sense: if $t \preceq t'$ then $I(t) \subseteq I(t')$. For example, consider a source whose ontology contains only three terms: `DB`, `AI` and `Computer Science`, where `DB` $\preceq$ `Computer Science`, and `AI` $\preceq$ `Computer Science`. Assume that in the stored interpretation I of the source we have: $I(\texttt{DB}) \neq \emptyset$, $I(\texttt{AI}) \neq \emptyset$ and $I(\texttt{ComputerScience}) = \emptyset$. Clearly, I does not respect the structure of the ontology, as `DB` $\preceq$ `Computer Science`, and yet $I(\texttt{DB}) \not\subseteq I(\texttt{ComputerScience})$. However, I is acceptable as we can "augment" it to a new interpretation I' that *does* respect the structure of the ontology. The interpretation I' is defined as follows: $I'(\texttt{DB}) = I(\texttt{DB})$, $I'(\texttt{AI}) = I(\texttt{AI})$, $I'(\texttt{ComputerScience}) = I(\texttt{ComputerScience}) \cup I(\texttt{DB}) \cup I(\texttt{AI})$. An interpretation such as I' that respects the structure of an ontology is what we call a *model* of that ontology.

Def 2.5 An interpretation I is a *model* of an ontology $(T, \preceq)$ if for all t, t' in T, if $t \preceq t'$ then $I(t) \subseteq I(t')$.

For brevity hereafter we shall sometimes write T instead of $(T, \preceq)$, whenever no confusion is possible.

Now, as there may be several models of T in general, we assume that each source answers queries from one or more *designated* models induced by its stored interpretation. Here, we will use *two* specific models for answering queries, the *sure model* and the *possible model*. In order to define these models formally we need to introduce the notions of *tail* and *head* of a term.

Def 2.6 Given a term $t \in T$ we define

$$tail(t) = \{s \in T \mid s \preceq t\} \quad \text{and} \quad head(t) = \{u \in T \mid t \preceq u\}$$

Note that t, and all terms that are equivalent to t, belong to both $tail(t)$ and $head(t)$. Also note that $tail(t)$ always contains the bottom term $\bot$ and $head(t)$ always contains the top term $\top$.

Def 2.7 Given an interpretation I of T we define the *sure model* of T generated by I, denoted I^-, as follows:

$$I^-(t) = \bigcup\{I(s) \mid s \in tail(t)\}$$

Intuitively the stored set $I(t)$ consists of the objects that are known to be indexed under t. The set $I^-(t)$ on the other hand consists of the objects known to be indexed under t plus the objects that are known to be indexed under terms subsumed by t. Therefore $I^-(t)$ consists of all objects that are *surely* indexed under t with respect to I and $\preceq$. Figure 2.10 shows an example of a source and its sure model I^-.

Prop. 2.1 If I is an interpretation of T then I^- is the unique minimal model of T which is greater than or equal to I.

Proof:

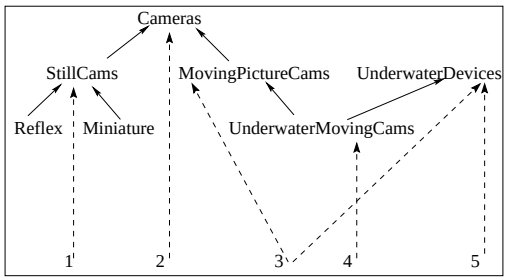
(I^- is a model of T)

$t \preceq t' \Rightarrow tail(t) \subseteq tail(t') \Rightarrow \bigcup\{I(s) \mid s \in tail(t)\} \subseteq \bigcup\{I(s) \mid s \in tail(t')\} \Rightarrow I^-(t) \subseteq I^-(t')$.

We conclude that I^- is a model of T.

(I^- is the unique minimal model of T which is greater than I)

Let I' be a model of T which is larger than I. Below we prove that $I^- \sqsubseteq I'$. By the definition of $I^-(t)$, if $o \in I^-(t)$ then either $o \in I(t)$ or $o \in I(s)$ for a term s such that $s \preceq t$. However, if $o \in I(t)$ then $o \in I'(t)$ too because I' is larger than I, and if $o \in I(s)$ for a term s such that $s \preceq t$ then $o \in I'(t)$ too because I' is a model of



(a)

Term	I	I^-	I^+
$\bot$	$\emptyset$	$\emptyset$	$\emptyset$
$\top$	$\emptyset$	**{1,2,3,4,5}**	{1,2,3,4,5}
Cameras	{2}	{**1**,2,**3**,**4**}	{1,2,3,4,**5**}
StillCams	{1}	{1}	{1,**2**,**3**,**4**}
Reflex	$\emptyset$	$\emptyset$	**{1}**
Miniature	$\emptyset$	$\emptyset$	**{1}**
MovingPictureCams	{3}	{3,**4**}	{**1**,**2**,3,4}
UnderwaterMovingCams	{4}	{4}	{**3**,4}
UnderwaterDevices	{3,5}	{3,**4**,5}	{**1**,**2**,3,4,5}

(b)

Figure 2.10: Graphical representation of a source

T. We conclude that for each $o \in I^-(t)$, it holds $o \in I'(t)$. Thus I^- is the unique minimal model T which is larger than I.

$\diamond$

Def 2.8 Given an ontology T and interpretation I we define the *possible model* of T generated by I, denoted I^+, as follows:

$$I^+(t) = \bigcap\{I^-(u) \mid u \in head(t) \text{ and } u \not\sim t\}$$

It is clear from the definition that the set $I^+(t)$ consists of the objects known to be indexed under each term strictly subsuming t. Therefore $I^+(t)$ consists of all objects that are *possibly* indexed under t with respect to I and $\preceq$. An example of the possible model of a source is given in Figure 2.10. In this example we have:

$$\begin{aligned} I^+(\texttt{Reflex}) &= \{1\} \\ I^+(\texttt{UnderwaterMovingCams}) &= \{3, 4\} \end{aligned}$$

Note that the possible interpretations of the terms `Cameras` and `UnderwaterDevices` is the set of *all* stored objects. This is so because the head of each of these terms contains only the term itself and the top term $\top$, thus we have:

$$I^+(\texttt{Cameras}) = I^+(\texttt{UnderwaterDevices}) = I^-(\top) = \bigcup\{I(s) \mid s \preceq \top\} = \bigcup\{I(t) \mid t \in T\}$$

Note that since $head(\top) = \{\top\}$, the set $\{\ u \in head(\top) \mid u \not\sim \top\}$ is empty. This means that $I^+(\top)$, i.e. $\bigcap\{I^-(u) \mid u \in head(\top) \text{ and } u \not\sim \top\}$ is actually the intersection of an empty family of subsets of Obj. However, according to the Zermelo axioms of set theory (see [15] for an overview), the intersection of an empty family of subsets of a universe equals to the universe.

In our case, the universe is the set of all objects known to the source, i.e. the set $I^-(\top)$, thus we conclude that $I^+(\top) = I^-(\top)$.

Prop. 2.2 If I is an interpretation of T then I^+ is a model of T and $I \sqsubseteq I^- \sqsubseteq I^+$.

Proof:

(I^+ is a model of T)

$t \preceq t' \Rightarrow \{u \mid u \in head(t)\} \supseteq \{u \mid u \in head(t')\} \Rightarrow$
$\{u \mid u \in head(t) \text{ and } u \not\sim t\} \supseteq \{u \mid u \in head(t') \text{ and } u \not\sim t'\} \Rightarrow$
$\bigcap\{I(u) \mid u \in head(t) \text{ and } u \not\sim t\} \subseteq \bigcap\{I(u) \mid u \in head(t') \text{ and } u \not\sim t'\} \Rightarrow$
$I^+(t) \subseteq I^+(t')$.

($I^- \sqsubseteq I^+$)

Clearly, if $t \in T$, $u \in head(t)$ and $u \not\sim t$ then in every model I of T we have $I(t) \subseteq I(u)$. Thus this also holds in the model I^-, i.e. $I^-(t) \subseteq I^-(u)$. From this we conclude that for every $t \in T$:
$I^-(t) \subseteq \bigcap\{I^-(u) \mid u \in head(t) \text{ and } u \not\sim t\} = I^+(t)$. Thus $I^- \sqsubseteq I^+$.

$\diamond$

It follows from the above proposition that for every term t we have $I^-(t) \subseteq I^+(t)$.

We view the stored interpretation I as the result of indexing. However, although we may assume that indexing is done correctly, certain objects may not have been indexed under all terms that could apply to them. For example object 1 in Figure 2.10 is indexed under `StillCams` but not under `Cameras`, and object 3 is indexed under `MovingPictureCams` and `UnderwaterDevices` but not under `UnderwaterMovingCams`. Note that object 3 could in fact be an `UnderwaterMovingCamera` but it was not indexed under this term because either the indexer by mistake did not use this term, or the term `UnderwaterMovingCamera` was defined after the indexing of object 3.

By consequence, given a query that consists of a single term t we may want to answer it in either of two ways: (a) by including in the answer only objects that are known to be indexed under t, or (b) by including in the answer objects that are possibly indexed under t. In the first case the answer is the set $I^-(t)$, while in the second it is the set $I^+(t)$.

Remark. If we consider that each term corresponds to a property or characteristic of the objects of the domain, then $t \preceq t'$ means that if an object has the property t then it also has the property t'. In this view,

- $I(t)$ consists of the objects that have the set of properties $head(t)$.
- $I^-(t)$ consists of the objects that have at least the set of properties $head(t)$. Each of these objects also has at least one property t' such that $t' \preceq t$.
- $I^+(t)$ consists of the objects that have at least the set of properties $head(t) \setminus \{t\}$.

If we consider that each term denotes a time interval then $t \preceq t'$ means that t is included in t'. In this view,

- $I(t)$ consists of the objects that exist/occur at the time interval t.
- $I^-(t)$ consists of the objects that exist/occur during the time interval t.
- $I^+(t)$ consists of the objects that exist/occur during the interval that is obtained by taking the intersection of all intervals that include t and are not equal to t.

An analogous discussion applies to the case where each term denotes a spatial region and $\preceq$ is a spatial inclusion relation.

Referring to Def. 2.3 let us now define query answering for a general query q.

Def 2.9 Let q be a query over a terminology T and let I be an interpretation of T.
(a) The sure answer $I^-(q)$ is defined as follows:

$$\begin{aligned}
I^-(t) &= \bigcup\{I(s) \mid s \in tail(t)\} \\
I^-(q \wedge q') &= I^-(q) \cap I^-(q') \\
I^-(q \vee q') &= I^-(q) \cup I^-(q') \\
I^-(q \wedge \neg q') &= I^-(q) \setminus I^-(q') \\
I^-(\epsilon) &= \emptyset
\end{aligned}$$

(b) The possible answer $I^+(q)$ is defined as follows:

$$\begin{aligned}
I^+(t) &= \bigcap\{I^-(u) \mid u \in head(t) \text{ and } u \not\sim t\} \\
I^+(q \wedge q') &= I^+(q) \cap I^+(q') \\
I^+(q \vee q') &= I^+(q) \cup I^+(q') \\
I^+(q \wedge \neg q') &= I^+(q) \setminus I^-(q') \\
I^+(\epsilon) &= \emptyset
\end{aligned}$$

It follows easily from the above definition that for every query q we have $I^-(q) \subseteq I^+(q)$. This means that the sure answer of a query q is always included in the possible answer of q.

Note that we interpret $I^+(q \wedge \neg q')$ by $I^+(q) \setminus I^-(q')$, and *not* by $I^+(q) \setminus I^+(q')$. This is because if we had interpreted $I^+(q \wedge \neg q')$ by $I^+(q) \setminus I^+(q')$ then we could have found queries q for

which $I^-(q) \supset I^+(q)$, contrary to intuition. For example, consider a terminology T with three terms a,b and c such that $c \preceq b \preceq a$, and an interpretation I such that $I(c) = \emptyset$, $I(b) = \{1\}$ and $I(a) = \{2\}$. Then for $q = a \wedge \neg c$ we would have had: $I^-(q) = I^-(a) \setminus I^-(c) = \{1, 2\}$ and $I^+(q) = I^+(a) \setminus I^+(c) = \{2\}$, i.e. $I^-(q) \supset I^+(q)$. However, with our definition we have $I^+(a \wedge \neg c) = I^+(a) \setminus I^-(c) = \{1, 2\}$, i.e. the relation $I^- \sqsubseteq I^+$ is preserved.

User interaction with the source consists of submitting a query q plus the nature of the desired answer (sure or possible). The system then responds by computing $I^-(q)$ or $I^+(q)$ according to the user's desire. The possibility of providing two types of answer to a query can enhance the quality of user interaction with the source. For example, the user may submit a query and require a sure answer. If the sure answer is empty this may mean either that no object has been indexed under the user's query or that the objects have been indexed at a coarser level. So, if the sure answer turns out to be empty, the user can ask for the possible answer to his query. In the possible answer the user can see objects related to, but not necessarily indexed under his query. Another possibility is that the sure answer to the query is not empty but the user just likes to see more objects related to his query, but at a coarser level. Then he can ask for a possible answer to his query.

2.5 Enhancing Answers with Object Descriptions

Consider a source that contains an object 1 indexed under two terms, `Cameras` and `Underwater`, and an object 2 also indexed under two terms, `Cameras` and `Miniature`. Next, assume that the source receives the query q = `Cameras` and is asked to return both the sure and the possible answer to that query. Clearly the objects 1 and 2 will be included in both answers returned by the source. However, instead of just returning the set $\{1, 2\}$, the source could return the following set

$$\{(1, \{\texttt{Cameras}, \texttt{Underwater}\}),\ (2, \{\texttt{Cameras}, \texttt{Miniature}\})\}$$

In this set each object is accompanied by the set of *all* terms under which the object is indexed. This information could provide valuable help to the user. Indeed, the user of our example may have actually been looking for miniature cameras, but he only used the term `Cameras` in his query for a number of possible reasons. For example,

- the user forgot to use the term `Miniature`; or
- the user did not know that the term `Miniature` was included in the terminology of the source; or

- the user did not know that the objects of the source were indexed in such specificity.

Including in the answer all terms under which the objects returned are indexed can aid the user in selecting the objects that are most relevant to his information need. In addition, such terms could aid the user in getting better acquainted with the ontology of the source. Indeed, more often than not, users are not familiar with the source ontology and know little about its specificity and coverage (for more about this problem see [85]). As a result user queries are often imprecise and do not reflect the real user needs. We believe that familiarity with the source ontology is essential for a precise formulation of user queries. Therefore we extend the notion of answer to be a set of objects each accompanied by its *index*, i.e. by the set all terms under which the object is indexed. This approach has an educative effect on the user about the information space.

Def 2.10 The *index* of an object o with respect to an interpretation I, denoted by $D_I(o)$, is the set of all terms that contain o in their interpretation, i.e. $D_I(o) = \{t \in T \mid o \in I(t)\}$.

For brevity hereafter we shall sometimes write $D(o)$ instead of $D_I(o)$, $D^-(o)$ instead of $D_{I^-}(o)$, and $D^+(o)$ instead of $D_{I^+}(o)$, when the interpretation I is clear from the context. Clearly the index of an object depends on the interpretation I, so the same object can have different indexes under different interpretations. Here are some examples of indexes in the source shown in Figure 2.10:

$$
\begin{aligned}
D(1) &= \{\texttt{StillCams}\} \\
D^-(1) &= \{\texttt{StillCams}, \texttt{Cameras}\} \\
D^+(1) &= \{\texttt{StillCams}, \texttt{Cameras}, \texttt{Reflex}, \texttt{Miniature}, \texttt{MovingPictureCams}, \texttt{UnderwaterDevices}\} \\
D(2) &= \{\texttt{Cameras}\} \\
D^-(2) &= \{\texttt{Cameras}\} \\
D^+(2) &= \{\texttt{Cameras}, \texttt{StillCams}, \texttt{MovingPictureCams}, \texttt{UnderwaterDevices}\}
\end{aligned}
$$

We have seen earlier that the user of a source can submit a query and ask for a sure or a possible answer. Following our discussion on indexes, the user can now also ask for the sure or possible index for each object in the answer. This means that the answer returned by the source to a given query q can have one of the forms shown in Table 2.1. It is up to the user to specify the desired form of the answer.

	object set	object index	answer returned
1	sure	sure	$\{ (o, D^-(o)) \mid o \in I^-(q)\}$
2	sure	possible	$\{ (o, D^+(o)) \mid o \in I^-(q)\}$
3	possible	sure	$\{ (o, D^-(o)) \mid o \in I^+(q)\}$
4	possible	possible	$\{ (o, D^+(o)) \mid o \in I^+(q)\}$

Table 2.1: The answers to a query q

Note that if I^- is stored at the source then the evaluation of D^- for an object o is straightforward. If however only the interpretation I is stored at the source then we can compute $D^-(o)$ as follows:

Prop. 2.3 $D^-(o) = \bigcup\{head(t) \mid o \in I(t)\}$, or equivalently, $D^-(o) = \bigcup\{head(t) \mid t \in D(o)\}$.
Proof:

$$\begin{aligned} D^-(o) &= \{t \in T \mid o \in I^-(t)\} = \{t \in T \mid o \in \cup\{I(s) \mid s \preceq t\}\} \\ &= \{t \in T \mid o \in I(s) \text{ and } s \preceq t\} = \bigcup\{head(s) \mid o \in I(s)\} \end{aligned}$$

◇

If I^+ is stored at the source then the evaluation of D^+ for an object o is straightforward. Otherwise, we can compute $D^+(o)$ through $D^-(o)$ as follows:

Prop. 2.4 $D^+(o) = \{t \mid head(t) \setminus \{t' | t' \sim t\} \subseteq D^-(o)\}$
Proof:

$$\begin{aligned} &t \in D^+(o) \Leftrightarrow o \in I^+(t) \Leftrightarrow o \in \bigcap\{I^-(u) \mid u \in head(t) \text{ and } u \not\sim t\} \Leftrightarrow \\ &o \in I^-(u) \; \forall\, u \in head(t) \text{ s.t. } u \not\sim t \Leftrightarrow u \in D^-(o) \; \forall\, u \in head(t) \text{ s.t. } u \not\sim t \Leftrightarrow \\ &u \in D^-(o) \; \forall\, u \in head(t) \setminus \{t' | t' \sim t\} \Leftrightarrow head(t) \setminus \{t' | t' \sim t\} \subseteq D^-(o). \end{aligned}$$

◇

2.6 Implementation

A source can be implemented using any of a number of data models. For example, using the relational model [28], we can implement a source as a database schema consisting of three tables, one for storing the terminology, one for storing the subsumption relation, and one for storing the interpretation I of the terminology:

TERMINOLOGY(term-id:`Int`, term-name:`Str`)

SUBSUMPTION(term1:`Int`, term2:`Int`)

ODB(term-id:`Int`, obj:`Int`)

Note that each term of the terminology is stored in the form of a pair <term-id, term-name> where "term-id" is an internal identifier. For the purposes of this chapter, however, we assume that term identifiers are integers and term names are strings. Figure 2.11 shown the implementation of the source shown in Figure 2.10.

TERMINOLOGY

term-id	*term-name*
1	Cameras
2	StillCams
3	Reflex
4	Miniature
5	MovingPictureCams
6	UnderwaterMovingPictureCams
7	UnderwaterDevices

SUBSUMPTION

term1	*term2*
2	**1**
3	**2**
4	**2**
5	**1**
6	**5**
6	**7**

ODB

term-id	*object*
2	*1*
1	*2*
5	*3*
6	*4*
7	*3*
7	*5*

Figure 2.11: The source of Figure 2.10 according to the relational model

2.7 Query Evaluation

Concerning query evaluation at a source, there are basically two approaches. The first approach consists of computing and storing the models I^- and I^+ and then using these stored models for computing answers to queries. This can be done using algorithms that follow easily from Definition 2.9. The advantage of this approach is that answers can be computed in a straightforward manner from the stored models. The disadvantage is increased space requirements as well as increased maintenance costs for the stored models. Indeed, whenever the ontology or the interpretation I change, I^- and I^+ must be updated appropriately. This requires an efficient method for handling updates since recomputing I^- and I^+ from scratch would be inefficient.

The second approach consists of storing only the interpretation I and, whenever a query q is received, computing the appropriate answer, $I^-(q)$ or $I^+(q)$ using I. The computation of $I^-(q)$

can be done in a straightforward manner following Definition 2.9.(a). The computation of $I^+(q)$ can be done again following Definition 2.9.(b) but requires the previous computation of $I^-(t)$ for all terms t that subsume terms which appear in the query. The advantage of this approach is that we have no additional space requirements and no additional maintenance costs. The disadvantage is increased time cost for the computation of the answers.

Clearly the relative merits of the two approaches depend on the application on hand as well as on the frequency by which the ontology and/or the stored interpretation of the source are updated.

Note that in both approaches we need algorithms for computing the head and the tail of a term. However, if we compute the transitive closure of the subsumption relation, by one of the existing algorithms (e.g. see [99]), then the algorithms for computing the head and tail of a term follow immediately from Definition 2.6. The complexity of evaluating the transitive closure of $\preceq$ is polynomial. For instance, the time complexity of the Floyd-Warshall algorithm is cubic in the number of terms, and the space used is at most quadratic in the number of terms. If the entire subsumption relation $\preceq$ is stored, i.e. if the transitive relationships are stored, then the computation of $head(t)$ and $tail(t)$ can be done in $O(|\preceq|)$ time. If only the interpretation I is stored, then the computation of $I^-(t)$ requires taking the union of at most $|T|$ subsets of Obj. If U denotes the number of objects that are stored in the source[5] then the union of two subsets of Obj can be computed in $O(U)$ time. Thus the computation of $I^-(t)$ can be done in $O(|T| * U)$ time [6]. If the sure model I^- is stored, then the computation of $I^+(t)$ requires taking the intersection of at most $|T|$ subsets of Obj. Thus the computation of $I^+(t)$ can be done in $O(|T| * U)$ time. If only the interpretation I is stored, then the computation of $I^+(t)$ can be done as follows:

$$I^+(t) = \bigcap_{u \succ t} (\bigcup \{I(s) \mid s \preceq u\})$$

This computation can be done in $O(|T|^2 * U)$ time.

2.8 Inference Mechanisms

This section describes mechanisms for performing some basic reasoning tasks over the ontology of a source. We shall exploit these mechanisms in the subsequent chapters.

[5] Specifically, $U = |adom(I)| \leq |Obj|$.

[6] Note that here we express the execution time with respect to two parameters: the size of the terminology and the number of the stored objects.

Since the objects of a source are indexed by one or more terms, we focus on conjunctions of terms, hereafter called *descriptions*.

Def 2.11 A *description* d over a terminology T is either a term $t \in T$ or a sequence of terms separated by "$\cdot$", i.e. any string derived by the following grammar:

$$d ::= d \cdot t \mid t$$

Let D denote the set of all descriptions over a terminology T. An interpretation I of T can be extended to an interpretation of D as follows: for any description $d = t_1 \cdot t_2 \cdot .. \cdot t_k$ in D we define $I(d) = I(t_1) \cap I(t_2) \cap ... \cap I(t_k)$.

Def 2.12 Let $A = (T, \preceq)$ be an ontology, let d, d' be two descriptions over T, and let I be an interpretation of T. We say that:

- d is *subsumed by* d' in I, and we write $I \models d \preceq d'$, if $I(d) \subseteq I(d')$,
- d *is subsumed by* d' in A, and we write $A \models d \preceq d'$, if $I \models d \preceq d'$ in every model I of A,
- d and d' are *equivalent* in I, and we write $I \models d \sim d'$, if $I(d) = I(d')$,
- d and d' *are equivalent* in A, and we write $A \models d \sim d'$, if $I \models d \sim d'$ in every model I of A.

Below we provide the mechanisms needed for checking *equivalence* and *subsumption* of descriptions in A.

Let $(T, \preceq)$ be an ontology and let $T/\!\sim$ be the set of equivalence classes induced by "$\preceq$" over T. We can extend the relation "$\preceq$" over the set $T/\!\sim$ as follows: for all c, c' in $T/\!\sim$, $c \preceq c'$ iff there is $t \in c$ and $t' \in c'$ such that $t \preceq t'$. We shall use the same symbol for both "$\preceq$" and its extension over $T/\!\sim$, as the distinction will be clear from the context. For each term $t \in T$ we denote by $c(t)$ the equivalence class of t.

Def 2.13 Given a description $d = t_1 \cdot ... \cdot t_k$, we call *reduction of* d, denoted by $r(d)$, the following subset of $T/\!\sim$: $\quad r(t_1 \cdot ... \cdot t_k) \;=\; min_{\preceq}\{\, c(t_1), ..., c(t_k) \,\}$

For example, if the terms t_i, t_j appear in d and if $c(t_i) \preceq c(t_j)$ then $c(t_j)$ is not an element of $r(d)$. Clearly if $r(d) = \{c_1, ...c_k\}$, then $c_i \not\preceq c_j$ for all $i, j = 1..k$, and note that there is only one reduction for any given d.

Prop. 2.5 Let A be an ontology and d a description over A. For every model I_m of A it holds $I_m(d) = \bigcap_{c \in r(d)} I_m(c)$
Proof:

Let $d = t_1 \cdot ... \cdot t_k$ and I_m be a model of A. It follows from Def. 2.5 that all terms of an equivalence class $c \in T/\sim$ have the same interpretation, and let us denote this interpretation by $I_m(c)$. Thus $I_m(d) = I_m(t_1) \cap .. \cap I_m(t_k) = I_m(c(t_1)) \cap .. \cap I_m(c(t_k))$. Moreover, and according to Def. 2.5, if $c \preceq c'$ then $I_m(c) \subseteq I_m(c')$, thus $I_m(c) \cap I_m(c') = I_m(c)$. This implies that $I_m(c(t_1)) \cap .. \cap I_m(c(t_k)) = \bigcap_{c \in min_{\preceq}\{c(t_1),...,c(t_k)\}} I_m(c)$. Hence, $I_m(d) = \bigcap_{c \in r(d)} I_m(c)$.

◇

Lemma 2.1 Let d, d' be two descriptions over A. If $r(d) = r(d')$ then $I_m(d) = I_m(d')$ in every model I_m of A.
Proof:

According to Prop. 2.5, in every model I_m of A we have $I_m(d) = \cap_{c \in r(d)} I_m(c)$ and $I_m(d') = \cap_{c \in r(d')} I_m(c)$. Since $r(d) = r(d')$ we conclude that $I_m(d) = I_m(d')$.

◇

Let d, d' be two descriptions over A. From the above lemma it is clear that if $r(d) = r(d')$ then certainly $A \models d \sim d'$. Thus we can use this method for checking the validity of synonymies. Moreover this method is complete, that is, $A \models d \sim d'$ iff $r(d) = r(d')$. This is proved by the next theorem.

Theorem 2.1 $A \models d \sim d' \Rightarrow r(d) = r(d')$
Proof:

Let a, b be two descriptions over A, such that $r(a) \neq r(b)$. Assume that $r(a) = \{a_1, .., a_k\}$ and $r(b) = \{b_1, .., b_l\}$, where $k, l \geq 1$ Below we prove that we can always construct a model m_x, such that $m_x(a) \neq m_x(b)$. As $r(a) \neq r(b)$, one of the following three must hold:
(i) $r(a) \not\subset r(b)$, or
(ii) $r(b) \not\subset r(a)$, or
(iii) $r(a) \not\subset r(b)$ and $r(b) \not\subset r(a)$.

Assume case (i), that is, $r(a) \not\subset r(b)$. This means that there exists at least one a_x such that $a_x \notin r(b)$. Now consider a model m of A such that $m(a) = m(b)$. We can "enlarge" m, to a model m_x such that $m_x(a) \neq m_x(b)$. This enlargement consists of two steps:

Step 1. Add a new object o_b to each $m(b_i)$

Step 2. Add the object o_b to the interpretation of each $c \in T/\sim$, such that $c \succeq b_i$ for some $i = 1, .., l$.

Let m' denote the resulting interpretation after Step 1. Clearly, $o_b \in m'(b)$, while $o_b \notin m'(a)$, since o_b was not added to the interpretation of a_x. Note that the interpretation m' is not necessarily a model of A. However the interpretation after Step 2 is certainly a model of A.

What remains to show is that Step 2 did not add the object o_b to each $m(a_i)$, since in that case it would be $m'(a) = m'(b)$. Clearly, this Step would add o_b to each $m(a_i)$ only if:

$$\forall\, a_i\ \exists\, b_j :\ a_i \succeq b_j$$

Note that certainly it cannot be: $\forall\, a_i\ \exists\, b_j :\ a_i \sim b_j$ since in that case it would be $r(a) \subseteq r(b)$, and due to $r(a) \neq r(b)$, it would be $r(a) \subset r(b)$, which contradicts the hypothesis $r(a) \not\subset r(b)$. Thus, in view of our hypothesis, we can say that Step 2 would update each $m'(a_i)$ only if:

$$\forall\, a_i\ \exists\, b_j :\ a_i \succeq b_j \text{ and } \exists\, a_y\ \exists\, b_y :\ a_y \succ b_y$$

From this formula we can see that it must be $b_y \in r(b) \setminus r(a)$, otherwise we would have $a_y \succ b_y = a_i$ for some i. which would contradict the definition of $r(a)$ (since it would be $a_y \succ a_i$, therefore a_y should not be an element of $r(a)$). Thus we can write

$$\forall\, a_y \in r(a) \setminus r(b) \quad \exists\, b_y \in r(b) \setminus r(a) \text{ such that } \quad a_y \succ b_y \tag{2.1}$$

Notice that this formula also implies that $r(b) \setminus r(a) \neq \emptyset$, that is $r(b) \not\subset r(a)$.

If formula (2.1) is satisfied then the enlargement process will result in a model m' such that $m'(a) = m'(b)$, which means that we failed to construct the model m_x that we are looking for. However in this case, we can try the opposite direction, that is, we can add a new object o_a to each $m(a_i)$. Certainly $o_a \in m(a)$ while $o_a \notin m(b)$ (since $r(b) \not\subset r(a)$). Now we enlarge m for making it a model. Again, what remains to show is that this step did not add o_a to each $m(b_i)$ too. By a similar analysis we reach the conclusion, that this step would update each $m(b_i)$ only if

$$\forall\, b_y \in r(b) \setminus r(a) \quad \exists\, a_y \in r(a) \setminus r(b) \text{ such that } \quad b_y \succ a_y \tag{2.2}$$

From this analysis we conclude that we cannot construct the model m_x that we are looking for, only if the formulas (2.1), and (2.2) *both* hold. Formula (2.1) imply that there exists $a_z \in r(a) \setminus r(b)$, and $b_z \in r(b) \setminus r(a)$, such that $a_z \succ b_z$. Formula (2.2) imply that there exists $a_{z'} \in r(a) \setminus r(b)$, such that $b_z \succ a_{z'}$. Notice that if $a_z \sim a_{z'}$ then this would contradict the definition of $r(a)$ (since it would be $a_z \succ a_{z'}$, therefore a_z should not be an element of $r(a)$). Thus we conclude that the formulas (2.1) and (2.2) cannot be both true.

This means that we can always construct a model m_x such that $m_x(a) \neq m_x(b)$. This implies that if $r(a) \neq r(b)$ then $A \not\models a \sim b$, that is, $A \models d \sim d' \Rightarrow r(d) = r(d')$.

$\diamond$

Concerning subsumptions, we can reduce subsumption checking to equivalence checking:

Prop. 2.6 $A \models d \preceq d'$ iff $A \models d \cdot d' \sim d$
Proof:
($\Rightarrow$)

$A \models d \preceq d'$ means that in every model m it holds $m(d) \subseteq m(d')$. The latter implies that $m(d) \cap m(d') = m(d)$, hence $A \models d \cdot d' \sim d$.

($\Leftarrow$)

$A \models d \cdot d' \sim d$ means that in every model m it holds $m(d) \cap m(d') = m(d)$. The latter implies that $m(d) \subseteq m(d')$, hence $A \models d \preceq d'$.

$\diamond$

An alternative method for checking equivalence and subsumption is described next.

If $d = t_1 \cdot ... \cdot t_k$ then let $head(d) = head(t_1) \cup ... \cup head(t_k)$.

Prop. 2.7 $r(d) = r(d')$ iff $head(d) = head(d')$.
Proof:

Trivial

$\diamond$

Thus we can use the heads of descriptions in order to check equivalence and subsumption.

Lemma 2.2
$A \models d \sim d'$ iff $head(d) = head(d')$.
$A \models d \preceq d'$ iff $head(d.d') = head(d)$.

Note that if the entire $\preceq$ is stored then the computation of $head(t_1 \cdot ... \cdot t_k)$ can be done in $O(k* |\preceq|)$. Thus given two descriptions $d = t_1 \cdot ... \cdot t_k$ and $d' = t'_1 \cdot ... \cdot t'_k$, we can check whether $A \models d \sim d'$ in $O(|\preceq| *max(k, k'))$ time and we can check whether $A \models d \preceq d'$ in $O(|\preceq| *l)$ time where $l = k + k'$.

2.9 Summary and Conclusion

In this chapter we introduced ontology-based sources. Examples of such sources include Web catalogs and personal bookmarks. We gave a model-theoretic interpretation to these sources and we described the query answering process. An essential feature of these sources is that they support a form of query relaxation. Specifically, they can provide two types of answer to a given query, namely, a *sure* answer or a *possible* answer. The first type of answer is appropriate for a user who does not want to retrieve objects which are not relevant to his information need, while the second for a user who does not want to miss objects which are relevant to his information needs.

Another feature of the sources described in this chapter, is that each object of an answer can be accompanied by its description, i.e. by all terms under which the object is indexed. These descriptions can aid the user in (a) selecting the objects of the answer that are most relevant to his information need, and (b) getting acquainted with the ontology of the source.

Chapter 3

Extended Faceted Ontologies

Chapter 2 presented sources which are based on ontologies that have a hierarchical structure. This chapter presents a novel scheme for designing ontologies that consist of *multiple* independent aspects, or *facets*. Faceted schemes have several advantages by comparison to the hierarchical classification schemes that are currently employed by Web catalogs, namely,

- *conceptual clarity* (they are easier to understand),
- *compactness* (they take less space), and
- *scalability* (update operations can be formulated easier and be performed more efficiently).

The distinctive feature of our faceted scheme is that it is equipped with a method for specifying the valid combinations of terms. We give a model-theoretic interpretation to this scheme and we provide mechanisms for inferring the valid combinations of terms. This inference service can aid the construction of ontology-based sources in many ways. It can be exploited for preventing errors during the indexing process, which is very important especially in the case where the indexing is done collaboratively by many users. In addition, it allows generating hierarchical navigation trees dynamically, thus addressing the difficulties involved in browsing, or in describing (indexing) objects by means of faceted ontologies.

The remaining of this chapter is organized as follows: Section 3.1 discusses hierarchical and faceted classification schemes, while Section 3.2 presents some examples that illustrate the advantages of faceted ontologies and the need for extending them. Section 3.3 defines faceted ontologies formally and Section 3.4 introduces the extended faceted ontologies. Section 3.5 describes the mechanism for inferring the validity of descriptions and Section 3.6 describes a mechanism for generating navigation trees for the extended faceted ontologies. Section 3.7 discusses the creation of sources using extended faceted ontologies, Section 3.8 discusses related work and finally, Section 3.9 summarizes and concludes this chapter.

3.1 Hierachical and Faceted Classification Schemes

Commonly, the ontologies of Web catalogs have a tree or a directed acyclic graph structure. The nodes correspond to terms (e.g. `Sciences`, `Mathematics`) and the edges correspond to subsumption relationships (e.g. `Mathematics` $\preceq$ `Sciences`). Such ontologies may contain thousands of terms, e.g. the ontology of Yahoo! contains 20 thousands terms while the ontology of OpenDirectory contains 300 thousands terms!

We can consider these ontologies as *hierarchical classification* schemes like the Library of Congress Subject Headings [4] and the Dewey Decimal system. For all their usefulness, these ontologies have certain drawbacks with regard to comprehensibility, storage requirements and scalability. Hierarchical classification schemes require an *a priori* division of concepts into subconcepts, according to combinations of various criteria, such that every object in the domain can be assigned a unique sufficiently detailed descriptive term. Refining or revising such a scheme is a complex operation that also entails possibly extensive re-classification of the objects.

By contrast, a *faceted classification* scheme (see [82] for a review) does not rely on the breakdown of a universe, but on building up, or *synthesizing*, from the subject statements of the objects. In a faceted scheme, subject statements are analyzed into their component elemental concepts, and these concepts are listed in the scheme. Their generic relationships are the only relationships displayed. To express a compound concept, one has to *assemble* its elemental concepts. This process is called *synthesis*, and the arranged groups of elemental concepts that make up the scheme are called *facets*. A faceted classification scheme consists of a finite set of *facets* where each facet consists of a *terminology*, i.e. a finite set of names, or *terms*, describing a domain from a particular aspect, and a binary relation over this terminology, namely *subsumption*. With a faceted classification scheme, objects are indexed (classified) by associating them with zero, one, or more terms from each facet. For example, to index a book we select from each facet the term that best describes each of the concepts in the title. Thus, a faceted scheme comprises independent sets of terms, corresponding to aspects, or facets, of the domain and the objects are assigned terms from the different facets. This process is more dynamic and it also limits the effects of a possible concept re-organization within the relevant facet. Faceted classification schemes seem to be superior to hierarchical classification schemes with regard to comprehensibility, storage requirements and scalability. They are also better suited for indexing collections that are subject to continuous expansion and change ([104]). Specifically:

- A faceted ontology is *more compact*, thus it requires less storage space. This happens because a faceted ontology does not contain terms for each compound concept.

- A faceted ontology is *more scalable* and can be maintained more easily. The additions/deletions of terms, as well as the structural changes, are easier and can be implemented more efficiently. The addition of a new term in a facet implies that many new dynamic combinations of terms are now available for indexing the objects of the domain. Moreover, to delete (or rename) a term, one only has to delete (or rename) that term from the facet in which it belongs. In contrast, in a hierarchical classification scheme one might have to delete/rename several terms in the tree structure.
- A faceted ontology is *easier to understand*. Since a facet is relatively smaller in size than a hierarchical ontology, the indexer or the user can browse each facet separately in order to understand the ontology. In contrast, understanding the ontology of a Web catalog such as Yahoo! through browsing is practically impossible due to its size. Furthermore, the structure of a hierarchical ontology is "synthetic" as it combines several classication criteria. In contrast, a faceted ontology consists of a set of facets each one structured by one basic criterion. This makes the faceted ontologies easier to understand.

3.2 Problems of Faceted Ontologies

A Web catalog can apparently reap concrete benefits from adopting a faceted ontology. However, these will be discounted by impingements to the *indexing* and *browsing* processes. In a faceted ontology *invalid* combinations (conjunctions) of terms coming from different facets may occur. A combination of terms is considered invalid if it cannot be applied to any of the objects of the domain. For example, in a tourist information application, the combination `Crete` $\wedge$ `WinterSports` (also denoted by `Crete.WinterSports`) would be invalid because there is not enough snow in Crete (here, we use dot to denote the conjunction of the term `Crete` and `WinterSports`). In contrast, the combination `Crete.SummerSports` is certainly valid. We can expect the frequency of this phenomenon to increase with the number of facets [1]. Note that a hierarchical catalog for the tourist application would only contain the term `Crete.SummerSports`, while in a faceted ontology both combinations, `Crete.WinterSports` and `Crete.SummerSports`, would be possible during indexing or browsing.

Being able to infer the validity of a combination in a faceted ontology would be very useful. It can be exploited in the indexing process in order to *aid* the indexer and *prevent* indexing errors. Such an aid is especially important in cases where the indexing is done by several people.

[1] Probably a faceted ontology of a wide domain, such as the ontology of Open Directory, will consist of many facets.

For example, the indexing of Web pages in the Open Directory (which is used by Netscape, Lycos, HotBot and several others search engines) is done by more than 20.000 volunteer human editors (indexers). On the other hand, the inability to infer the valid combinations of terms may give rise to problems in browsing. In a hierarchical ontology one browses until reaching the desired objects. In a faceted ontology, an invalid combination of terms will yield no objects. However if we could infer the valid combinations of terms of a faceted ontology then we would be able to generate navigation trees *on the fly*, which consist of nodes that correspond to valid combinations of terms.

As a first example, assume that the domain of interest is a set of hotel home pages, and suppose that we want to provide access to these pages according to the *location* of the hotels and the *sport facilities* they offer. Figure 3.1.(I). shows a hierarchical ontology. Notice that it consists of nodes that correspond to valid combinations of terms. For instance, there is no node that corresponds to the conjunction of the terms `SeaSports` and `Olympus` (since Olympus is a mountain. Similarly, there is no node that corresponds to the conjunction `WinterSports.Crete` (since there is not enough snow in Crete). A faceted organization would look as in Figure 3.1.(II) and consists of two facets: one for *sports* and the other for *locations*.

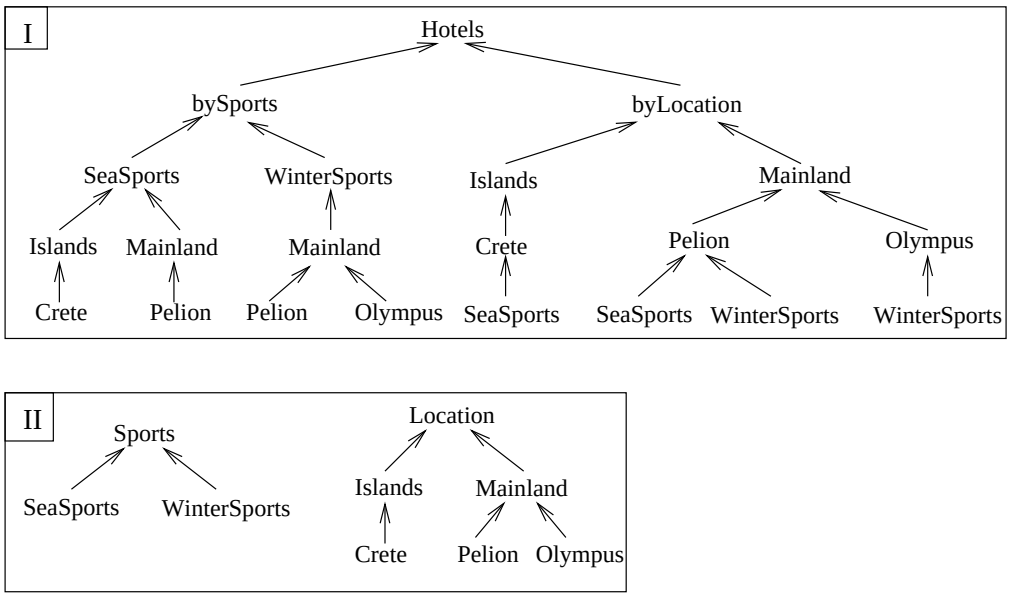


Figure 3.1: Hierarchical vs faceted organization

Let us now compare the organizations (I) and (II): (I) consists of 21 nodes and 20 edges, while (II) consists of 9 nodes and 7 edges. Now suppose that it is decided to delete the term `Mainland`.

Then,

- for updating (I) we have to delete three terms and three relationships and to redirect five relationships, while
- for updating (II) we have to delete one term and one relationship and to redirect two relationships.

Moreover, (I) cannot be considered as being "complete" for supporting browsing because it does not allow the user to *cross facets* while navigating from the root towards the leaves. For example, Figure 3.2 shows that under the term `Mainland` we can create nodes for the terms of the facet Sports. A user who has followed the path `Hotels>ByLocation>Mainland` can now select the desired kind of sports, e.g. `SeaSports`, and subsequently select the desired place in the mainland. Note that here the nodes under the dotted triangle correspond to mainland locations only, so `Crete` does not appear under the node `SeaSports`.

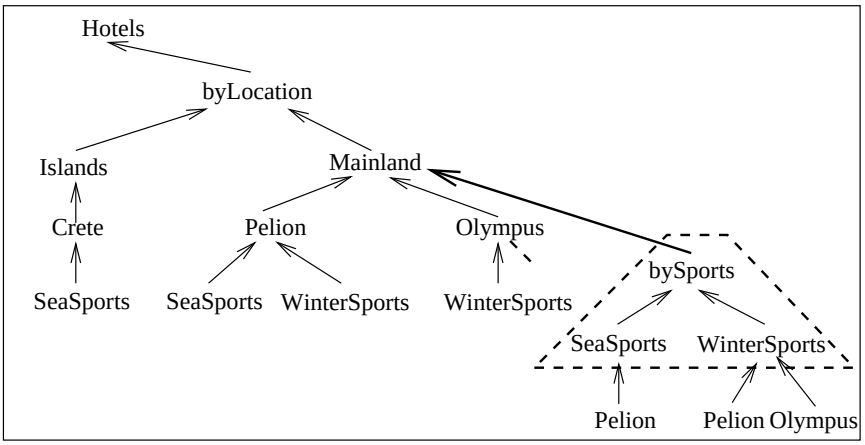


Figure 3.2: Facet Crossing

Now consider the faceted ontology shown in Figure 3.3.(I) which has the additional facet `Accommodation`. Figure 3.3.(II) sketches the corresponding hierarchical organization. We can see that the number of nodes grows exponentially, and notice that this figure does not sketch the nodes needed for supporting "facet crossing".

From the above examples it is evident why the ontologies of existing Web catalogs contain so many terms, and are still not complete (in terms of nodes), especially at the deep levels of their hierarchies. For example, in OpenDirectory under the node

`Artificial Intelligence` > `Knowledge Representation` > `Ontologies` > `People`

there could be a node `byRegion` or a node `byProfessionalStatus`, etc. The reason for this

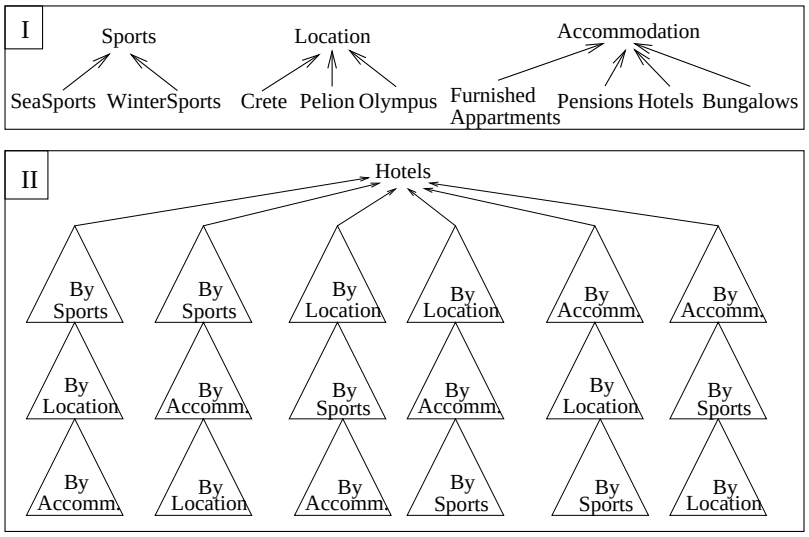


Figure 3.3: An ontology with three facets and a sketch of the corresponding hierarchical ontology

is that a "complete" navigation tree might require millions of nodes. In the extended faceted scheme that we will introduce shortly, we are able to derive complete navigation trees *dynamically* and these trees consist of valid nodes.

3.3 Faceted Ontologies: Definition

A *faceted ontology* is defined by a finite sets of facets. Each *facet* consists of a *terminology*, i.e. a finite set of names or *terms*, structured by a *subsumption* relation. Each facet is designed separately, and it models the domain from a distinct aspect. For instance, the faceted ontology for the domain of UNIX tools, which is presented in [104], consists of four facets, namely, "ByAction", "ByObject", "ByDataStructure" and "BySystem". Each facet consists of a set of terms. For example, the facet "ByDataStructure" consists of the following set of terms: { `buffer, tree, table, file, archive` }. Some other well known faceted ontologies are listed in Table 3.1.

Below we define precisely what a faceted ontology is, and we introduce the notations that will be used in the sequel.

First recall that an *ontology* is a pair $(T, \preceq)$ where T is a *terminology*, i.e. a finite set of names, or *terms*, and $\preceq$ is a *subsumption* relation over T, i.e. a reflexive and transitive relation over T.

Now a faceted ontology is defined as follows:

Def 3.1 A *faceted ontology* is a finite set $\mathcal{F} = \{F_1, ..., F_k\}$ of *ontologies* in which each $F_i = (T_i, \preceq_i)$ is called a *facet*.

Purpose:Source	*Facets*
Document Classification: Ranganatham [106]	Space, Time, Energy, Matter, Personality
Art and Architecture: AAT [62]	Associated Concepts, Physical Attributes, Styles and Periods, Agents, Activities, Materials, Objects
Science and Technology Classification: Vickery [137]	Place, Time, Attributes, Properties, Object, Parts, Processes, Substances, Recipient
Dissertation abstracts: Sugarman [122]	Operation, Object, Properties
Software classification: Diaz [103]	Function, Object, Medium

Table 3.1: Examples of faceted schemes

Without loss of generality we assume that every term of a facet in a faceted ontology $\mathcal{F}$ has a name that is unique within $\mathcal{F}$. If the same term appears in two different facets then we consider the two appearances as two different terms. This is denoted here by subscripting each term of a facet F_i by the subscript i, and can be implemented in practice by, say, prefixing terms by the names of the facets. We will use $\mathcal{T}$ to denote the union of the terminologies of all facets of $\mathcal{F}$, i.e. $\mathcal{T}= \bigcup_{i=1}^{k} T_i$, and $\preceq$ to denote the union of all subsumption relations $\preceq_i$, i.e. $\preceq= \bigcup_{i=1}^{k} \preceq_i$. Clearly the pair $(\mathcal{T}, \preceq)$ is an ontology (according to Def. 2.1). Note that a facet $F_i = (T_i, \preceq_i)$ can be unstructured, that is, $\preceq_i$ consists only of trivial relationships of the form $t \preceq_i t$. In such a case the facet is just a "flat" terminology. For instance, the faceted ontology in [104] consists of flat facets.

Facets are not arbitrary partitions of an underlying ontology. Rather, they are designed separately, effectively modeling different aspects of a common underlying domain. Consequently, the description of an object by means of a faceted ontology is achieved by associating the object with a conjunction of terms from different facets.

We do *not* focus on facet analysis, i.e. which facets should be selected and how they should be constructed. This process can be carried out either formally (see for example [78], [137], [36], or Section 3.8 for more), or informally, as it is usually done by the designers of Web catalogs. Moreover, we do *not* focus on assigning descriptions to objects, but rather in inferring valid or invalid descriptions. Here by *description* we mean just a conjunction of terms.

Now, a combination of terms, or *description*, is defined as follows:

Def 3.2 A *description* d over $\mathcal{F}$ is either a term $t \in \mathcal{T}$ or a sequence of terms separated by "$\cdot$", i.e. any string derived by the following grammar $d ::= d \cdot t \mid t$

In Figure 3.3, for example, the following string is a description: `SeaSports.Crete.Pension`.

The order of terms in a description does not make any difference. Hereafter we will use $D_{\mathcal{F}}$, or simply D, to denote the set of all descriptions over a faceted ontology $\mathcal{F}$. Note that D is an infinite set, even if T contains just a single term. Indeed t, $t.t$, $t.t.t$, and so on, are descriptions over T. However, the set of equivalence classes of D is finite. Recall that two descriptions d and d' are *equivalent in* an ontology $(T, \preceq)$, if $I(d) = I(d')$ is every model I of $(T, \preceq)$.

Now, the problem with descriptions is that $\mathcal{F}$ does not itself specify which descriptions are valid and which are not. One approach to overcome this problem, would be to

(a) first enumerate all descriptions (specifically, all equivalence classes of descriptions),

(b) then partition this set into two disjoint subsets: one containing all valid descriptions and the other containing all invalid descriptions,

(c) and finally store one of the sets (either the set of valid, or the set of invalid descriptions).

For example, for the faceted ontology of Figure 3.1, we could define:

Valid Descriptions	:	`Sports.Location`, `SeaSports.Location`, `WinterSports.Location`, `Sports.Islands`, `SeaSports.Islands`, `Sports.Mainland`, `SeaSports.Mainland`, `WinterSports.Mainland`, `Sports.Crete`, `SeaSports.Crete`, `Sports.Pelion`, `SeaSports.Pelion`, `WinterSports.Pelion`, `Sports.Olympus`, `WinterSports.Olympus`
Invalid Descriptions	:	`WinterSports.Islands`, `WinterSports.Crete`, `SeaSports.Olympus`

Notice that in this example we considered descriptions which consist of exactly one term from each facet. Defining manually these sets even for facets of relatively small size, would be a formidable task. Even in the special case where a description consists of exactly one term from each facet, there are $|T_1| * ... * |T_k|$ different combinations of terms, e.g. if $k = 5$ and $|T_i| = 10$ for each $i = 1..k$, there are 100.000 different combinations. Moreover, this approach has prohibitive storage space requirements.

In what follows, we first give a semantic interpretation to $\mathcal{F}$ and then (in section 3.4) we define the extended faceted ontologies and we exploit this semantic interpretation in order to infer the validity or invalidity of a description.

We employ the semantic interpretation that we gave in Chapter 2, that is:

(a) we consider a denumerable set of objects Obj,

(b) an interpretation of a terminology T over Obj is any function $I : T \rightarrow 2^{Obj}$, and

(c) an interpretation I of T is a model of an ontology $(T, \preceq)$ if for all $t, t' \in T$, if $t \preceq t'$ then $I(t) \subseteq I(t')$.

Def 3.3 Given a faceted ontology $\mathcal{F}$, we call interpretation of $\mathcal{F}$ any interpretation of the terminology $\mathcal{T}$.

Recall that if $\mathcal{F}$= $\{F_1, ..., F_k\}$ then $\mathcal{T}$= $T_1 \cup ... \cup T_k$ and $\preceq$=$\preceq_1 \cup ... \cup \preceq_k$. We shall call $(\mathcal{T}, \preceq)$ *the* ontology of $\mathcal{F}$.

Def 3.4 An interpretation I of $\mathcal{T}$ is a *model* of $\mathcal{F}$, if it is a model of the ontology $(\mathcal{T}, \preceq)$.

Note that if I is a model of $\mathcal{F}$ then the restriction of I on T_i, denoted $I_{|T_i}$, is a model of the facet F_i for each $i = 1, .., k$.

An interpretation I of $\mathcal{T}$ can be extended to an interpretation of D as follows: for any description $d = t_1 \cdot t_2 \cdot .. \cdot t_k$ in D we define $I(d) = I(t_1) \cap I(t_2) \cap ... \cap I(t_k)$.

Now, we define what we shall call valid and invalid description. A description d is *valid* in $\mathcal{F}$ if $I(d) \neq \emptyset$ in every model I of $\mathcal{F}$. A description d is *invalid* in $\mathcal{F}$ if $I(d) = \emptyset$ in every model I of $\mathcal{F}$. We shall use the symbols VD and ID to denote the set of all valid descriptions and all invalid descriptions, respectively, that is:

$$\begin{aligned} VD &= \{d \in D \mid I(d) \neq \emptyset \text{ in every model } I \text{ of } \mathcal{F}\} \\ ID &= \{d \in D \mid I(d) = \emptyset \text{ in every model } I \text{ of } \mathcal{F}\} \end{aligned}$$

However both sets turn out to be empty, i.e. $VD = ID = \emptyset$, because for any description d we can construct a model I such that $I(d) \neq \emptyset$, and a model I' such that $I'(d) = \emptyset$. For example, consider two intepretations I and I' such that $I(t) = \{1\}$ and $I'(t) = \emptyset$ for each $t \in \mathcal{T}$. Clearly, both I and I' are models of $\mathcal{F}$. Notice that for each $d \in D$, $I(d) \neq \emptyset$, while $I'(d) = \emptyset$.

In the following section we propose two different extensions of an ontology that allows us to infer valid or invalid descriptions from other descriptions that have been declared as valid or invalid by the designer of the faceted ontology.

3.4 Establishing Description Validity by Ontology Extensions

In this section, given a faceted ontology $\mathcal{F}$, we introduce two extensions of $\mathcal{F}$.

In the first extension we consider $\mathcal{F}$ together with a given set of descriptions that the designer considers *valid*. Using this set our inference mechanism will infer *all* valid descriptions. In this

extension, a description is considered invalid if it is not valid. Note that, in doing so, we adopt a closed world assumption.

In the second extension we consider $\mathcal{F}$ together with a given set of descriptions that the designer considers *invalid*. Using this set our inference mechanism will infer *all* invalid descriptions. In this extension, a description is considered valid if it is not invalid. Note that, in doing so, we again adopt a closed world assumption.

Def 3.5 A *Positive Extended Faceted Ontology*, or *PEFO* for short, is a pair $< \mathcal{F}, P >$ where $\mathcal{F}$ is a faceted ontology and P is a set of descriptions over $\mathcal{T}$. An interpretation I of $\mathcal{T}$ is a *model* of $< \mathcal{F}, P >$ if:

(a) I is a model of $\mathcal{F}$, and

(b) for each $d \in P$, $I(d) \neq \emptyset$.

Now, the set of valid descriptions VD of a *PEFO* $< \mathcal{F}, P >$ is defined as follows:

$$VD = \{d \in D \mid I(d) \neq \emptyset \text{ in every model } I \text{ of } <\mathcal{F}, P>\}$$

If a description is not an element of the set VD, then it is considered invalid, i.e. $ID = D \setminus VD$.

Lemma 3.1 If $d' \in P$ then for every description d such that $\mathcal{F} \models d' \preceq d$, it holds $I(d) \neq \emptyset$ in every model I of $< \mathcal{F}, P >$.

If we write $< \mathcal{F}, P > \models d' \preceq d$ whenever $I(d') \subseteq I(d)$ in every model I of $< \mathcal{F}, P >$, then we can proceed to the following lemma:

Lemma 3.2

$$\begin{aligned} VD &= P \cup \{d \in D \mid \exists\, d' \in P \text{ such that } <\mathcal{F}, P> \models d' \preceq d\} \\ &= P \cup \{d \in D \mid \exists\, d' \in P \text{ such that } \mathcal{F} \models d' \preceq d\} \end{aligned}$$

This implies that the designer does not have to specify all the valid descriptions. He only provides some of them and from these other valid descriptions can be inferred. For example, if `Crete.Seasports` $\in P$ then the description `Islands.Seasports` and `Crete.Sports` are valid too. We can also easily see that if a description in P consists of m terms, then all descriptions that can be formed by using a subset of these terms are valid too, e.g. if `Crete.SeaSports.Bungalows` $\in P$ then the descriptions `Crete.SeaSports`, `Crete.Bungalows`, and `SeaSports.Bungalows`, are valid too.

All atomic descriptions, i.e. all descriptions that consist of only one term, are always valid, therefore $\mathcal{T}$ should be a subset of P. We consider this by default, thus the designer does not have to explicitly include the elements of $\mathcal{T}$ in P.

Let us now introduce the Negative Extended Faceted Ontologies.

Def 3.6 A *Negative Extended Faceted Ontology*, or *NEFO* for short, is a pair $< \mathcal{F}, N >$ where $\mathcal{F}$ is a faceted ontology and N is a set of descriptions over $\mathcal{T}$. An interpretation I of $\mathcal{F}$ is a *model* of $< \mathcal{F}, N >$ if:

(a) I is a model of $\mathcal{F}$, and

(b) for each $d \in N$, $I(d) = \emptyset$.

Now, the set of invalid descriptions ID of a *NEFO* $< \mathcal{F}, N >$ is defined as follows:

$$ID = \{d \in D \mid I(d) = \emptyset \text{ in every model } I \text{ of } < \mathcal{F}, N >\}$$

If a description is not an element of the set ID, then it is considered valid, i.e. $VD = D \setminus ID$.

Lemma 3.3 If $d' \in N$ then for every description d such that $\mathcal{F} \models d \preceq d'$, it holds $I(d) = \emptyset$ in every model I of $< \mathcal{F}, N >$.

If we write $< \mathcal{F}, N > \models d \preceq d'$ whenever $I(d) \subseteq I(d')$ in every model I of $< \mathcal{F}, N >$, then we can proceed to the following lemma:

Lemma 3.4

$$\begin{aligned} ID &= N \cup \{d \in D \mid \exists\, d' \in N \text{ such that } < \mathcal{F}, N > \models d \preceq d'\} \\ &= N \cup \{d \in D \mid \exists\, d' \in N \text{ such that } \mathcal{F} \models d \preceq d'\} \end{aligned}$$

This implies that from the set N of invalid descriptions, other invalid descriptions can be inferred. For instance, if $t.t' \in N$ then every description that contains the term t, or a term subsumed by t, and the term t', or a term subsumed by t', is invalid too. For example, if `Crete.WinterSports` $\in N$ then this implies that the description `Crete.WinterSports.Hotel` is invalid too, as well as the description `Crete.SnowBoarding` (assuming that `SnowBoarding` $\preceq$ `WinterSports`).

3.4.1 Choosing between *PEFO* and *NEFO* Extension

The designer can employ a *PEFO* or a *NEFO* extension depending on the faceted ontology of the application. If the majority of the descriptions are valid (e.g. see Figure 3.4.(a)), then it is better to employ a *NEFO*, so as to specify only the invalid descriptions. Conversely, if the majority of the descriptions are invalid (e.g. see Figure 3.4.(b)), then it is better to employ a *PEFO* so as to specify only the valid descriptions.

Concerning the methodology for defining the set N, it is more efficient for the designer to put in N "short" descriptions that consist of "broad" terms. The reason is that from such descriptions a large number of new invalid descriptions can be inferred. For example in the hypothetical case

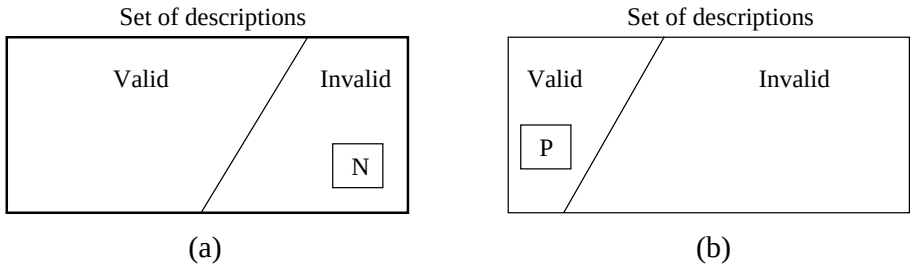


Figure 3.4: Choosing between *PEFO* and *NEFO* extension

that we want to specify that all descriptions over the faceted ontology of Figure 3.1 are invalid, it suffices to put in N one description, i.e. the description `Sports.Location`.

Concerning the methodology for defining the set P, it is more efficient for the designer to put in P "long" descriptions that consist of "narrow" terms, since from such descriptions a large number of new valid descriptions can be inferred. For example in the hypothetical case that we want to specify that all descriptions over the faceted ontology of Figure 3.1 are valid, it suffices to put in P just the following description: `SeaSports.WinterSports.Crete.Pelion.Olympus`.

Figure 3.5 shows how we can specify the valid/invalid descriptions of the faceted ontology of Figure 3.1 (i.e. the sets "Valid Descriptions" and "Invalid Descriptions" as presented in Section 3.3) by employing a *PEFO* or a *NEFO* extension.

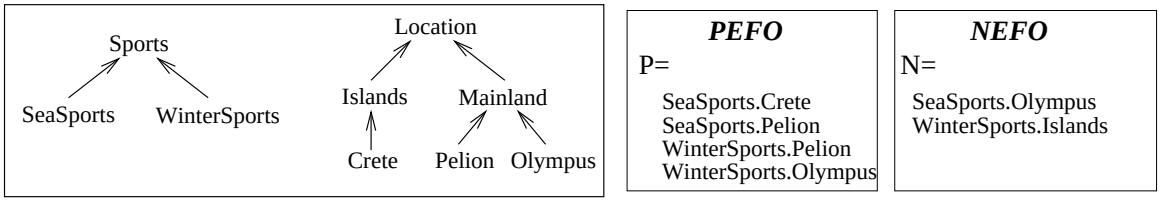


Figure 3.5: Example of a *PEFO* and a *NEFO* for the same domain

3.5 Inference Mechanisms for Deciding the Validity of Descriptions

In Section 3.4 we introduced two ontology extensions and we gave a semantic interpretation to each one of them. In this section we describe the inference mechanism needed in each of these two cases in order to check whether a description is valid or invalid.

Let $<\mathcal{F},P>$ be a *PEFO*. A description d is valid in $<\mathcal{F},P>$ if there is a description $p \in P$ such that $<\mathcal{F},P> \models p \preceq d$. Thus for checking whether d is valid we can check whether there is

a $p \in P$ such that $< \mathcal{F}, P > \models p \preceq d$. If there is such a p then certainly the description d is valid, otherwise it is invalid. Thus for checking the validity of a description we need to perform $|P|$ inclusion checks. For checking inclusions we can exploit the inference mechanism described in Section 2.8. As an inclusion check can be performed in $O(|\preceq| * k)$ time (where k is the maximum number of terms that appear in the involved descriptions), the validity of a description can be checked in $O(\mid P \mid * \mid \preceq \mid * k)$ time.

An alternative inference procedure is described below. Specifically we look for a special model, denoted by m, such that $m(d) \neq \emptyset$ iff $I(d) \neq \emptyset$ in every model I of $< \mathcal{F}, P >$. Below we provide an algorithm (Algorithm 3.1) which takes as input a pair $< \mathcal{F}, P >$ and returns the model m. In this algorithm, we assume a function *witness* that takes as argument a description d in P and returns a set containing a single object from Obj. We assume that *witness* is injective over the set of descriptions. The main idea is the following: In Step 1 we construct an interpretation m so that every description in P has non empty interpretation. Then, in Steps 2 and 3, we "enlarge" this interpretation to a model of $\mathcal{F}$. Clearly, the resulting interpretation is a model of $< \mathcal{F}, P >$.

Algorithm 3.1
Input: A *PEFO* $< \mathcal{F}, P >$
Output: The model m of $< \mathcal{F}, P >$

Step 1: For each $d = t_1 \cdot ... \cdot t_m \in P$
  If $m(d) = \emptyset$ then
    $m(t_1) := m(t_1) \cup witness(d)$
    ...
    $m(t_m) := m(t_m) \cup witness(d)$
  EndIf
Step 2: For each $t \preceq t'$
  If $m(t) \not\subseteq m(t')$ then
    $m(t') := m(t) \cup m(t')$
Step 3: If changes in Step 2 then
  Goto Step 2
  else return m

The table that follows shows the model constructed by this algorithm for the *PEFO* extension shown in Figure 3.5. As mentioned earlier, we assume that P by default *includes* the set of atomic terms $\mathcal{T}$. This means that the set P in this example consists of the 9 terms of $\mathcal{T}$ plus the 4 descriptions of P that are shown in Figure 3.5. The objects which are returned by the

function *witness* are represented by natural numbers.

$t \in \mathcal{T}$	$m(t)$ after Step 1	$m(t)$ returned
`Sports`	{1}	{1,2,3,10,11,12,13 }
`SeaSports`	{2,10,11}	{2,10,11 }
`WinterSports`	{3,12,13}	{3,12,13 }
`Location`	{4}	{4,5,6,7,8,9,10,11,12,13 }
`Islands`	{5}	{5,7,10 }
`Mainland`	{6}	{6,8,9,11,12,13 }
`Crete`	{7,10}	{7,10 }
`Pelion`	{8,11,12}	{8,11,12 }
`Olympus`	{9,13}	{9,13}

Prop. 3.1 The Algorithm 3.1 produces a model of $< \mathcal{F}, P >$.
Proof:

For each $i = 1..k$ the restriction of m on T_i is a model of F_i. This holds because if $t \preceq t'$ then $m(t) \subseteq m(t')$ due to steps 2 and 3. Moreover, due to Step 1, for each $d \in P$ holds $m(d) \neq \emptyset$. From these we conclude that m is a model of $< \mathcal{F}, P >$.

$\diamond$

Prop. 3.2 $m(d) \neq \emptyset$ iff $I(d) \neq \emptyset$ in every model I of $< \mathcal{F}, P >$.
Proof:
(If $I(d) \neq \emptyset$ in every model I of $< \mathcal{F}, P >$ then $m(d) \neq \emptyset$)

This holds since m is a model of $< \mathcal{F}, P >$.

(If $m(d) \neq \emptyset$ then $I(d) \neq \emptyset$ in every model I of $< \mathcal{F}, P >$)

Below we prove that the descriptions that are valid in m are valid in every model I of $< \mathcal{F}, P >$.

First we introduce some notations. Let G denote the $PEFO < \mathcal{F}, P >$, $V(G)$ denote all descriptions that are valid in G, and $V(m)$ denote all descriptions that are valid in the model m which is produced by the algorithm. Given a term t, $Prod(t)$ denotes all descriptions which consist of one or more repetitions of t, i.e. t, $t \cdot t$, ... etc. Clearly, if $d \in Prod(t)$ then $I(d) = I(t)$ in any interpretation I. Given a set of terms $S \subseteq \mathcal{T}$, by $Prod(S)$ we denote all descriptions consisting of one or more repetitions of each term in S. For example $Prod(\{t, t'\})$ contains descriptions of the form: $t \cdot t'$, $t \cdot t' \cdot t$, $t' \cdot t' \cdot t$, ... etc. Clearly, if $S = \{t_1, ..., t_k\}$, and $d \in Prod(S)$, then $I(d) = I(t_1 \cdot ... \cdot t_k)$ in any interpretation I.

Below we prove that $V(m) \subseteq V(G)$ by showing that the descriptions that are valid in m, as it is produced in each step of the algorithm, are valid in every model of G too.

Step 1:

Let m_{S1} be the interpretation produced by Step 1 of the algorithm. Below we prove that $V(m_{S1}) \subseteq V(G)$.

Below we prove by induction that $V(m_{S1})$ includes the descriptions $\{Prod(terms(p)) \mid p \in P\}$, where by $terms(p)$ we denote the terms that appear in the description p. This means that we will prove that $V(m_{S1}) = \{Prod(terms(p)) \mid p \in P\}$.

Let $m_0, m_1, ...$ denote the interpretations produced by each iteration of the loop in Step 1, where m_0 is the interpretation produced by Step 1, that is $m_0 = m_{S1}$, and m_{j+1} is the interpretation produced by m_j after one iteration.

Let m_j be an interpretation in which $V(m_j)$ consists of descriptions of the form described above. Let m_{j+1} be the interpretation after one iteration of the loop in Step 1, and let d be a description such that $d \in V(m_{j+1})$, but $d \notin V(m_j)$. Let the difference between m_{j+1} and m_j be the following pair of statements:

$$\begin{aligned} m_{j+1}(t) &= m_j(t) \cup witness(t \cdot t') \\ m_{j+1}(t') &= m_j(t') \cup witness(t \cdot t') \end{aligned}$$

due to a description $p = t \cdot t' \in P$ and the fact that $m_j(t) \cap m_j(t') = \emptyset$.

Certainly the description d should contain at least one (or both) of the terms t, t'. Let us assume that d contains only the term t, that is we may write $d = t \cdot x$ where x is a description which does not contain t or t'. We have:

$$\begin{aligned} m_{j+1}(d) &= m_{j+1}(t) \cap m_j(x) \\ &= (m_j(t) \cup witness(t \cdot t')) \cap m_j(x) \\ &= (m_j(t) \cap m_j(x)) \cup (m_j(x) \cap witness(t \cdot t')) \\ &= m_j(d) \cup (m_j(x) \cap witness(t \cdot t')) \\ &= m_j(x) \cap witness(t \cdot t') \end{aligned}$$

Since $m_{j+1}(d) \neq \emptyset$, it must be $m_j(x) \cap witness(t \cdot t') \neq \emptyset$, that is, the object $witness(t \cdot t')$ must be an element of $m_j(x)$. But since the function "witness" is injective, the object $witness(t \cdot t')$ belongs only to $m_{j+1}(t)$ and $m_{j+1}(t')$. Thus, x should consist of repetitions of t or t', and x should not contain any other term. Thus the hypothesis that x does not contain the terms t or t' is false. Hence, we can write $V(m_{S1}) = \cup\{Prod(terms(p)) \mid p \in P\}$.

Now let I be a model of G. If $t \cdot t' \in P$ then certainly $I(t \cdot t') \neq \emptyset$, and clearly $I(d) \neq \emptyset$ for every $d \in Prod(terms(p))$ since $m(d) = m(t \cdot t')$. Hence we conclude that $V(m_{S1}) \subseteq V(G)$.

Step 2:

Below we prove that $V(m_{S2}) \subseteq V(G)$ where m_{S2} is the interpretation produced by Step 2.

Let $m_0, m_1, ...$ denote the interpretations produced by each iteration of the loop in Step 2, where m_0 is the interpretation produced by Step 1, that is $m_0 = m_{S1}$, and m_{j+1} is the interpretation produced by m_j after one iteration. We prove that $V(m_{S2}) \subseteq V(G)$ by induction, that is, we prove that if $V(m_j) \subseteq V(G)$ then $V(m_{j+1}) \subseteq V(G)$.

Let d be a description such that $d \in V(m_{j+1})$, but $d \notin V(m_j)$. Let the difference between m_{j+1} and m_j be the following statement:

$$m_{j+1}(t') = m_j(t) \cup m_j(t')$$

due to a relationship $t \preceq t'$ and the fact that $m_j(t) \not\subseteq m_j(t')$.

Certainly the description d should contain the term t', that is we can write $d = t' \cdot x$ where x is a description which does not contain t'. We have:

$$\begin{aligned} m_{j+1}(d) &= m_{j+1}(t') \cap m_j(x) \\ &= (m_j(t') \cup m_j(t)) \cap m_j(x) \\ &= (m_j(t') \cap m_j(x)) \cup (m_j(t) \cap m_j(x)) \\ &= m_j(d) \cup (m_j(t) \cap m_j(x)) \\ &= m_j(t) \cap m_j(x) \end{aligned}$$

Let I be a model of G. Since $m_{j+1}(d) \neq \emptyset$, we have $m_j(t) \cap m_j(x) \neq \emptyset$ too. But since $V(m_j) \subseteq V(G)$ this means that $I(t) \cap I(x) \neq \emptyset$. Due to $t \preceq t'$ it must be $I(t) \subseteq I(t')$. Combining these two facts we obtain $I(t') \cap I(x) \neq \emptyset$. Hence $V(m_{S4}) \subseteq V(G)$.

Thus we proved that $V(m) \subseteq V(G)$. Moreover, since m is a model of G, we have $V(G) \subseteq V(m)$. Hence we conclude that $V(m) = V(G)$.

$\diamond$

Thus for checking the validity of a description we run Algorithm 3.1 once, which produces the model m and then it suffices to check the validity of descriptions in this model. Thus for checking the validity of a description $d = t_1 \cdot ... \cdot t_k$ we have to compute $m(t_1 \cdot ... \cdot t_k)$, i.e. we have to perform k intersections of sets. As the Algorithm 3.1 creates at most $|T| + |P|$ objects, an intersection can be perfomed in $O(|T| + |P|)$ time. Thus $m(d)$ can be computed in $O(k * (|T| + |P|))$ time. Clearly, this method is more efficient than performing $|P|$ inclusion checks. Recall that the algorithm that peforms $|P|$ inclusion checks has execution time in $O(\mid P \mid * \mid \preceq \mid * k)$.

Now, let $<\mathcal{F},N>$ be a *NEFO*. A description d is *invalid* in $<\mathcal{F},N>$ if there is an $n \in N$ such that $<\mathcal{F},N> \models d \preceq n$. If there is such n then certainly the description d is invalid, otherwise it is valid. Thus for checking whether d is invalid one can check whether there is an $n \in N$ such that $<\mathcal{F}, N> \models d \preceq n$, thus we need to perform $|N|$ inclusion checks. For checking inclusions we can exploit the inference mechanism described in Section 2.8. As an inclusion check can be performed in $O(|\preceq| * k)$ time (where k is the maximum number of terms that appear in the involved descriptions), the validity of a description can be checked in $O(\mid N \mid * \mid \preceq \mid * k)$ time.

For the *NEFO* extensions we have not yet managed to find an inference procedure that is based on a single special model m. This is an issue for further research.

3.6 Generating Navigation Trees

Faceted ontologies cannot be browsed easily especially when they consist of many facets. For browsing a faceted ontology one would have to select one or more terms from each facet (and some facets may contain many terms) in order to formulate a description that reflects one's information need. However many of the resulting descriptions might be meaningless, i.e. invalid, thus yielding no objects. Therefore we would like a *navigation tree* that is, a tree with nodes that correspond to only valid descriptions. The tree should have nodes that enable the user to start browsing in one facet and then *cross* to another, and so on, until reaching the desired level of specificity. It is important to note that the same navigation tree can be used by the indexer during the indexing of the objects of the domain. This tree can certainly aid the indexer since it can speed up the indexing process and prevent indexing errors.

Let us first introduce some notations before presenting the algorithm for generating the navigation tree. Given a description $d = t_1 \cdot ... \cdot t_k$ we denote by $head(d)$ the union of the broader terms of all terms that appear in d, i.e. $head(t_1 \cdot ... \cdot t_k) = head(t_1) \cup ... \cup head(t_k)$. By $head(d)/\sim$ we denote the set of equivalence classes of the set $head(d)$. For brevity hereafter we shall use $head(d)$ to denote $head(d)/\sim$.

A *navigation tree* is a directed acyclic graph (N, R) where N is the set of *nodes* and R is the set of *edges*. Let $\mathcal{G}$ be an extended faceted ontology (either *PEFO* or *NEFO*). The navigation tree that we construct for $\mathcal{G}$ has the following property:

> for each valid description $d \in VD$,
> the navigation tree has a path (starting from the root)
> for each *topological sort* of the nodes
> of the directed acyclic graph $(head(d), \preceq_{|head(d)})$.

For example consider the faceted taxonomy shown in Figure 3.5, and suppose that `Crete.SeaSports` $\in VD$. The navigation tree in this case will include the following paths:

```
Location > Greece > Crete > Sports > SeaSports
Location > Greece > Sports > Crete > SeaSports
Location > Greece > Sports > SeaSports > Crete
Location > Sports > Greece > SeaSports > Crete
...
...
Sports > SeaSports > Location > Greece > Crete
```

Moreover, and in order to further aid the user, whenever we have facet crossing a new node

is created which presents the name of the facet (specifically its top term prefixed by the string "By") that we are crossing to.

There are two approaches to deriving the navigation tree. The first approach is to generate a "complete" static navigation tree, i.e. a tree that includes nodes for all valid descriptions and for this purpose we need an algorithm that takes as input a $\mathcal{G}$ and returns a navigation tree [2]. The second approach is to design a mechanism that generates the navigation tree *on the fly*, i.e during browsing. Prior to presenting our algorithm let us first define precisely what a navigation tree is.

Without loss of generality below we assume that each facet F_i has a greatest term from which all terms of the facet are "hanged" and we will denote this term by $top(F_i)$. Specifically, each node n of the navigation tree (N, R) has:

- a *description*, denoted by $D(n)$.

  As we shall see below, we construct navigation trees with nodes whose descriptions are valid.

- a *focus term*, denoted by $Fc(n)$.

  The focus term of a node n is a distinguished term among those that appear in the description $D(n)$ of the node.

- a *name*, denoted by $Nm(n)$.

  The name of a node is used for presenting the node at the user interface. It coincides with the focus term of n, unless n is a node for facet crossing. In the latter case the name of n is the name of the top term of the facet we are crossing to, prefixed by the string "By".

Below we describe an algorithm which takes as input a $\mathcal{G}$ and returns a navigation tree (N, R). Roughly, the navigation tree is constructed as follows: At first we create a node for the greatest element of each facet. Specifically for each facet F_i we create a node with description the greatest element of F_i i.e. $top(F_i)$; we set as name and focus term of each such node the term $top(F_i)$ too. Now, for each node n we create *two* groups of children. The descriptions of the nodes in the first group are the result of replacing the focus term of n (i.e. $Fc(n)$) by an immediately narrower term of $Fc(n)$, while the second group consists of nodes for facet crossing.

Instead of presenting the algorithm for constructing the entire navigation tree, in Algorithm 3.2 we present the first step, i.e. the creation of a node for each facet, and the steps for creating the children of a node. These steps can be synthesized to get an algorithm that constructs the entire navigation tree (given next). The algorithm uses a function `IsValid`(< *description* >)

[2] In the domain of the Web the navigation tree would be a set of inter-linked Web pages.

which returns `True` if its argument is a valid description and `False` otherwise. The command `addChild` ($< name >$,$< descr >$,$< focusterm >$) creates a new child of the current node. In the presentation of the algorithm we adopt the notation described next.

We denote each facet F_i of a faceted ontology $\mathcal{F}$= $\{F_1, ..., F_k\}$ by the integer i. For example for the faceted ontology $\mathcal{F}$= {`Sports`, `Location`} shown in Figure 3.5, the facet `Sports` is denoted by the integer 1 and the facet `Location` is denoted by the integer 2. Now, by $f(t)$ we denote the facet in which the term t belongs, e.g. f(`SeaSports`) = 1 and f(`Crete`) = 2. By $\Pi_i(d)$ we denote the subterm of d that belongs to facet i, or the empty string if there is no such subterm, e.g. Π_1(`SeaSports.Crete`) = `SeaSports`, Π_2(`SeaSports.Crete`) = `Crete`, Π_1(`Crete`) = "". By $top(i)$ we denote the greatest term of facet i. By $Nar(t)$ we denote the set of immediately narrower terms of t, e.g. Nar(`Location`) = {`Islands`, `Mainland`}.

Figure 3.6 shows a part of the navigation tree that is generated by this algorithm for the ontology of Figure 3.5. In this figure, each node n is presented by its name, $Nm(n)$. As an example, the node n_{22} has $Nm(n_{22})$ = `Mainland`, $D(n_{22})$ = `Sports.Mainland`, $Fc(n_{22})$ = `Mainland`. The nodes n_{23} and n_{27} are generated by the part B.1 of the algorithm, while the node n_{30} is generated by the part B.2.

Algorithm 3.2 Navigation Tree (short version)
Input: An extended faceted ontology $\mathcal{G}$
Output: A navigation tree (N, R)

```
Part A  // Creation of one node for each facet
        For each i = 1..k
            addChild( top(i), top(i), top(i))      // addChild(name, descr, focuterm)

Part B  // Creating the children of a node n
B.1     // Creating the children of a node on the basis of the focus term
        NarF := Nar(Fc(n))
        For each t ∈ NarF
            If IsValid(D(n).t) then
                addChild( t, D(n).t, t)

B.2     // Creating the children of a node for "facet crossing"
        For each i = 1..k and i ≠ f(Fc(n))
            If Π_i(D(n)) = "" then          // the description of n does not contain any term in F_i
                If IsValid(D(n).top(i)) then
                    addChild("by" + top(i), D(n).top(i),top(i))
            Else
                If ∃t' ∈ Nar(Π_i(D(n)) such that IsValid(D(n).t') then
                    addChild("by" + top(i), D(n), Π_i(D(n)))
```

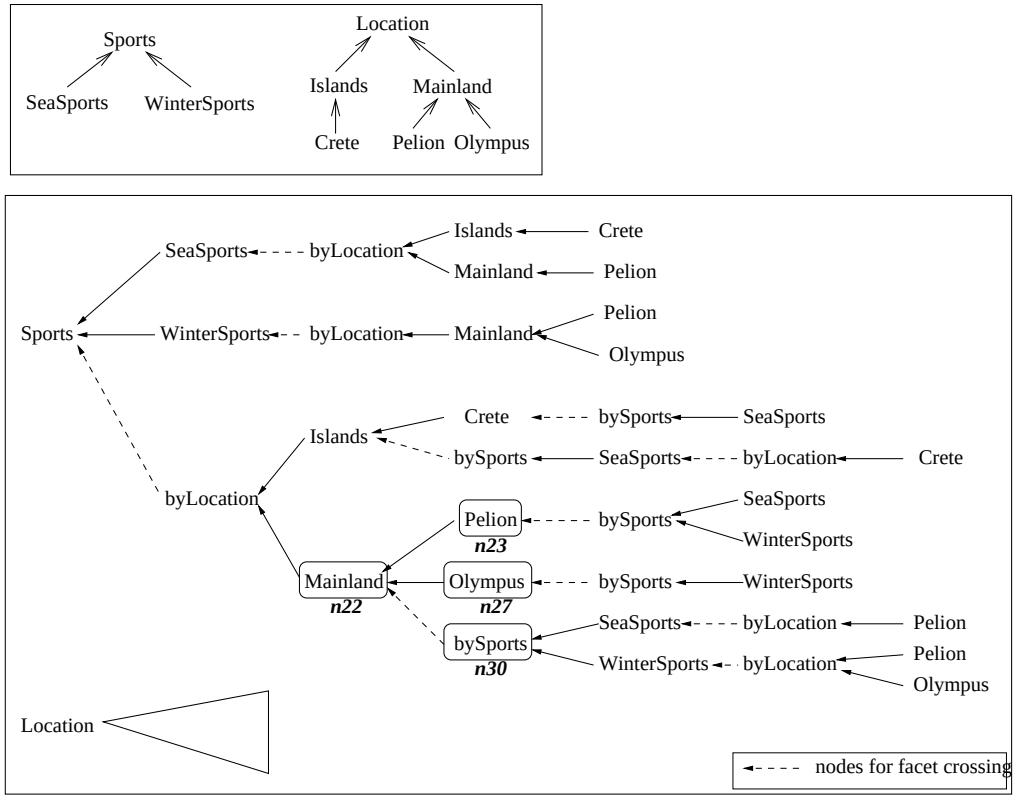


Figure 3.6: Example of a dynamically generated navigation tree

Algorithm 3.2. Navigation Tree (complete version)
Input: An extended faceted ontology $\mathcal{G}$
Output: A navigation tree (N, R)

```
subroutine CreateTree(n: Node)
        if Nm(n)="Top" then
                For each i = 1..k
                    n' := CreateNode( top(i), top(i), top(i))      // CreateNode(name, description, focuterm)
                    AddChild(n, n')                                // makes node n' child of n
                    CreateTree(n')
                End For
        Else
                NarF := Nar(Fc(n))
                For each t ∈ NarF
                    If IsValid(D(n).t) then
                        n' := CreateNode( t, D(n).t, t)
                        AddChild(n, n')
                        CreateTree(n')
                    End If
                End For
                For each i = 1..k and i ≠ f(Fc(n))
                    If Π_i(D(n)) = "" then
                        If IsValid(D(n).top(i)) then
                            n' := CreateNode("by" + top(i), D(n).top(i),top(i))
                            AddChild(n, n')
                            CreateTree(n')
                        End If
                    Else
                        If ∃t' ∈ Nar(Π_i(D(n)) such that IsValid(D(n).t') then
                            n' := CreateNode("by" + top(i), D(n), Π_i(D(n)))
                            AddChild(n, n')
                            CreateTree(n')
                        End If
                    End Else
                End For
End CreateTree


BEGIN
        topNode := CreateNode( "Top", "", "")              // creation of the top node
        CreateTree(topNode)
        return topNode
END
```

3.7 Building a Source using an Extended Faceted Ontology

Building a source using an extended faceted ontology (*PEFO* or *NEFO*) implies the constraint that each object should be associated with a valid description, i.e. for each object o it must be $D_I(o) \in VD$.

However, if for indexing an object o, i.e. for associating a description d to o, the indexer selects a description from the dynamically derivable navigation tree, then the validity of the descriptions of the objects is ensured. In this case, we do not need any additional mechanism for checking the validity of descriptions.

3.8 Discussion

Below we discuss the classifications schemes in library and information science, facet analysis, and object-oriented conceptual schemas, in order to describe what is currently used for information indexing. At last, we investigate whether we can represent the extended faceted ontologies in some logic-based languages.

3.8.1 Library and Information Science

In Library and Information Science we can identify three approaches to classification: *enumerative, hierarchical* and *analytico-synthetic* (or faceted). *Enumerative classification* attempts to assign headings for every subject and alphabetically enumerates them. *Hierarchical classification* establishes logical rules for dividing topics into classes, divisions and subdivisions. *Analytico-synthetic* (or *faceted*) classification assigns terms to individual concepts and provides rules for the local cataloguer to use in constructing headings for composite subjects. Ranganathan was the first to introduce the word "facet" into library and information science in 1933. Ranganathan demonstrated that *analysis*, which is the process of breaking down subjects into their elemental concepts, and *synthesis*, the process of recombining those concepts into subject strings, could be applied to all subjects, and demonstrated that this process could be systematized. The phrase "analytico-synthetic classification" derives from these two processes: analysis and synthesis. For a wider review on the faceted access to information see [82].

3.8.2 Facet Analysis

As commented in [36], the basic principle of facet analysis is that concepts can be grouped using a characteristic of division which is not necessarily hierarchical. In other words, subjects which have previously been sub-divided by progressive hierarchical arrangement, forming the familiar

"tree structures" of conventional indexing theory, can be looked on as patterns of horizontal division as well as vertical divisions. In any area of complex ideas, there are difficulties in accommodating subjects comfortably into one or other subject division, and by using horizontal groupings, new subjects are formed. Information theorists interested in structuring knowledge for the purpose of clarifying understanding recognized this horizontal grouping as recurring "facets" or planes of understanding, and defined the term facet analysis as the process by which a subject is analyzed based on this principle. B.C. Vickery [137] compares the semantic model of human memory structures used by Linsday and Norman [78] with the analysis of subjects by facet used by himself and others in subject classification. Lindsay and Norman described "roles which characterize parts of an event" as:

Action, Agent, Conditional, Instrument, Location, Object, Purpose, Quality, Recipient, Time

These correspond closely with some of the facets defined by Vickery [137] as being useful within a science and technology classification:

Attributes, Object, Parts, Place, Processes, Properties, Substances, Time, Recipient

Research based on facet analysis [36] theory has been able to define facets which may be labeled differently in different domains, but which are essentially transferable. Table 3.2 shows examples taken from knowledge bases built for research purposes using the hypertext system "NoteCards".

generic labeling	*catering*	*social skills*
parts	ingredients	attitudes
processes	processes	processes
procedures	recipes	procedures
agents	equipment	people
properties	characteristics	situations
products	dishes	

Table 3.2: Examples of facet labeling

There are many faceted ontologies that have been proposed for specific application domains (e.g. see Table 3.1). For example Ruben Prieto-Diaz ([103, 104]) has proposed "faceted classification" for a reusable software library. In a faceted classification scheme, the facets may be considered to be dimensions in a cartesian classification space, and the value of a facet is the position of the artifact in that dimension. For software, one might have facets with values such as "Operand", "Functionality", "Platform", "Language", etc. Prieto-Diaz claims that a fixed (and small) number of facets is sufficient for classifying all software.

3.8.3 Conceptual Schemas

Consider an object oriented conceptual schema that supports the classic abstraction mechanisms, *classification*, *generalization* and *attribution* (see [14] for an introduction to conceptual modeling), having a class `Hotel` which has two attributes, namely `location` and `sports`, as shown in Figure 3.7. This conceptual schema can be represented straightforwardly in several conceptual modeling systems, such as the Semantic Index System [71]. Although winter sports are not possible in Heraklion, this schema allows the following instance of this schema: `Hotel(1), Hotel(1).location=Heraklion, Hotel(1).sports=Snowboard`. In our model we can declare that WinterSports are not possible in Crete and from this declaration we can infer that all descriptions which are "less than" the description `Crete.WinterSports` are not valid either. Concerning navigation, note that in an object-oriented schema usually one can only navigate the class hierarchy. The navigation with respect to the values of the attributes is usually not supported because, as we previously showed, we cannot define the combinations of values that are valid or invalid. Thus we could say that a "conventional" conceptual model usually does not store the information needed for preventing the indexing errors, and for aiding the indexing process.

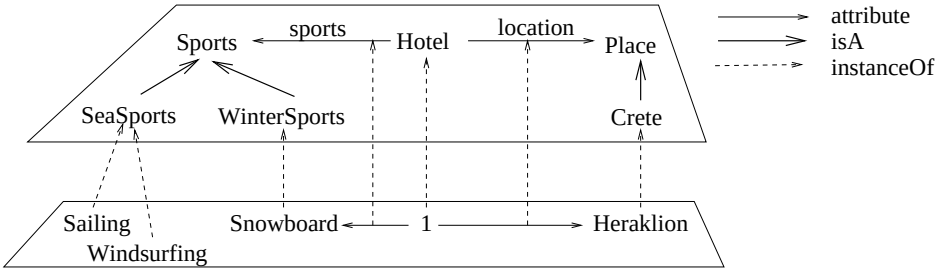


Figure 3.7: A conceptual model with IsA and Attributes

3.8.4 Knowledge Representation and Reasoning Approaches

Below we investigate whether we can represent the extended faceted ontologies in some logic-based languages, in particular, Propositional Calculus, First Order Predicate Calculus, Horn Clauses and Description Logics. We study each language in the following manner. At first we look for a method for "translating" an ontology G, to a set of well formed formulas Δ_G of that language. Then we investigate whether the semantics and the corresponding inference rules of the language allow checking the validity of descriptions.

- **Propositional Calculus**

The semantics of a logical language has to do with associating the elements of the language with elements of a domain of discourse. For propositional logic, atoms, where an atom can be a string or one of the distinguished atoms TRUE and FALSE, are associated with *propositions* about the world, and an association of atoms with propositions is called an interpretation. Under a given interpretation atoms have the *values* TRUE or FALSE. Propositional wffs are atoms connected by the logical connectives $\vee, \wedge, \neg, \rightarrow$. An interpretation is a model of a set of wff Δ if it satisfies (makes true) all wffs in Δ.

The next table describes a candidate method for "translating" a $PEFO < \mathcal{F}, P >$ to a set of propositional wff's Δ_G. For each term $t \in \mathcal{T}$ we assume a propositional atom also denoted by t.

$< \mathcal{F}, P >$	Propositional Calculus
$t \preceq t'$	$t \rightarrow t'$
$t_1 \cdot t_2 \cdot ... \cdot t_k \in P$	$t_1 \wedge ... \wedge t_k$

Clearly a model of Δ_G will satisfy each conjunction that corresponds to a description in P, and since $\mathcal{T} \subseteq P$, it will satisfy all atoms that correspond to terms in $\mathcal{T}$. This implies that each model of Δ_G will also satisfy all conjunctions of atoms, that is, *all* descriptions. From this we conclude that an extended faceted ontology cannot be represented in propositional calculus.

For analogous reasons we cannot represent a $NEFO$ in propositional calculus.

- **First Order Predicate Calculus**

  Predicate logic is more powerful than propositional logic since it can refer to objects in the world as well as to propositions about the world. In particular, an interpretation of an expression is an assignment (denotation) that maps object constants into objects in the world, n-ary relation constants to n-ary relation, etc. The set of objects to which object constant assignments are made is called the domain of the interpretations. An atom is a relation constant of arity n followed by n terms, where a term can be an object constant, a variable, or a function constant of arity k followed by k terms. An atom has the value True in a given interpretation just in case the denoted relation holds for those individuals denoted by its terms, otherwise it has value false.

  The next table describes how a $PEFO < \mathcal{F}, P >$ can be translated to a set of first order predicate calculus wffs Δ_G.

$< \mathcal{F}, P >$	First Order Predic. Calc.
$t \preceq t'$	$\forall\ X\ [t(X) \rightarrow t'(X)]$
$t_1 \cdot ... \cdot t_k \in P$	$\exists\ X\ [t_1(X) \wedge ... \wedge t_k(X)]$

A description $t_1 \cdot ... \cdot t_k$ is valid in $< \mathcal{F}, P >$ iff $\Delta_G \models \exists\ X\ [t_1(X) \wedge ... \wedge t_k(X)]$.

The next table describes how a $NEFO < \mathcal{F}, N >$ can be translated to a set of first order predicate calculus wffs Δ_G.

$< \mathcal{F}, N >$	First Order Predic. Calc.
$t \preceq t'$	$\forall\ X\ [t(X) \rightarrow t'(X)]$
$t_1 \cdot ... \cdot t_k \in N$	$\nexists\ X\ [t_1(X) \wedge ... \wedge t_k(X)]$

A description $t_1 \cdot ... \cdot t_k$ is invalid in $< \mathcal{F}, N >$ iff $\Delta_G \models \nexists\ X\ [t_1(X) \wedge ... \wedge t_k(X)]$.

Concerning the inference procedures of first order predicate calculus, for checking the validity of a wff we can apply the resolution refutation [109] which is a sound and complete inference procedure for predicate calculus. However recall that predicate calculus is semi-decidable, and that even on problems for which resolution refutation terminates, the procedure is NP-hard - as is any sound and complete inference procedure for the first order predicate calculus [16].

- **Horn Clauses**

  Horn clauses form the basis of the language PROLOG. Horn clauses are clauses that have at most one positive literal. A clause having at least one negative literal and a single positive literal, is called a *rule*. A clause with no negative literals is called a *fact*. A clause with no positive literals is called a *goal*.

  The next table describes a candidate method for translating a $PEFO < \mathcal{F}, P >$ to a set of Horn Clauses Δ_G.

$< \mathcal{F}, P >$	Horn Clauses
$t \preceq t'$	$t'(X) : -t(X)$
$t_1 \cdot t_2 \cdot ... \cdot t_k \in P$	$t_1(C) : -$ $t_2(C) : -$ ... $t_k(C) : -$

  For each relationship $t \preceq t'$ we derive a rule, while for each description $t_1 \cdot ... \cdot t'_k \in P$, we produce k facts, $t_1(C), ...t_k(C)$, where C denotes a new, unused constant. Note that this resembles the way that resolution for predicate calculus eliminates existential quantifiers by Skolem functions. In particular the wff $\exists\ X\ t(X)$ becomes $t(Sk)$ where Sk is a new constant.

  A description $t_1 \cdot ... \cdot t_k$ is valid in $< \mathcal{F}, P >$ iff Δ_G implies the goal "$: -t_1(X), ..., t_k(X)$".

However we cannot represent a $NEFO$ with Horn rules as we cannot express the wff $\nexists X\ [t_1(X) \wedge ... \wedge t_k(X)]$ as a fact or a rule.

- **Description Logic**

  Attempts to providing a formal ground to Semantic Networks [61, 74] and Frames [90] led to the development of systems with explicit model-theoretic semantics such as KL-ONE [17]. More recently, KR systems based on Description Logics or terminological systems [35], have been proposed as successors of KL-ONE, with the Tarskian semantics for first order logic. The main challenge of DLs is to identify the fragment of formal logic that both captures the features needed for representation purposes and still allows for the design of effective and efficient reasoning methods. Thus there are many variations of DL according to the provided expressivity, that is, according to the supported concept constructors (e.g. see [35]). Any Description Logic is a fragment of First Order Logic (FOL). In particular, any (basic) DL is a subset of $\mathcal{L}_3$ i.e. the function-free FOL using only at most *three* variable names.

  A knowledge base, also referred as a DL theory, denoted by Σ, is formed by two components: the *intensional* one, called TBox (denoted by $\mathcal{TB}$), and the the *extensional* one, called ABox (denoted by $\mathcal{AB}$), thus we write $\Sigma = (\mathcal{TB}, \mathcal{AB})$. The former contains the definition of predicates, while the latter contains the assertion over constants. The representation in DL is at the "predicate level": no variables are present in the formalism.

  The next table describes the method for deriving a $\Sigma_G = (\mathcal{TB}_G, \mathcal{AB}_G)$ from a $PEFO$ $< \mathcal{F}, P >$, where $\mathcal{TB}_G$ is a *simple*-TBox, and the $\mathcal{AB}_G$ is an empty ABox.

$< \mathcal{F}, P >$	Description Logic
$t \preceq t'$	$t \dot{\leq} t'$
$t_1 \cdot t_2 \cdot ... \cdot t_k \in P$	$x \dot{\leq} t_1 \sqcap ... \sqcap t_k$

  For each $t \preceq t'$ we derive a *primitive concept specification* ($\dot{\leq}$). For each description $d = t_1 \cdot ... \cdot t_k \in P$ we define a new concept x defined as $x \dot{\leq} t_1 \sqcap ... \sqcap t_k$.

  A description $d = t_1 \cdot ... \cdot t_k$ can be written as a DL expression $\phi_d = t_1 \sqcap ... \sqcap t_k$, and clearly d is valid in $< \mathcal{F}, \mathcal{P} >$ iff $\Sigma_G \models \phi_d$.

  For checking the validity of descriptions we can exploit the *consistency* reasoning service, in particular, a description ϕ_d is valid if the TBox $\mathcal{TB}_G \cup \{\neg\phi_d\}$ is unsatisfiable. Concerning the inference procedures of DL, tableaux calculus is a decision procedure for solving the

problem of satisfiability.

The next table describes the method for deriving $\Sigma_G = (\mathcal{TB}_G, \mathcal{AB}_G)$ from a $NEFO$ $< \mathcal{F}, N >$, where $\mathcal{TB}_G$ is a *free*-TBox, and the $\mathcal{AB}_G$ is an empty ABox.

$< \mathcal{F}, N >$	Description Logic
$t \preceq t'$	$t \dot{\leq} t'$
$t_1 \cdot t_2 \cdot ... \cdot t_k \in P$	$t_1 \sqcap ... \sqcap t_k \dot{\leq} \bot$

A description d is invalid in $< \mathcal{F}, N >$ iff $\Sigma_G \models \phi_d \equiv \bot$

3.9 Summary and Conclusion

We presented a novel model for indexing and retrieving objects according to multiple aspects or facets. The proposed model is a faceted scheme enriched with a method for specifying the combinations of terms that are valid. This feature allows generating dynamically navigation trees which are suitable for browsing. Moreover these trees can be used in the process of indexing (to aid the indexer and prevent indexing errors).

Specifically, we provided two methods for establishing description validity. In the first one, the designer provides a set of valid descriptions P, while in the second he provides a set of invalid descriptions N. The designer does not have to specify all the valid descriptions in the first case, nor all the invalid in the second. This is because we gave a semantic interpretation to each one of these extended faceted schemes which allows us to infer new valid descriptions (in the first case) and new invalid descriptions (in the second). This reduces the effort needed in order to establish description validity.

Chapter 4

Mediators over Ontology-based Sources

Searching for information in various sources (such as bibliographic databases, web catalogs) often requires the user to pose the query using a controlled vocabulary. While the use of controlled vocabulary makes the indexing and retrieval of information effective, it forces the user of the database to become familiar with the terms in the ontology. This requirement can pose a considerable burden on the user, especially when the user may want to extract information from more than one sources which may use different ontologies for indexing their objects. The need for using more than one sources has been necessitated by the increasingly distributed nature of information. While it is reasonable to expect the user to be conversant with an ontology, to expect the user to be familiar with all the ontologies that are used to index the various databases is definitely not so. The problem faced by users who are not familiar with many ontologies can be addressed by standardization and the use of a single, universal ontology. However, except for very narrowly defined subject matter, such a solution is not feasible. An alternative solution is to provide software solutions that will permit the user to pose his queries using terms from an ontology that was not necessarily used to index the database being searched. The software will then translate the query into terms from the ontology actually used to index the database.

This chapter describes a flexible and efficient model for building *mediators* over ontology-based sources. Roughly, a mediator (initially proposed in [140]) is a "secondary" information source aiming at providing a uniform interface to a number of underlying sources (which may be primary or secondary). Users submit queries to the mediator. Upon receiving a user query, the mediator queries the underlying sources. This involves selecting the sources to be queried and formulating the query to be sent to each source. These tasks are accomplished based on what the mediator "knows" about the underlying sources. Finally , the mediator appropriately

combines the returned results and delivers the final answer to the user.

As exact translation of user queries to the mediator is not always possible, we define two types of approximate translation of user queries, namely, *lower* and *upper* translation. What kind of translation will be used at the mediator level and what kind of answer will be requested at the source level is decided by the mediator designer at design time and/or the mediator user at query time. Therefore a prominent feature of our approach is that sources and mediators can operate in a *variety of modes* according to specific application needs. As a consequence, our mediators are quite flexible and can adapt to a variety of situations.

Another advantage of our approach is that a mediator can be constructed quite easily, therefore ordinary Web users can use our model in order to define their own mediators. In this sense, our approach can be used for defining user views over existing Web catalogs.

The remaining of this chapter is organized as follows: Section 4.1 defines the problem and Section 4.2 defines the mediators. Section 4.3 discusses the query answering process and Section 4.4 the compatibility condition. Section 4.5 discusses query evaluation and Section 4.6 discusses enhancements of the query answering process. Section 4.7 describes mediators which have also a stored interpretation and Section 4.8 discusses implementation issues. Section 4.9 discusses related work and, finally, Section 4.10 concludes the chapter and discusses further research.

4.1 The Problem

Although several sources may carry information about the same domain, they often employ different ontologies, with terms that correspond to *different natural languages*, or *different levels of granularity*. For example, consider two sources S_1 and S_2 that both provide access to electronic products as shown in figures 4.1.(a) and 4.1.(b). Each source consists of an ontology plus a database that indexes objects under the terms of that ontology. However, the two sources provide *different* information about electronic products - as seen in the figures. Suppose now that we want to provide unified access to these two sources through a single ontology which is familiar to a specific group of users. An example of such a unifying ontology is shown in Figure 4.1.(c), and constitutes part of what we call a "mediator".

A *mediator* is a secondary source that can bridge the heterogeneities that may exist between two or more sources in order to provide unified access to those sources. Specifically, a mediator has an ontology with terminology and structure that reflect the needs of its potential users, but does *not* maintain a database of objects. Instead, the mediator maintains a number of *articulations* to the sources. An articulation to a source is a set of relationships between the

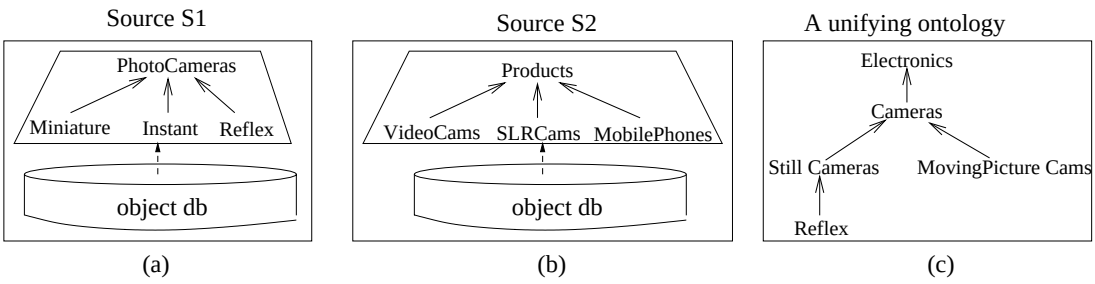


Figure 4.1: Two sources providing access to electronic products

terms of the mediator and the terms of that source. These relationships are defined by the designer of the mediator at design time and are stored at the mediator. A method for assisting the desigher to define these relationships will be presented in Chapter 5. Figure 4.2 shows the general architecture of a mediator.

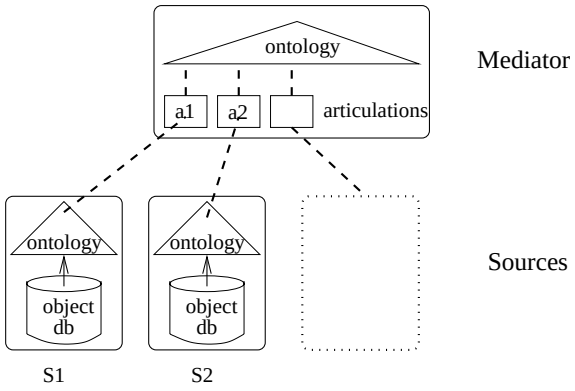


Figure 4.2: The mediator architecture

Users formulate queries over the ontology of the mediator and it is the task of the mediator to choose the sources to be queried, and to formulate the query to be sent to each source. To this end, the mediator uses the articulations in order to *translate*, or *approximate*, queries over its own ontology to queries over the ontologies of the articulated sources. Then it is again the task of the mediator to combine appropriately the results returned by the sources in order to produce the final answer.

4.2 Mediators: Definition

According to Chapter 2, an ontology-based source over an underlying set of objects Obj consists of:

1) an ontology $(T, \preceq)$, and

2) a stored interpretation I of T.

The terminology T contains terms that are familiar to the users of the source; the subsumption relation $\preceq$ contains relationships between terms of T; and the stored interpretation I associates each term t with the objects that are indexed under t (by the indexer).

Consider now a set of sources $S_1, ..., S_k$ over the *same* underlying set of objects Obj. In general, two different sources may have different terminologies either because the users of the two sources are familiar with different sets of terms, or because one source indexes objects at a different level of granularity than the other. The two sources may also have different subsumption relations as the relationships between any two given terms may be perceived differently in the two sources. Finally, two different sources may have different stored interpretations, for example some objects may have been indexed by one source but not by the other.

Clearly if one wants to combine or *integrate* information coming from different sources one has to cope with the above heterogeneities. One way of rendering all these heterogeneities transparent to users is through the use of *mediators* (initially proposed in [140]).

The problem of information integration has attracted considerable attention in the last few years, especially in the area of databases (see [47] for a comprehensive overview). The main idea is to have users access the information sources through a common schema that reflects their needs. Two main approaches seem to have emerged, namely the *virtual view* approach and the *materialized view* approach. In the first, only the common schema is stored (but no data), while in the second both the common schema and data over that schema are stored. Our approach is similar in spirit to the virtual view approach.

In our approach, a mediator M has an ontology $(T, \preceq)$ that reflects the needs of its potential users but has *no* stored objects. Instead, each term at the mediator is related directly or indirectly with terms in the underlying sources. More formally, a mediator is defined as follows:

Def 4.1 A *mediator* M over k sources $S_1 = (T_1, \preceq_1)$,...,$S_k = (T_k, \preceq_k)$ consists of:
1) an ontology $(T, \preceq)$, and
2) a set of *articulations* a_i, one for each source S_i; each articulation a_i is a subsumption relation over $T \cup T_i$.

So, a mediator is just like a source but with an important difference: there is no interpretation stored at the mediator. What is stored at the mediator, instead, is the set of articulations a_i, one for each source S_i. For example, suppose that we want to integrate two Web catalogs which provide access to pages about electronic products. In particular, consider the sources S_1 and

S_2 shown in Figure 4.3 and assume that we want to provide access to these sources through a mediator M as shown in that figure. To achieve integration we enrich the mediator with articulations, i.e. with relationships that relate the terms of the mediator with the terms of the sources as shown in Figure 4.3. The articulations a_1 and a_2 shown in Figure 4.3 are the following sets of subsumption relationships:

$$
\begin{aligned}
a_1 \;=\; & \{\texttt{PhotoCameras} \preceq \texttt{Cameras},\ \texttt{StillCameras} \preceq \texttt{PhotoCameras},\ \texttt{Miniature} \preceq \texttt{StillCameras},\\
& \texttt{Instant} \preceq \texttt{StillCameras},\ \texttt{Reflex}_1 \preceq \texttt{StillCameras},\ \texttt{Reflex}_1 \preceq \texttt{Reflex},\ \texttt{Reflex} \preceq \texttt{Reflex}_1\}\\
a_2 \;=\; & \{\texttt{Products} \preceq \texttt{Electronics},\ \texttt{SLRCams} \preceq \texttt{Reflex},\\
& \texttt{VideoCams} \preceq \texttt{MovingPictureCams},\ \texttt{MovingPictureCams} \preceq \texttt{VideoCams}\}
\end{aligned}
$$

Note that a_1 is a subsumption relation over $T \cup T_1$ and a_2 is a subsumption relation over $T \cup T_2$, as required by the definition of an articulation (Def. 4.1).

Figure 4.4 shows another example of a mediator over three sources. These three sources provide access to tourist information and the information is organized by location.

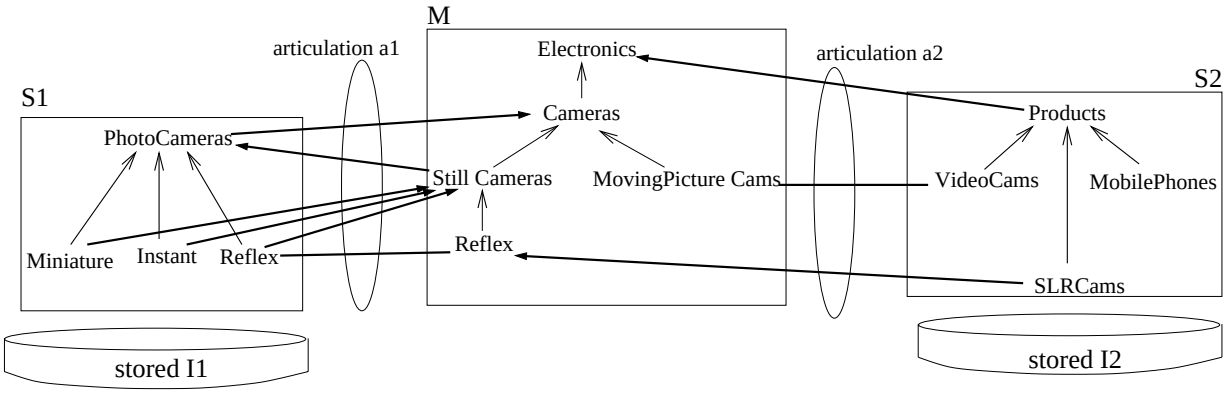


Figure 4.3: A mediator over two catalogs of electronic products

Now, in the presence of several sources, one and the same term may appear in two or more sources. If the same term appears in two different sources then we consider the two appearances as two different terms. This is denoted here by subscripting each term of a source S_i by the subscript i, and can be implemented in practice by, say, prefixing terms by the names of sources. Take, for example, the term DB and suppose that it appears in sources S_i and S_j. Then, from the mediator's point of view there are two distinct terms: the term $\texttt{DB}_i$ in source S_i and the term $\texttt{DB}_j$ in source S_j. This is reasonable as the same term can have different interpretations

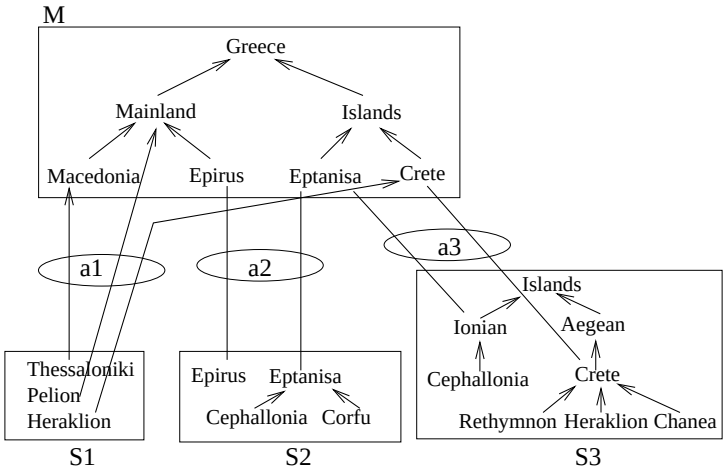


Figure 4.4: A mediator over three catalogs of tourist information

(meanings) in different sources. Thus for every $i \neq j$ we assume $T_i \cap T_j = \emptyset$; and for every i we assume $T \cap T_i = \emptyset$. In this way we overcome the problems of homonyms. Under these assumptions, two terms are considered equivalent, e.g. $DB_i \sim DB_j$, only if they can be shown to be equivalent through the articulations a_i and a_j, e.g. DB_i and DB_j are equivalent if there is a term t in T such that $t \sim_{a_i} DB_i$ and $t \sim_{a_j} DB_j$.

Integrating objects from several sources often requires *restoring the context* of these objects, i.e. adding information that is missing from the original representation of the objects which concerns the context of the objects. Consider for example a mediator which provides access to electronic products according to the *type* of the products and according to the *location* of the stores that sell these products. Suppose that the mediator has two underlying sources S_1 and S_2 as shown in Figure 4.5.

Assume that S_1 is the source of a store located in Heraklion, while S_2 is the source of a store located in Paris. The context of the objects of each source, here the location of the store that sells each product, can be restored by adding to the articulations appropriate relationships. Specifically, for defining that all `PhotoCameras` of the source S_1 are available through a store located in `Heraklion`, it suffices to put in the articulation a_1 the relationship:

$$\texttt{PhotoCameras}_1 \preceq \texttt{Heraklion}$$

while for defining that all products of the source S_2 are available through a store located in `Paris`, it suffices to put in the articulation a_2 the following relationship:

$$\top_2 \preceq \texttt{Paris}$$

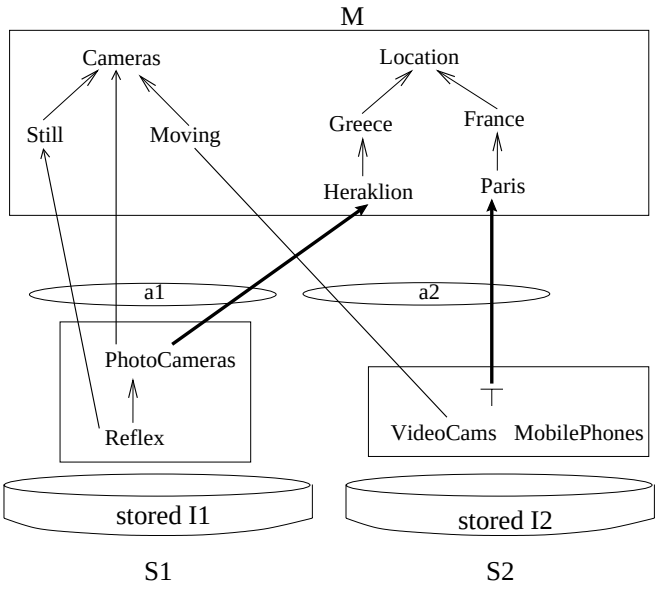


Figure 4.5: Using articulations to restore the context of the objects of the sources

This example demonstrates how the articulations of the mediator can restore the context of the objects of the sources.

4.3 Query Answering

The mediator receives queries over its own terminology T. As it does not have a stored interpretation of T, the mediator answers queries using an interpretation of T obtained by *querying* the underlying sources. However, as the mediator and the sources have different terminologies, for computing the interpretation of a term $t \in T$, the mediator sends to each source S_i a query that *translates* the term t to a query that can be answered by the source, and then it takes the *union* of the answers returned by the sources. The definition of translations is based on the articulations of the mediator.

Thus we will actually define an interpretation I of the mediator terminology, based on the interpretations I_i stored at the sources, on the one hand, and on the articulations a_i, $i = 1, .., k$, on the other. Conceptually, once the interpretation I of the mediator is defined, the mediator can answer queries just like any other source does, i.e. from its sure model I^- and from its possible model I^+.

In order to define the mediator interpretation I we proceed as follows: for every term t of the mediator terminology T:

1. first, we define a translation t^i of t in a_i, in the form of a query to source S_i, $i = 1, .., k$;
2. then, we evaluate the query t^i at source S_i, $i = 1, .., k$, and
3. finally, we define $I(t)$ by taking the union of the answers to the queries t^i returned by the sources.

Now, there are two ways to translate t using the articulation a_i, that we shall call the *upper approximation* of t and the *lower approximation* of t in a_i. We will also use the terms *narrow translation* and *broad translation* for lower approximation and upper approximation respectively. Roughly, the upper approximation of t in a_i is the conjunction of all terms of T_i that subsume t in a_i, and the lower approximation of t in a_i is the disjunction of all terms of T_i that t subsumes in a_i. In order to define these notions formally we need the notions of *tail* and *head* of a term *relative* to an articulation:

Def 4.2 Given a term $t \in T$ and articulation a_i we define

$$tail_i(t) = \{s \in T_i \mid sa_it\} \quad \text{and} \quad head_i(t) = \{u \in T_i \mid ta_iu\}$$

Def 4.3 Let $M = (T, \preceq, a_1, ..., a_k)$ be a mediator over sources $S_1, .., S_k$. If t is a term of T then

- the *lower approximation* of t with respect to a_i, denoted t^i_l, is defined by

$$t^i_l = \bigvee tail_i(t)$$

- the *upper approximation* of t with respect to a_i, denoted t^i_u, is defined by

$$t^i_u = \begin{cases} \bigwedge head_i(t), & \text{if } head_i(t) \neq \emptyset \\ t^i_l, & \text{otherwise} \end{cases}$$

Note that if $head_i(t) = \emptyset$ then we consider that $t^i_u = t^i_l = \bigvee tail_i(t)$. The reason behind this choice is that we want the interpretation obtained by using lower approximation to be less than or equal to ($\sqsubseteq$) the interpretation obtained by using the upper approximation.

Here are some examples of approximations for the mediator shown in Figure 4.3:

$\texttt{StillCameras}^1_l = \texttt{Miniature} \vee \texttt{Instant} \vee \texttt{Reflex}$

$\texttt{StillCameras}^1_u = \texttt{PhotoCameras}$

$\texttt{Reflex}^1_l = \texttt{Reflex}$

$\texttt{Reflex}^1_u = \texttt{Reflex} \wedge \texttt{PhotoCameras}$

$\texttt{Reflex}^2_l = \texttt{SLRCams}$

$\texttt{Cameras}^1_l = \texttt{PhotoCameras} \vee \texttt{Miniature} \vee \texttt{Instant} \vee \texttt{Reflex}$

$\texttt{Cameras}^1_u = \texttt{PhotoCameras} \vee \texttt{Miniature} \vee \texttt{Instant} \vee \texttt{Reflex}$

$\texttt{MovingPictureCams}^1_u = \texttt{MovingPictureCams}^1_l = \epsilon$

Note that for a given term $t \in T$ the evaluation of t_u^i requires the previous evaluation of $head_i(t)$, and the evaluation of t_l^i requires the previous evaluation of $tail_i(t)$. However, if we compute the transitive closure of a_i then the evaluation of $head_i(t)$ and $tail_i(t)$ is straightforward.

Now, the approximations t_u^i and t_l^i of t are actually queries to the source S_i, and as such each can have a sure answer and a possible answer (see Chapter 2). As a consequence, we can define at least four different interpretations I for the mediator. Assuming for simplicity that *all* sources respond in the same manner, i.e. either all give a sure answer or all give a possible answer, we can define exactly four interpretations for the mediator that we shall denote by I_{l-}, I_{l+}, I_{u-}, I_{u+}. These interpretations are defined as follows:

1 Lower approximation of t at mediator and sure answer from sources:
$I_{l-}(t) = \bigcup_{i=1}^{k} I_i^-(t_l^i)$

2 Lower approximation of t at mediator and possible answer from sources:
$I_{l+}(t) = \bigcup_{i=1}^{k} I_i^+(t_l^i)$

3 Upper approximation of t at mediator and sure answer from sources:
$I_{u-}(t) = \bigcup_{i=1}^{k} I_i^-(t_u^i)$

4 Upper approximation of t at mediator and possible answer from sources:
$I_{u+}(t) = \bigcup_{i=1}^{k} I_i^+(t_u^i)$

So, the mediator can answer queries submitted by its users based on any of the above four interpretations. Moreover, for any of these four interpretations, the mediator can give either a sure answer or a possible answer - just like any source can (see Chapter 2). By consequence, we can distinguish eight possible modes in which a mediator can operate. Each mode essentially corresponds to a different answer model of the mediator. The operation modes of a mediator and the corresponding interpretations are summarized in Table 4.1.

operation mode at the mediator	*term approximation at mediator*	*query evaluation at source*	*query evaluation at mediator*	*the answer model of the mediator*
1	lower	sure	sure	I_{l-}^-
2	lower	possible	sure	I_{l+}^-
3	upper	sure	sure	I_{u-}^-
4	upper	possible	sure	I_{u+}^-
5	lower	sure	possible	I_{l-}^+
6	lower	possible	possible	I_{l+}^+
7	upper	sure	possible	I_{u-}^+
8	upper	possible	possible	I_{u+}^+

Table 4.1: Modes in which a mediator can operate

Very roughly speaking, as we go down the table (from mode 1 to 8) the answer to the same user query is more likely to contain objects that are not "relevant" to the query. This is described more precisely in the following proposition.

Prop. 4.1 The answer models of the mediator are ordered as follows:

$$
\begin{array}{rcl}
(a)\ I^-_{l-} & \sqsubseteq & I^-_{l+} \\
(b)\ I^-_{u-} & \sqsubseteq & I^-_{u+} \\
(c)\ I^+_{l-} & \sqsubseteq & I^+_{l+} \\
(d)\ I^+_{u-} & \sqsubseteq & I^+_{u+} \\
(e)\ I^-_{l-} & \sqsubseteq & I^+_{l-} \\
(f)\ I^-_{l+} & \sqsubseteq & I^+_{l+} \\
(g)\ I^-_{u-} & \sqsubseteq & I^+_{u-} \\
(h)\ I^-_{u+} & \sqsubseteq & I^+_{u+}
\end{array}
$$

Proof:

The proofs of the propositions (a)-(d) follow easily from the fact that in every model I_i of a source S_i holds: $I_i^- \sqsubseteq I_i^+$.

The proofs of the propositions (e)-(h) follow easily from the fact that in every model I of the mediator holds: $I^- \sqsubseteq I^+$.

◇

Figure 4.6 shows graphically the orderings of the above proposition. The nodes represent the answer models shown in Table 4.1; for example, m_1 represents I^-_{l-}, m_2 represents I^-_{l+}, and so on. An arrow from node m_i to a node m_j means that $m_i \sqsubseteq m_j$.

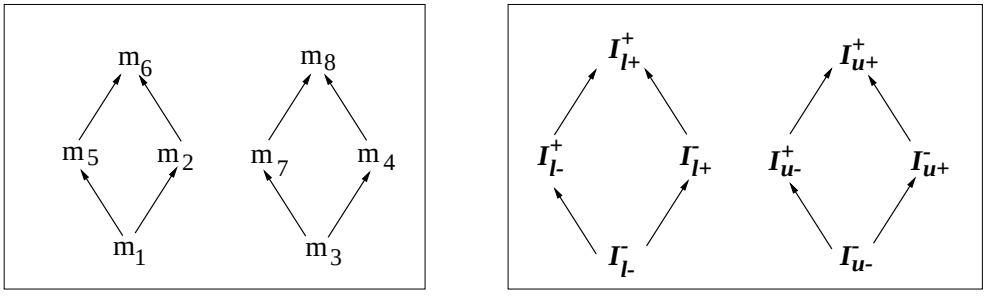


Figure 4.6: The ordering ($\sqsubseteq$) of the eight answer models of the mediator

For example, the interpretation of the term `RelationalDatabases` in each of the models $I^-_{l-}, I^-_{l+}, I^-_{u-}, I^-_{u+}$ of the mediator shown in Figure 4.7 follow:

$$I^-_{l-}(\texttt{RelationalDatabases}) \quad = \quad \{1\}$$

$$I^-_{l+}(\texttt{RelationalDatabases}) = \{1,2\}$$
$$I^-_{u-}(\texttt{RelationalDatabases}) = \{1,2,3\}$$
$$I^-_{u+}(\texttt{RelationalDatabases}) = \{1,2,3,4\}$$

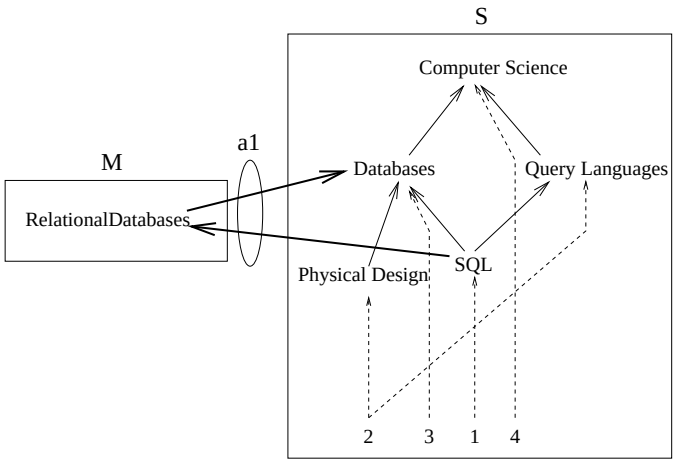


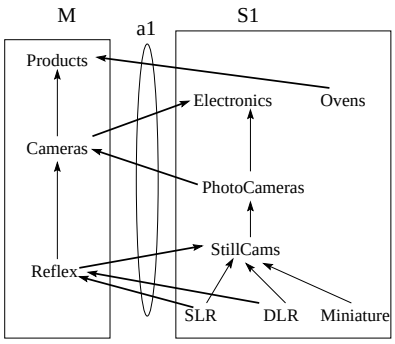
Figure 4.7: A mediator over one source

Another example of mediator operation is given in Figure 4.8. Figure 4.8.(a) shows a mediator having an articulation to a source S_1 and Figure 4.8.(b) shows two tables. The table at the upper part of the figure shows the interpretation I_1 of source S_1 and the corresponding (sure and possible) models. The first column of the table at the bottom part shows three queries which are actually the three terms of T. The subsequent columns show what the mediator returns in each of the first four operation modes.

The operation modes of the mediator can either be decided (and fixed) by the mediator designer at design time, or indicated by the mediator users at query time. We can distinguish at least three approaches:

- *Fixed Approach.* The mediator designer selects and fixes one of the eight possible modes of operation for the mediator and the sources, and users simply submit their queries to the mediator without any further indication.
- *Variable Approach.* The mediator users submit their queries along with a specification for the query evaluation mode they wish. This is done by providing values to the mediator for selecting one of the eight operation modes from Table 4.1. For example, the following user specification selects the operation mode number 3 from Table 4.1:

  term approximation at mediator = `upper`



(a)

T_1	I_1	I_1^-	I_1^+
$\bot$	$\emptyset$	$\emptyset$	$\emptyset$
$\top$	$\emptyset$	{0,1,2,3,4,5,6}	{0,1,2,3,4,5,6}
Ovens	{6}	{6}	{0,1,2,3,4,5,6}
Electronics	{5}	{0,1,2,3,4,5}	{0,1,2,3,4,5,6}
PhotoCameras	{4}	{0,1,2,3,4}	{0,1,2,3,4,5}
StillCams	{3}	{0,1,2,3}	{0,1,2,3,4}
SLR	{2}	{2}	{0,1,2,3}
DLR	{1}	{1}	{0,1,2,3}
Miniature	{0}	{0}	{0,1,2,3}

Q	1:I_{l-}^-	2: I_{l+}^-	3: I_{u-}^-	4 :I_{u+}^-
Products	{0,1,2,3,4,6}	{0,1,2,3,4,5,6}	{0,1,2,3,4,5,6}	{0,1,2,3,4,5,6}
Cameras	{0,1,2,3,4}	{0,1,2,3,4,5}	{0,1,2,3,4,5}	{0,1,2,3,4,5,6}
Reflex	{1,2}	{0,1,2,3}	{0,1,2,3}	{0,1,2,3,4}

(b)

Figure 4.8: A mediator with one articulation to a source S_1

query evaluation at source = `sure`

query evaluation at mediator = `sure`

- *Mixed Approach.* The mediator designer selects and fixes some of the attributes of Table 4.1, and the user provides the remaining ones. For example, the designer may select and fix the query evaluation mode at source (i.e. sure or possible) and the kind of term approximation at the mediator (i.e. lower or upper approximation), during design time, while the users select the query evaluation mode at the mediator, during query time.

Clearly, selecting one of the above approaches depends on several factors, such as the reliability of the sources or the level of expertise of the users, and so on. One can even think of more sophisticated modes of mediator operation than those presented in Table 4.1. For example, the mediator designer may assign a degree of reliability to each source and then ask sources to evaluate queries in a mode depending on their degree of reliability. In this chapter, however, we do not pursue this idea any further.

4.4 The Compatibility Condition

We have seen so far how the mediator communicates with the sources through the articulations. In fact, the articultions are the *only* means of communication between the sources and the mediator. Now, certain kinds of articulation are better than others. One kind of articulations that are of particular interest are those that ensure what we call *"compatibility"* between the sources and the mediator.

Def 4.4 A source S_i is *compatible* with the mediator M if for any terms s, t in T_i, if $sa_i t$ then $s \preceq_i t$.

That is, S_i is compatible with the mediator whenever the following condition holds: for all terms s and t in T_i, if s is subsumed by t in the articulation a_i then s is also subsumed by t in $\preceq_i$.

For example, the source S_1 of Figure 4.3 is compatible with the mediator since we have

`Miniature` a_1 `PhotoCameras` and `Miniature` $\preceq_1$ `PhotoCameras`,

`Instant` a_1 `PhotoCameras` and `Instant` $\preceq_1$ `PhotoCameras`,

`Reflex` a_1 `PhotoCameras` and `Reflex` $\preceq_1$ `PhotoCameras`.

An interesting consequence of compatibility is that if a source S_i is compatible with the mediator, then in every model I_i of S_i the following condition holds: $I_i(t_l^i) \subseteq I_i(t_u^i)$ for each mediator term t, where t_l^i is the lower approximation of t and t_u^i is the upper approximation of t. From this property we infer that if all sources are compatible with the mediator then the ordering relation over the eight answer models of the mediator (see Figure 4.6), is enriched as stated by the following proposition.

Prop. 4.2 If all sources are compatible with the mediator then:

$$\begin{array}{lll} (1)\ I^-_{l-} & \sqsubseteq & I^-_{u-} \ (\text{ that is, } m_1 \sqsubseteq m_3) \\ (2)\ I^-_{l+} & \sqsubseteq & I^-_{u+} \ (\text{ that is, } m_2 \sqsubseteq m_4) \\ (3)\ I^+_{l-} & \sqsubseteq & I^+_{u-} \ (\text{ that is, } m_5 \sqsubseteq m_7) \\ (4)\ I^+_{l+} & \sqsubseteq & I^+_{u+} \ (\text{ that is, } m_6 \sqsubseteq m_8) \end{array}$$

Proof:

Let t be a term of T. Clearly, for every $s \in tail_i(t)$ and $u \in head_i(t)$ holds $sa_i u$ (because $sa_i t$ and $ta_i u$). Since the source S_i is compatible we know that $sa_i u \Rightarrow s \preceq_i u$. This implies that in every model I_i of T_i holds:

$$\begin{array}{rcl} \bigcup\{I_i(s) \mid sa_i t\} & \subseteq & \bigcap\{I_i(u) \mid ta_i u\} \Leftrightarrow \\ I_i(t_l^i) & \subseteq & I_i(t_u^i) \end{array}$$

From $I_i(t_l^i) \subseteq I_i(t_u^i)$ we infer that $I_i^-(t_l^i) \subseteq I_i^-(t_u^i)$ and $I_i^+(t_l^i) \subseteq I_i^+(t_u^i)$. From this we obtain propositions (1)-(4).

For example, the proof of proposition (1) i.e. $I^-_{l-} \sqsubseteq I^-_{u-}$, and proposition (3) i.e. $I^+_{l-} \sqsubseteq I^+_{u-}$, is obtained as follows: Since $\forall\ t \in T$ and $\forall\ i = 1..k$ holds $I_i^-(t_l^i) \subseteq I_i^-(t_u^i)$, we conclude that:

$$\bigcup_{i=1..k} I_i^-(t_l^i) \subseteq \bigcup_{i=1..k} I_i^-(t_u^i) \Leftrightarrow I_{l-}(t) \subseteq I_{u-}(t) \Rightarrow \left\{ \begin{array}{ll} I^-_{l-} \sqsubseteq I^-_{u-} & (m_1 \sqsubseteq m_3) \\ I^+_{l-} \sqsubseteq I^+_{u-} & (m_5 \sqsubseteq m_7) \end{array} \right.$$

◇

As a result, the two diagrams of Figure 4.6 are now connected in a single diagram as shown in Figure 4.9.

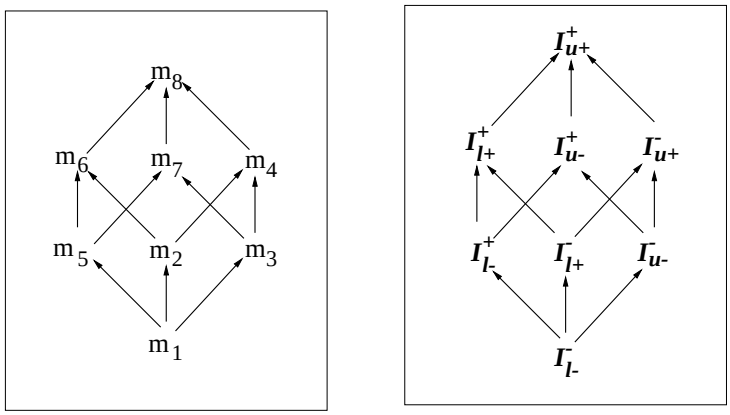


Figure 4.9: The ordering ($\sqsubseteq$) of the eight answer models of the mediator in the case where all sources are compatible with the mediator

Note that the above ordering relationships do not necessarily hold if the sources are not compatible with the mediator. For example, consider a source S_1 with terminology $T_1 = \{b, b'\}$ and no subsumption relationships, such as the one shown in Figure 4.10. Suppose that the source has a stored interpretation I_1 defined as follows: $I_1(b) = \{1\}$ and $I_1(b') = \{2\}$. Now consider a mediator connected to source S_1 through the articulation $a_1 = \{b \preceq t, t \preceq b'\}$, where t is a term of the mediator. Notice that S_1 is not compatible with the mediator because b is subsumed by b' in a_1 while b is not subsumed by b' in $\preceq_1$, i.e. ba_1b' and $b \not\preceq_1 b'$. Here we have $t^1_l = b$ and $t^1_u = b'$, thus $I^-_1(t^1_l) = \{1\}$ and $I^-_1(t^1_u) = \{2\}$. It follows that $I^-_1(t^1_l) \not\subseteq I^-_1(t^1_u)$, which implies $I^-_{l-}(t) \not\subseteq I^-_{u-}(t)$. From this example we see that if the underlying sources are not compatible with the mediator then $I^-_{l-} \sqsubseteq I^-_{u-}$ does not hold.

Another interesting implication of compatibility concerns the *efficiency* of query evaluation. Let s, t be two terms in T_i which are known to the mediator (through a_i) and assume that the mediator knows that source S_i is compatible. In this case if sa_it then $s \preceq_i t$. From this knowledge the mediator can conclude that $I_i(s) \subseteq I_i(t)$, in every model I_i of T_i, and thus $I_i(s) \cap I_i(t) = I_i(s)$ and $I_i(s) \cup I_i(t) = I_i(t)$. This means that the mediator can retain only the minimal elements of the set $head_i(t)$ and still obtain the same answer for the query t^i_u from source S_i. Therefore, if the mediator knows that source S_i is compatible, then instead of sending to source S_i the query $\bigwedge head_i(t)$, the mediator can send the query $\bigwedge min(\ head_i(t)\ )$. Similarly, in the set $tail_i(t)$ the

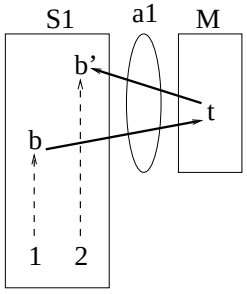


Figure 4.10: A mediator with an incompatible source

mediator can retain only the maximal elements and still obtain the same answer for the query t_l^i from source S_i, i.e., instead of sending the query $\bigvee tail_i(t)$ to source S_i, the mediator can send the query $\bigvee max(\ tail_i(t)\ )$.

For example, in Figure 4.3, as source S_1 is compatible with the mediator, the lower approximation of the term `Camera` is the term `PhotoCameras`. If S_1 were not compatible then the lower approximation of `Camera` would be the disjunction `PhotoCameras` $\vee$ `Miniature` $\vee$ `Instant` $\vee$ `Reflex`.

Thus if S_i is compatible then $t_u^i = \bigwedge min(head_i(t))$ and $t_l^i = \bigvee max(tail_i(t))$. In this case the evaluation of t_u^i and t_l^i can be done more efficiently without having to compute the transitive closure of a_i. Specifically, for evaluating $max(tail_i(t))$ we traverse in depth-first-search the relation a_i starting from the term t. If an element t' of T_i is reached then this term is "collected" and the algorithm does not traverse any other element subsumed by t' (in a_i). All elements of T_i collected during the traversal are then returned. Analogously we can evaluate $min(head_i(t))$. We conclude that if a source is compatible then the approximation of a term for that source can be done more efficiently especially when the articulation to that source is big. Moreover the resulting approximations are shorter, which implies that their transmission requires less time and the underlying source can evaluate these queries more efficiently.

Note that maintaining compatibility is not an easy task. Of course, the designer of the mediator can initially design articulations such that the underlying sources are compatible. However, an update at a source S_i or at the mediator (changing either T or a_i) may destroy compatibility. Therefore the mediator should (periodically) check the compatibility of its sources, e.g. by submitting to them queries allowing to check whether $t \preceq_i t'$. In Section 2.8 we described an inference mechanism that can be used by a source in order to answer to this kind of queries.

4.5 Query Evaluation

The two possible approximations at the mediator (lower or upper) and the two possible query evaluation modes at the sources (sure or possible) gave rise to four possible interpretations at the mediator: I_{l-}, I_{l+}, I_{u-} and I_{u+}. If these four interpretations were stored at the mediator then the interaction between a user and the mediator would be straightforward, i.e.

- the user submits a query to the mediator (as if it were a usual source)
- the mediator and/or the user specifies the answer model to be used
- the mediator uses the specified model to provide a sure or possible answer to the query (as it is done in a usual source)

However, there is *no* interpretation actually stored at the mediator, so, to answer queries, the mediator has to call on the underlying sources, submit to them appropriate queries, then merge the results to produce the final answer for the user. Therefore, the crucial tasks for the evaluation of user queries at the mediator can be summarized as follows:

- translate the user's query into a set of queries to the underlying sources, i.e. determine *what* queries to send to *which* sources;
- merge the results returned by the sources in order to produce the answer to the user's query.

Clearly, the complexity of these tasks depends on the nature of the user query, i.e.

- the form of the query (single term, disjunction of terms, etc.),
- the answer model used by the mediator.

In what follows we analyze the complexity of query evaluation at the mediator with respect to the form that a user query can have, and the answer model used by the mediator for evaluating the query.

The complexity measure that we use in our analysis is the *number of queries* that the mediator sends to the sources in order to answer the user's query, and the *execution time* expressed in terms of several parameters such as the size of the mediator terminology, the size of the articulations, the number of sources, the length of the query and the multitude of the objects of the domain. We consider these to be reasonable measures of complexity as they depend solely on the structure and functioning of the mediator. However, we believe that the number of queries that the mediator needs to send to the sources in order to answer a user query is the most important measure of the complexity of query evaluation at the mediator, as the mediator spends a lot of time waiting the answers of the sources.

We are aware that, in doing so, we do not take into account the complexity of query evaluation at each source. However, the mediator has little or no control over how queries are evaluated at individual sources. This is especially true for the applications that we have in mind, where the mediator is set up by individual users who have no control over the underlying sources (which are Web sources).

In the complexity analysis that follows we consider a mediator over k sources, $S_1, ..., S_k$. Note that we write I_l instead of I_{l-} or I_{l+}, and I_u instead of I_{u-} or I_{u+}, since the translation and the evaluation of queries at the mediator does not depend on the evaluation of queries at the underlying sources. At first we describe the evaluation of queries in the sure models of the mediator, i.e. in the models I_l^- and I_u^-.

An interesting remark that we must mention here is that the mediator *will not* necessarily query all sources. A source is queried only if the evaluation of the answer requires sending a subquery to that source, otherwise the source is not queried. Thus query translation also determines the selection of the sources.

We identify the following forms of queries:

- *Single Term Queries*

  Prop. 4.3 If the query is a single term, i.e. $q = t \in T$, then $I_l^-(t)$ and $I_u^-(t)$ can be evaluated as follows:

$$I_l^-(t) = \bigcup_{i=1..k} I_i(q_l^i(t)) \quad \text{where} \quad q_l^i(t) = \bigvee\{s_l^i \mid s \preceq t\}$$
$$I_u^-(t) = \bigcup_{i=1..k} I_i(q_u^i(t)) \quad \text{where} \quad q_u^i(t) = \bigvee\{s_u^i \mid s \preceq t\}$$

  Proof:

$$\begin{aligned} I_l^-(t) &= \bigcup\{I_l(s) \mid s \in tail(t)\} = \bigcup\{I_l(s) \mid s \preceq t\} = \bigcup\{\cup_{i=1..k} I_i(s_l^i) \mid s \preceq t\} \\ &= \bigcup_{i=1..k} \{\cup I_i(s_l^i) \mid s \preceq t\} = \bigcup_{i=1..k} I_i(\bigvee\{s_l^i \mid s \preceq t\}) = \bigcup_{i=1..k} I_i(q_l^i(t)) \end{aligned}$$

  Analogously, we prove that $I_u^-(t) = \bigcup_{i=1..k} I_i(q_u^i(t))$.

  $\diamond$

  This means that the mediator M can evaluate the query by sending at most one query to each source. Thus M will send at most k queries. Note that if $q_l^i(t) = \epsilon$ (or $q_u^i(t) = \epsilon$) then M does not have to send any query to source S_i.

Moreover, if the mediator knows that a source S_i is *compatible* then the mediator can set

$$q_l^i(t) = \bigvee \ max\ (\bigcup\{tail_i(s) \mid s \preceq t\})$$

If the entire articulation a_i is stored (including the transitive relationships), then the computation of t_l^i can be done in $O(|a_i|)$ time. The same holds for t_u^i. The computation of $q_l^i(t)$ can be done in $O(|T| * |a_i|)$ time. The same holds for $q_u^i(t)$. Thus the computation of all $q_l^i(t)$, or $q_u^i(t)$, for $i = 1..k$ can be done in $O(|T| * |a|)$ where a denotes the union of all articulations, i.e. $a = a_1 \cup ... \cup a_k$.

The set operations over the answers returned by the sources that are needed for computing $I_l^-(t)$, can be performed in $O(k * U)$ time.

Thus the total computations needed by the mediator can be done in $O(|T| * |a| + k * U)$ time.

- *Disjunctive Queries*

  If the query is a disjunction of terms, i.e. $q = t_1 \vee ... \vee t_n$ then

$$\begin{aligned} I_l^-(t_1 \vee ... \vee t_n) &= \bigcup_{i=1..k} I_i(q_l^i(t_1) \vee ... \vee q_l^i(t_n)) \\ I_u^-(t_1 \vee ... \vee t_n) &= \bigcup_{i=1..k} I_i(q_u^i(t_1) \vee ... \vee q_u^i(t_n)) \end{aligned}$$

  Again M can evaluate the query by sending at most one query to each source.

  If, furthermore, a source S_i is *compatible* then the mediator can send to S_i the query:

$$\bigvee \ max(\ \bigcup_{j=1..n} (\cup\{tail_i(s) \mid s \preceq t_j\})\ )$$

  The computation of each $q_l^i(t_1) \vee ... \vee q_l^i(t_n)$ can be done in $O(|T| * |a_i| * n)$ time. Thus the computation of all $q_l^i(t_1) \vee ... \vee q_l^i(t_n)$, for $i = 1..k$ can be done in $O(|T| * |a| * n)$ time.

  The set operations for computing $I_l^-(t)$ can be performed in $O(k * U)$ time.

  Thus the total computations needed by the mediator can be done in $O(|T| * |a| * n + k * U)$ time.

- *Conjunctive Queries*

  If the query is a conjunction of terms, i.e. $q = t_1 \wedge ... \wedge t_n$, then

$$\begin{aligned} I_l^-(t_1 \wedge ... \wedge t_n) &= \bigcap_{j=1..n} (\ \bigcup_{i=1..k} I_i(q_l^i(t_j))\ ) \\ I_u^-(t_1 \wedge ... \wedge t_n) &= \bigcap_{j=1..n} (\ \bigcup_{i=1..k} I_i(q_u^i(t_j))\ ) \end{aligned}$$

Thus the mediator has to send at most one query to each source for each term which appears in the conjunction. This means that M will send at most $k * n$ queries.

The computation of all $q_l^i(t_j)$, for $j = 1..n$, can be done in $O(|T| * |a| * n)$ time.

The set operations for computing $I_l^-(t)$ can be performed in $O(k * n * U)$ time.

Thus the total computations needed by the mediator can be done in $O(|T|*|a|*n+k*U*n)$ time.

- *Conjunctive Normal Form Queries (CNF Queries)*

  A CNF query is a conjunction of maxterms where each maxterm is either a single term or a disjunction of distinct terms ([44]), i.e. $q = d_1 \wedge ... \wedge d_m$ where $d_j = t_{j1} \vee ... \vee t_{jn_j}$, $j = 1..m, n_j \leq |T|$. In this case:

$$
\begin{aligned}
I_l^-(q) &= \bigcap_{j=1..m} (\bigcup_{i=1..k} I_i(q_l^i(t_{j1}) \vee ... \vee q_l^i(t_{jn_j}))) \\
I_u^-(q) &= \bigcap_{j=1..m} (\bigcup_{i=1..k} I_i(q_u^i(t_{j1}) \vee ... \vee q_u^i(t_{jn_j})))
\end{aligned}
$$

  The mediator first evaluates each maxterm (disjunction) by sending at most one query to each source and then it takes the intersection of the returned results. This means that M will send at most $k * m$ queries where m is the number of maxterms.

  Let l be the length of the query, that is, the number of term appearances in the query, i.e. $l = \sum_{j=1..m} n_j$. The computation of $q_l^i(t)$, $i = 1..k$, for all t that appear in q, can be done in $O(|T| * |a| * l)$ time.

  The set operations for computing $I_l^-(t)$ can be performed in $O(k * m * U)$ time.

  Thus the total computations needed by the mediator can be done in $O(|T|*|a|*l+k*m*U)$ time.

- *Disjunctive Normal Form Queries (DNF Queries)*

  A DNF query is a disjunction of minterms where a minterm is either a single term or a conjunction of distinct terms, i.e. $q = c_1 \vee ... \vee c_m$ where $c_j = t_{j1} \wedge ... \wedge t_{jn_j}$, $j = 1..m, n_j \leq |T|$. In this case:

$$
\begin{aligned}
I_l^-(q) &= \bigcup_{j=1..m} (\bigcap_{h=1..n_j} (\bigcup_{i=1..k} I_i(q_l^i(t_{jh})))) \\
I_u^-(q) &= \bigcup_{j=1..m} (\bigcap_{h=1..n_j} (\bigcup_{i=1..k} I_i(q_u^i(t_{jh}))))
\end{aligned}
$$

Thus M will send at most $k * l$ queries, where l is the lenght of the query, that is, the number of term appearances in the query, i.e. $l = \sum_{j=1..m} n_j$.

The computation of all $q_l^i(t)$, for $i = 1..k$, for all t that appear in q can be done in $O(|T| * |a| * l)$ time.

The set operations for computing $I_l^-(t)$ can be performed in $O(k * l * U)$ time.

Thus the total computations needed by the mediator can be done in $O(|T| * |a| * l + k * l * U)$ time.

Table 4.2 summarizes the *number of calls* complexity and Table 4.3 the time complexity. Note that any query that contains the logical connectives $\wedge$ and $\vee$ can be converted to DNF or CNF by using one of the existing algorithms (e.g. see [44]). In our case CNF is preferred to DNF since the evaluation of a query in CNF requires sending a smaller number of queries to the sources.

Query Form		*Maximum number of calls (assuming k sources)*
single term	t	k
disjunction	$t_1 \vee ... \vee t_n$	k
conjunction	$t_1 \wedge ... \wedge t_n$	$k * n$
CNF	$d_1 \wedge ... \wedge d_m$ where $d_j = t_{j1} \vee ... \vee t_{jn_j}$	$k * m$
DNF	$c_1 \vee ... \vee c_m$ where $c_j = t_{j1} \wedge ... \wedge t_{jn_j}$	$k * l$ where $l = \sum_{j=1..m} n_j$

Table 4.2: The *number of calls* complexity of query evaluation at the mediator (for the sure model)

Query Form		*Time Complexity (wrt $\|T\|, \|a\|, k, U$)*
single term	t	$O(\|T\| * \|a\| + k * U)$
disjunction	$t_1 \vee ... \vee t_n$	$O(\|T\| * \|a\| * n + k * U)$
conjunction	$t_1 \wedge ... \wedge t_n$	$O(\|T\| * \|a\| * n + k * U * n)$
CNF	$d_1 \wedge ... \wedge d_m$ where $d_j = t_{j1} \vee ... \vee t_{jn_j}$	$O(\|T\| * \|a\| * l + k * m * U)$
DNF	$c_1 \vee ... \vee c_m$ where $c_j = t_{j1} \wedge ... \wedge t_{jn_j}$	$O(\|T\| * \|a\| * l + k * l * U)$

Table 4.3: The time complexity of query evaluation at the mediator (for the sure model)

We conclude this section by describing the evaluation of queries in the possible models of the mediator, i.e. in the models I_l^+ and I_u^+. The evaluation of a single term query in I_l^+ or I_u^+ is done by evaluating a conjunction of terms in I_l^- or I_u^-, respectively:

$$I^+(t) = \bigcap \{I^-(u) \mid u \in head(t) \text{ and } u \not\sim t\} = I^-(\bigwedge \{u \mid u \in head(t) \text{ and } u \not\sim t\})$$

where $I^+(t)$ stands for $I_l^+(t)$ or $I_u^+(t)$, and I^- stands for I_l^- or I_u^-, respectively. Therefore the complexity analysis of evaluating $I^+(t)$ can be done using Tables 4.2 and 4.3. Finally, the evaluation of a disjunction in I^+ is done by evaluating a DNF query in I^-, and the evaluation of a conjunction in I^+ is done by evaluating a conjunction in I^-.

4.6 Enhancing the Quality of Answers with Object Descriptions

By analogy to the single source case, a mediator can return answers consisting of objects which are accompanied by their indexes. In other words, a mediator can return a set of pairs $(o, D_I(o))$, where I is the model used by the mediator for answering queries. For example consider two sources, S_1 and S_2, providing information about animals (e.g. photos) as shown in Figure 4.11. The terms of source S_1 are in English, while the terms of source S_2 are in French. Moreover a mediator M integrates the information of the two sources and provides a unified access through an ontology with English terms. Assume now that the mediator receives the query $q = \texttt{Animal}$ in which case the mediator sends the query $q_1 = \texttt{Animal} \vee \texttt{Dog}$ to source S_1, and the query $q_2 = \texttt{Mammifères} \vee \texttt{Chat}$ to source S_2 . Moreover, assume that the sources S_1 and S_2 return objects accompanied by their sure indexes. Then, the source S_1 will return the answer

$$\{\ (1, \{\texttt{Dog}, \texttt{Animal}\}),\ (2, \{\texttt{Canis}, \texttt{Dog}, \texttt{Animal}\})\ \}$$

and the source S_2 the answer

$$\{\ (1, \{\texttt{Mammifères}\}),\ (3, \{\texttt{Chat}, \texttt{Mammifères}\})\ \}$$

Next, assume that the mediator operates under operation mode 1 (see Table 4.1), that is, the mediator uses the model I_{l-}^- for answering queries. Moreover assume that the mediator returns objects accompanied by their sure indexes (in I_{l-}^-). In this case the mediator will return the following answer:

$$\{\ (1, \{\texttt{Dog}, \texttt{Mammal}, \texttt{Animal}\}),\ (2, \{\texttt{Dog}, \texttt{Mammal}, \texttt{Animal}\}),\ (3, \{\texttt{Mammal}, \texttt{Animal}\})\ \}$$

Let I denote any of the four interpretations I_{l-}, I_{l+}, I_{u-} and I_{u+} of the mediator, and assume that we want to compute $D^-(o)$, i.e. the sure index of some object o, at the mediator. Since the interpretation I is not stored at the mediator we cannot compute $D^-(o)$ like we do for a source (see Prop. 2.3). Instead, we must exploit the articulations a_i and the indexes $D_i(o)$ returned by the sources. Specifically, the mediator can compute $D_l(o)$ (i.e. the index of o with respect to I_l) and $D_u(o)$ (i.e. the index of o with respect to I_u) as stated by the following

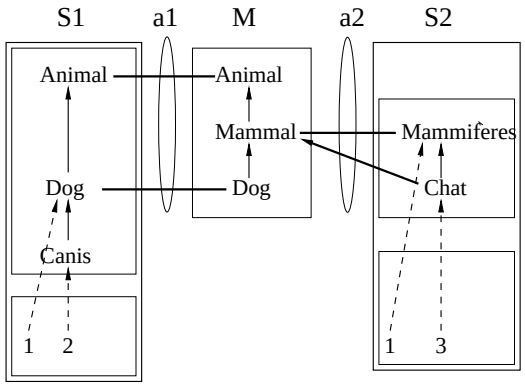


Figure 4.11: A mediator over two sources

proposition. Note that again we write I_l instead of I_{l-} or I_{l+}, and I_u instead of I_{u-} or I_{u+}, since the computation of the object indexes at the mediator does not depend on the evaluation of queries at the underlying sources.

Prop. 4.4

$$D_l(o) = \bigcup_{i=1..k} D_l^i(o), \text{ where } D_l^i(o) = \{t \in T \mid t_i \in D_i(o) \text{ and } t_i\ a_i\ t\}$$

$$D_u(o) = \bigcup_{i=1..k} D_u^i(o), \text{ where } D_u^i(o) = \{t \in T \mid (head_i(t) \neq \emptyset \text{ and } head_i(t) \subseteq D_i(o)) \text{ or } (head_i(t) = \emptyset \text{ and } t_i \in D_i(o) \text{ and } t_i\ a_i\ t)\}$$

Proof.

Consider a mediator over a single source S_i and let o be an object stored at that source. Let $t_i \in D_i(o)$ where I_i is the answer model of the source S_i used by the mediator. If $\exists\ t \in T$ such that $t_i\ a_i\ t$ then certainly $o \in I_l(t)$ (since $I_l(t) = \bigcup\{I(t_i) \mid t_i\ a_i\ t\}$), thus $t \in D_l(o)$. Hence $D_l(o) = \{t \in T \mid t_i \in D_i(o) \text{ and } t_i\ a_i\ t\}$. However, since there are many sources, we denote the right part of the above formula by $D_l^i(o)$, and since an object may belong to more than one source, we arrive at the following: $D_l(o) = \bigcup_{i=1..k} D_l^i(o)$.

Consider again a mediator over a single source S_i and let o be an object stored at that source. If $\exists\ t \in T$ such that $head_i(t) \neq \emptyset$ and $head_i(t) \subseteq D_i(o)$ then certainly $o \in I_u(t)$ (since $I_u(t) = \bigcap\{I(t_i) \mid t\ a_i\ t_i\}$), thus certainly $t \in D_u(o)$. If $\exists\ t \in T$ such that $head_i(t) = \emptyset$ and there is a $t_i \in D_i(o)$ and $t_i\ a_i\ t$ then certainly $o \in I_u(t)$ (since in this case $I_u(t) = \bigcup\{I(t_i) \mid t_i\ a_i\ t\}$). Hence
$D_u(o) = \{t \in T \mid (head_i(t) \neq \emptyset \text{ and } head_i(t) \subseteq D_i(o)) \text{ or } (head_i(t) = \emptyset \text{ and } t_i \in D_i(o) \text{ and } t_i\ a_i\ t)\}$. However, since there are many sources, we denote the right part of the above formula by $D_u^i(o)$, and since an object may belong to more than one source, we arrive at the following: $D_u(o) = \bigcup_{i=1..k} D_u^i(o)$.

◇

Now, $D_l^-(o)$ and $D_u^-(o)$ can be computed just like in the single source case, i.e. by applying Prop. 2.3 to $D_l(o)$ and $D_u(o)$ respectively. Similarly, $D_l^+(o)$ and $D_u^+(o)$ can be computed by applying Prop. 2.4 to $D_l^-(o)$ and $D_u^-(o)$ respectively.

Summarizing, with our mediators the user apart from being able to pose queries in terms of an ontology that was not used to index the objects of the sources being searched, he/she gets an answer comprised of objects which are accompanied by descriptions over the mediator's ontology.

4.7 Mediators with Stored Interpretations

We can easily extend a mediator so as to *also* store an interpretation of its terminology T. Figure 4.12 shows graphically the architecture of a mediator of this kind. Such an extension can prove quite useful in the context of the Web: a Web user can define his own mediator consisting of an ontology that is familiar to him, a set of articulations to other Web catalogs, *and* a stored interpretation of the mediator's ontology. Note that the ontology of the mediator and its stored interpretation resembles the bookmarks facility of Web browsers. However, the addition of articulations now allows the user to browse and query remote catalogs.

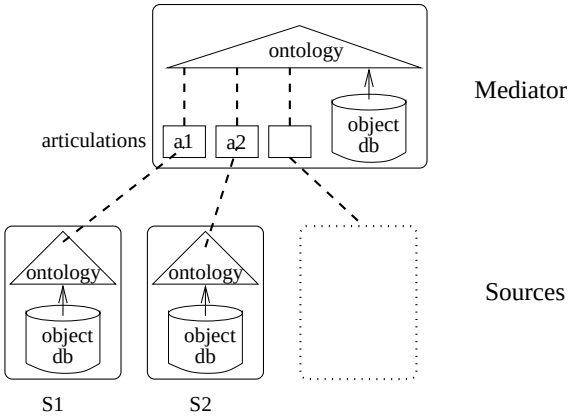


Figure 4.12: The architecture of a mediator with a stored interpretation

Let I_M denote the stored interpretation of T. When a user sends a query to the mediator he has three choices:

(a) he can ask for an answer derived from I_M,

(b) he can ask for an answer derived from the interpretations of the remote sources, or

(c) he can ask for an answer derived from both I_M and the interpretations of the remote sources.

In the first case the mediator operates as a source (see Chapter 2), in the second case it operates as the mediators that we described earlier, while in the third case it again operates as the mediators that we described earlier but with one difference: the interpretations I_{l-}, I_{l+}, I_{u-}, I_{u+} are now defined by taking the union of I_M and the interpretations of the sources. For instance the interpretation I_{l-} is now defined as:

$$I_{l-}(t) = I_M(t) \cup (\bigcup_{i=1}^{k} I_i^-(t_l^i))$$

In the case where the mediator also stores an interpretation I_M of T then the mediator's ability to "translate" the descriptions of the objects returned by the underlying sources drives to an interesting scenario for the Web. Consider a user who has submitted a query to the mediator and assume that the mediator has returned a set of objects to the user. If some of these objects are of real interest to the user (e.g. a set of beautiful images, good papers etc) then the user can store these objects in the database of the mediator. These objects will be stored under terms of the mediator's ontology, i.e. in the interpretation I_M of T.

For example, consider the mediator shown in Figure 4.11. The mediator can store objects 1 and 2 under the terms `Dog`, `Mammal` and `Animal`, and object 3 under the terms `Mammal` and `Animal`

However one can easily see that it suffice to store the objects 1 and 2 under the term `Dog` and the object 3 under the term `Mammal`. More formally for storing an object o in I_M the mediator associates this object with the following terms of T:

$$min_{\preceq_M} D_l(o) \quad \text{or} \quad min_{\preceq_M} D_u(o)$$

4.8 Implementation

A mediator can be implemented using any of a number of data models. For example, using the relational model [28], we can implement a mediator as a database schema consisting of two tables, one for storing the terminology and one for storing the subsumption relation and the articulations.

Alternatively we can store each articulation is a separate table.

The functionality of the mediators described in this chapter presumes that each source can provide a sure answer and a possible answer. However, the ontology-based sources that can be

found in the Web, e.g. Yahoo! or ODP, do not currently provide such answers. This means that the functionality of our mediators cannot be implemented straightforwardly. Nevertheless, we can implement the functionality of our mediators over such sources by employing appropriate *wrappers*.

First note that the ontology and the interpretation of a Web catalog is published as a set of Web pages. For each term t of the ontology there is a separate Web page. This page contains the name of the term, and links pointing to pages which correspond to the terms which are subsumed by t. In addition, the page contains links pointing to the objects, here Web pages, which have been indexed under the term t. However we can employ a wrapper in order to parse each such page and extract the name of the term, the subsumed terms and the indexed objects.

Now, the architecture for implementing the mediators over Web catalogs is shown in Figure 4.13. The key point is that the interpretation of a term t of a source S_i in the sure model I_i^- and in the possible model I_i^+ can be computed at the mediator side. This can be achieved by building an appropriate wrapper for that source. In particular, for computing $I_i^+(t)$ the wrapper will fetch the pages of all terms t' such that $t \preceq_i t'$ and then it will derive $I_i^+(t)$ by computing the intersection $\cap\{I_i^-(t')|t \preceq t'\}$.

According to this architecture our mediators can be implemented by using the standard HTTP protocol.

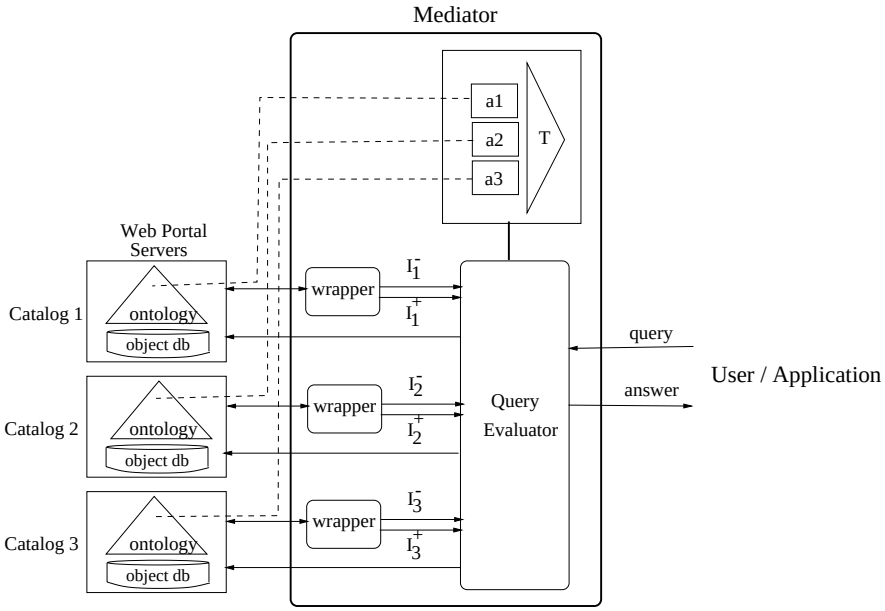


Figure 4.13: An architecture for implementing mediators over the catalogs of the Web

4.9 Related Work

The need for integrated and unified access to multiple information sources has stimulated research on *mediators*. The concept of mediator was initially proposed by Wiederhold [140]. Since then many different approaches have been proposed. In order to compare our approach with the approaches that have been proposed, we first describe a set of layers from which we can view a source, and then we use these layers in order to discuss the kinds of heterogeneity between two sources. Subsequently, we describe the mediator approaches that have been proposed, and finally, we illustrate the distinctive characteristics of our approach.

4.9.1 The Layers of a Source

We can view a source at five different layers: the *domain*, the *conceptualization*, the *conceptual model*, the *data model*, and the *query language* layer. There is a dependency relation between these layers as shown in Figure 4.14. For example, the query language layer of a source depends on the data model layer of the source, and so on.

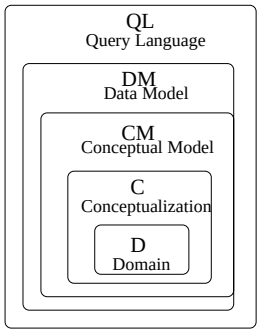


Figure 4.14: The layers of a source

Each source stores information about a part of the real world which we call the *domain* layer of the source. For example, the domain of a source can be the set of all universities, the universities of Greece, or the CSD of the University of Crete.

The *conceptualization* of a domain is the intellectual lens through which the domain is viewed. For example, one conceptualization of the CSD domain may describe its static aspects, i.e. what things exist (e.g. persons, buildings, classrooms, computers), their attributes and their interelationships. Another conceptualization may describe its dynamic aspects in terms of states, state transitions and processes (e.g. enrollments, graduations, attendances, teachings).

A *conceptual model* is used to describe a particular conceptualization of a domain in terms of

a set of (widely accepted) structuring mechanisms (which are appropriate for the conceptualization). For example, a conceptual model that describes the static aspects of the CSD domain, using generalization and attribution, is shown in Figure 4.15, while a conceptual model that describes the dynamic aspects of the CSD domain, using states and state transitions, is shown in Figure 4.16.

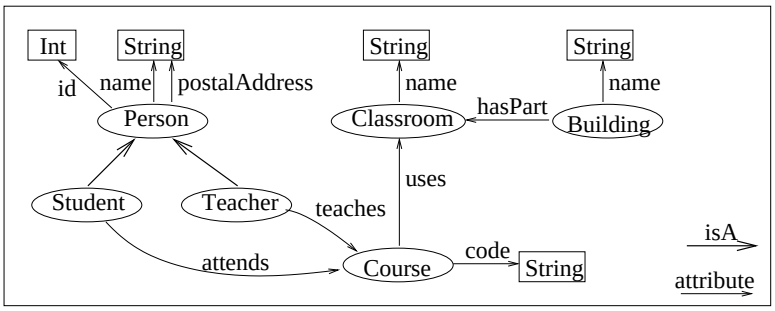


Figure 4.15: A conceptual model describing the static aspects of the CSD domain

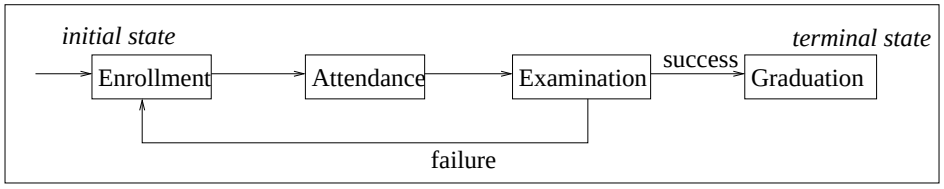


Figure 4.16: A conceptual model describing the dynamic aspects of the CSD domain

Usually, the representation of a conceptual model in a computer is done according to a specific *data model* (e.g. relational, object-oriented, semantic network-based, semistructured). For example, the class `Person` of the conceptual model of Figure 4.15 can be represented in the relational model as a database scheme consisting of the following relation scheme:

`Person(`<u>`id`</u>`:`*`Int`*`, name:`*`Str`*`, postalAddress:`*`Str`*`)`

Alternatively, in a different source, it could be also represented as a scheme consisting of two relation schemes:

`PERSON(`<u>`id`</u>`:`*`Int`*`, name:`*`Str`*`), addressId:`*`Int`*`)`

`POSTALADDRESS(`<u>`id`</u>`:`*`Int`*`, address:`*`Str`*`).`

However there are data models which allow a straightforward representation of the conceptual model. For example, in the semantic network-based data model of SIS-Telos [71], the class `Person` can be represented by the following declaration:

```
TELL Individual Person in S_Class
with attribute
   id: Telos_Integer ;
   name: Telos_String ;
   postalAddress: Telos_String
End
```

Finally, each source can answer queries expressed in a particular query language. For example, a source may respond to Datalog queries, while another may respond only to SQL queries. In this case we say that the *query language layers* of these sources are different.

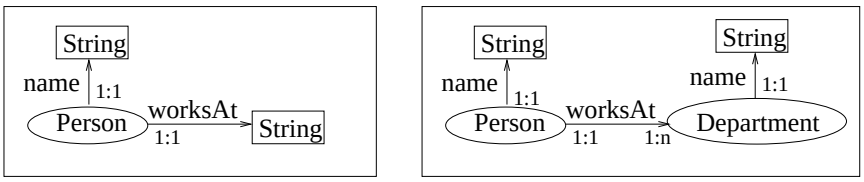
An important remark that we have to mention here is that given an existing source, we usually have in our disposal only its data model and query language layer, and more often than not, from these two layers we cannot infer the conceptual model or the conceptualization layer of the source. For example, suppose a relational schema consisting of the following relation:

PERSON(name:*Str*, worksAt:*Str*).

The underlying conceptual model could be any of the ones shown in Figure 4.17, as the translations of both (a) and (b) to the relational model (by using an algorithm such as the one described in [14]) are identical. Note that according to (a) the domain consists of entities of one kind, i.e. persons, while according to (b) the domain consists of two kinds of entities: persons and departments. Moreover, although two sources may have the same conceptual model, e.g. the conceptual model (a), their representation in the data model may differ. For example the conceptual model (a) could be represented in the relational model as a database scheme consisting of one relation scheme (as we saw before), or as follows:

PERSON(name:*Str*, worksAt:*Int*)

DEP(depId:*Int*, name:*Str*)



(a) (b)

Figure 4.17: Two conceptual models of the CSD domain

4.9.2 Kinds of Heterogeneity

Given a source S_i we will use D_i to denote the domain, C_i the conceptualization, CM_i the conceptual model, DM_i the data model, and QL_i the query language layer of S_i. Consider two sources S_1 and S_2. We may have the following kinds of heterogeneity:

- $D_1 = D_2$, but $C_1 \neq C_2$.

  Here the sources store information about the same domain but conceptualize it differently. For example, C_1 may describe the static aspects, while C_2 the dynamic aspects of the domain.

- $C_1 = C_2$ but $CM_1 \neq CM_2$.

  Here the sources have the same conceptualization of the (same) domain, but they employ different conceptual models. Even if the conceptual models are expressed using the same structuring mechanisms (e.g. generalization, attribution), they may differ due to:

  - *different naming schemes* (also called naming conflicts).
    A frequent phenomenon is the presence of homonyms and synonyms.
  - *different scaling schemes*
    They occur when different reference systems are used to measure a value. For example, 1 foot vs 0.304 meter, 23 $^{\circ}C$ vs 73 $^{\circ}F$.
  - *different levels of granularity*
    For example CM_1 may contain a class `Cameras`, while CM_2 may contain the classes `StillCameras` and `MovingPictureCameras`.
  - *structural differences*
    For example CM_1 may contain a class `Person` having an attribute `owns` with range the class `ArtificialObject`, and a class `Car` isA `ArtificalObject`, while CM_2 may contain a class `Car` having an attribute `owner` with range the class `Person`.

- $CM_1 = CM_2$ but $DM_1 \neq DM_2$.
  Here the sources have the same conceptual model but these models are represented using different data models. Moreover S_1 and S_2 may employ the same data model, e.g. the relational, but DM_1 and DM_2 may differ because they may represent the conceptual model differently.

- $DM_1 = DM_2$ but $QL_1 \neq QL_2$.
  Here the sources have the data model layer, but they support different query languages.

For example S_1 may respond to Datalog queries, while S_2 may respond only to SQL queries.

As we can see we can have several forms of heterogeneity between two sources.

4.9.3 Kinds of Mediators

A *mediator* is a device that aims at providing a uniform query interface to several sources. It is a device that expresses how the integration of multiple sources is achieved. There are several approaches for building mediators over relational databases (e.g. see [77, 46, 47, 144]), SGML documents (e.g. see [27]), information retrieval systems (e.g. see [138, 51, 43, 127, 97]) and Web-based sources (e.g. see [9, 23, 24]). As information retrieval systems accept free queries, i.e. natural language queries, a mediator over this kind of sources does not have to translate the mediator queries, therefore we omit this kind of mediators from our discussion.

In general, the mediator architecture consists of a *global/mediator view* describing the conceptual model in terms of the data model (or logical formalism) employed, *source descriptions* and *wrappers* which describe the contents and/or the querying capabilities of each source, and an *exchange data model*, which is used to convey information among sources. The mediator accepts queries expressed with respect to the mediator view. Upon receiving a user query, the mediator queries the underlying sources. This involves selecting the sources to be queried and formulating the query to be sent to each source. These tasks are accomplished based on what the mediator "knows" about the underlying sources. Specifically, source descriptions are exploited for translating the mediator (or global) queries to source-specific queries. Finally , the mediator appropriately combines the returned results and delivers the final answer to the user. Figure 4.18 shows the functional overview of the mediator.

There are many different approaches for implementing this general architecture each one with different applicability, advantages and limitations.

Various data models are used for representing the mediator's view, i.e. the conceptual model of the mediator. For example, the relational model (e.g. Infomaster [48], [38]), semantic network-based models (e.g. SIMS [69]), F-logic (e.g. OntoBroker [12], [33]), description logics (e.g. Information Manifold [77], OBSERVER [87],[66], [67], PICSEL [73]). Most of the approaches assume a single mediator's view, however there are cases where more than one conceptual models are allowed. For example, in OBSERVER the mediator's ontology (conceptual model) is obtained by merging the conceptual models of the underlying sources. A different approach is taken in TSIMMIS [25], [46] where the mediator's view is a set of *query templates*, and in

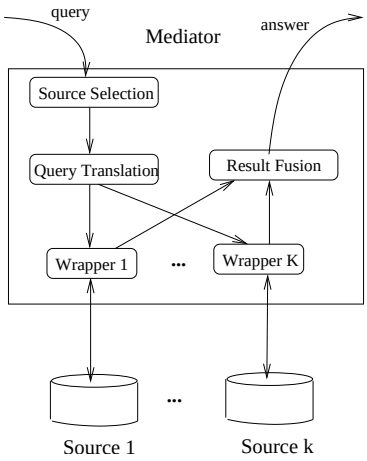


Figure 4.18: Functional Overview of the Mediator

HERMES where the mediator's view is a set of rules combining the results of predefined calls to the underlying sources.

Source descriptions describe the contents of the sources. They are exploited for translating the mediator (or "global") queries to source-specific queries. In the relational mediators (see [47] for a review) the mediator's view is a relational database schema. Concerning source descriptions, we can distinguish the *local-as-global* and the *global-as-local* approach. In the *local-as-global* approach the source relations are defined as relational views over the mediator's relations, while in the *global-as-local* approach the mediator relations are defined as views of the source relations. The former approach offers flexibility in representing the contents of the sources, but query answering is "hard" because this requires answering queries using views ([37],[76], [135]). On the other hand, the global-as-local approach offers easy query answering (expansion of queries until getting to source relations), but the addition of a new source implies rewriting the global view. The local-as-global approach is taken in the systems Information Manifold, Infomaster, while the global-as-local approach is taken in the system TSIMMIS, as the queries supported by the mediator are expressed with respect to the query templates exported by the wrappers of the sources. In PICSEL project the CARIN language (a language combining Datalog rules and Description Logics) is used for representing the mediator's view and the contents of the sources. Specifically, each underlying source is described by a set of implicative sentences indicating which kinds of data allowed being extracted, and a number of DL assertions expressing constraints that are known on the data that can be obtained from the source. In TSIMMIS, each source is described by a set of query templates which are supported by the source, while in HERMES

each source is modelled as a set of functional calls.

The data model or data structure used to convey information between the mediator and the sources varies. It may have the form of tuples (e.g. in Infomaster, SIMS, Information Manifold, OBSERVER), or it may be tuples that encode graph data structures (like the OEM in TSIMMIS, or the YAT in [27]). In Web-applied approaches, Web pages are also used as containers for exchanging information.

4.9.4 Relevant System and Projects

Below we shortly describe some interesting and indicative information integration/mediation systems and projects:

- *HERMES* ([121]). The Heterogeneous Reasoning and Mediator System is a system which approaches integration from a software engineering point of view. Each source is described as a set of domain calls. The mediator's language (a rule-based language) allows the declarative combination of these calls and the handling of the conflicts which may arise by various criteria (e.g. recency, preference, etc).

- *Infomaster* ([48], [38]). Infomaster demonstrates an approach for integrating relational databases. The mediator's view is a relational schema and translation rules are used to describe how each source relates to this schema, and how the clients want to access the data. The transformation rules allow attribute renaming, case-based attribute value conversion, and the usage of mathematical expressions for attribute value conversion and for synthesizing two or more attributes.

- *SIMS* ([69]). It can be characterized as an agent-based approach as it proposes a network of cooperating information retrieval agents. Each agent carries a detailed model of its own expertise (called *domain model*), and models of other agents and information sources that can provide relevant information (called *information source models*). Both types of models are represented in the LOOM knolwedge representation language. The domain model consists of classes, subclasses and relations, while an information source model describes the contents of the information source and the relationships between this model and the domain model (for enabling the transformation a domain-level query into a set of queries to actual information sources). The communication between the information agents is done using KQML [42]. The problem of source selection is faced as a search problem, and query processing is faced as a planning problem.

- *Information Manifold* ([77]). It is a system for browsing and quering multiple networked information sources, mainly relational databases. The mediator's view is expressed in a object-relational data model, a combination of the relational model with an object-oriented model (using CLASSIC). The source description language enables (a) relating the contents of a source to the mediator's view, (b) expressing that a source relation contains complete information about a fragment of the mediator's view, and (c) specifying the (query) capabilities of a source. Focus is given on minimizing, at run-time, the number of sources to be queried.

- *TSIMMIS*([25], [46]). TSIMMIS takes an operational approach of encapsulating sources. It approaches the integration problem by defining a list of query-templates. For each template, a query plan expressing how to evaluate a query of this form is given. The wrapper of a source describes the query capabilities of the source, the objects made available by the source, and the conditions that must be satisfied by source objects. When the mediator receives a query, it finds those predefined templates that match the user query, and executes the corresponding stored query plans. With this approach the number of possible queries the user can ask is limited, and adding a new source to the system requires recoding the related query plans. The exchange of information between the mediator and the sources is based on the Object Exchange Model (OEM), a lightweight self describing data model (it does not depend on a schema), which allows simple nesting, hence enables simulating many information structures (relational, object-oriented, etc). The Mediator Specification Language (MSL) enables declarative specification of wrappers and mediators in terms of OEM.

- *OBSERVER* ([87],[66], [67]). (Ontology Based System Enhanced with Relationships for Vocabulary hEterogeneity Resolution). In this system multiple pre-existing ontologies are used to access heterogeneous, distributed and independently developed sources. The integration of a new source is achieved by mapping ontological concepts to the data structures of the source. The ontologies, as well as the interontology relations (intentional and extensional), are represented in description logics. Queries are rewritten using these interontology relationships to obtain translations across ontologies.

- *PICSEL project* ([73]). In PICSEL project the mediator uses the CARIN language (a language combining Datalog rules and Description Logics) for representing the mediator's view and the contents of the sources. CARIN is used in a way that mixes the global-as-local and local-as-global approach, so as to avoid the query reformulation problem. Each

underlying source is described by a set of implicative sentences indicating the kinds of data to be extracted, and a number of description logic assertions expressing constraints that are known on the data that can be obtained from the source.

- A system for the design of data conversion programs which can serve as the basis for building a mediator/wrapper system is described in [27]. It provides tools for the specification and implementation of data conversions among heterogeneous data sources. It relies on the YAT model, a data model based on named trees with ordered and labelled nodes. A rule-based language offering pattern matching and restructuring facilities is provided for specifying the conversions. Note that many integration approaches rely on graph or tree data models as one can easily map anything into a tree or a graph.

- *ONION* ([93], [81]). The ONION (ONtology compositION) system aims at providing a scalable achitecture for ontology integration. It is based on the ideas of Wiederhold [141] about an algebra of ontologies. In particular, given two ontologies O1 and O2 and a number of mathing rules M, he proposed three algebraic operators (intersection, union and difference). A graph-based data model is used for representing the ontologies. The matching/articulation rules are either semantic implication links, or functional rules (e.g. conversion functions) between terms or graph patterns of the underlying ontologies. These rules are represented using the graph-based data model too, and form "articulation ontologies" which do not stand on their own as ontologies. The rules are generated by a semi-automatic articulation tool with the help of a domain expert.

- One approach that considers approximate translations is [23, 24]. The queries considered there are boolean expressions of constraints of the form *[attr1 op value]* or *[attr1 op attr2]* and mapping rules are employed in order to handle differences in operators, data formats and attribute names. The translated queries minimally subsume the original ones.

4.9.5 Comparison with other Mediator Approaches

Let us now compare our work with other mediator approaches.

- Relational mediators have some critical differences with our mediators. Relational mediators and their sources are *schema-based* while our mediators and their sources are *ontology-based*. Also recall that the relational model is value-based and not object-based. This means that the conceptualization and the conceptual model of a source is hidden, or not clear. Therefore the mediators over such sources "work" on the DM layer.

Relational mediators try to construct *exact translations* of SQL queries while our mediators allow *approximate translations* of boolean expressions through their articulations. We could say that the answers returned by a relational mediator, correspond to the answers returned by an ontology-based mediator in the I_{l-}^{-} model.

Moreover, in several approaches (e.g. in Infomaster) a predicate corresponding to a source relation, can appean only in the head or in the tail of a rule. This means that granularity heterogeneities cannot be tackled easily.

- A different approach to mediators can be found in [21] which presents the fundamental features of a declarative approach to information integration based on Description Logics. The authors describe a methodology for integrating relational sources and they resort to very expressive logics in order to bridge the heterogeneities between the unified view of the mediator and the source views. However the reasoning services for supporting translations have exponential complexity, as opposed to the complexity of our mediators which is polynomial.
- The difference between our approach and OBSERVER's approach is that OBSERVER requires merging the ontologies of all underlying sources. Instead, we just articulate the ontologies of the sources with the ontology of the mediator. Moreover, the compatibility condition introduced here allows the mediator to draw conclusions about the structure of a source ontology without having to store that ontology.
- In [23, 24] the translated queries minimally subsume the original ones. However the functionality offered by our mediators is different, firstly because we support negation while they do not, and secondly because our mediators support multiple operation modes, one of which is the case where the translated queries subsume the original ones.
- An alternative solution to the problem of query relaxation in mediators is described in [6]. If the submitted query yields no answer then the mediator provides an answer to a similar query. The selection of this query is based on a measure of similarity between the concepts and the predicates which is based on the taxonomic structure of the mediator's ontology.

4.10 Summary and Conclusion

We have presented an approach for providing uniform access to multiple ontology-based sources through mediators that render the heterogeneities (naming, contextual, granularity) of the

sources transparent to users. A user of the mediator apart from being able to pose queries in terms of an ontology that was not used to index the objects of the sources being searched, he/she gets an answer comprised of objects which are accompanied by descriptions over the mediator's ontology.

A mediator is seen as just another source but *without* stored interpretation. An interpretation for the mediator is defined based on the interpretations stored at the sources and on the *articulations* between the mediator and the sources; and in fact, we have seen *eight* different ways for defining a mediator interpretation depending on the nature of the answers that the mediator provides to its users (see Table 4.1). Since the resulting mediator models are ordered our mediators can use them in order to support a form of *query relaxation*. As for the articulations that we consider, they can be defined by humans, but they can also be constructed automatically or semi-automatically in some specific cases. A data driven method for articulating ontologies is given in Chapter 5.

The key characteristics and advantages of our approach are the following:

- We consider that the domain of all sources is the Web, and that each source has the *same* conceptualization of the domain. Specifically, each source views the Web as a set of objects Obj (URLS), and stores information about a subset of it (i.e. $O_i \subseteq O$). This means that each object has a *unique identity* over all sources. From this point of view, we could call our mediators object-oriented in contradistinction with the mediators over relational sources which we could call value-oriented, as the relational model is a value-oriented data model.

- We consider that the conceptual model of each source is a triple $(T, \preceq, I)$. This conceptual modeling approach has several advantages if the conceptualization is fixed (i.e. a denumerable set of objects): (a) it is very easy to create the conceptual model of a source or a mediator, (b) the integration of information from multiple sources can be done very easily. Indeed, the articulations offer a *uniform* method to bridge naming, contextual and granularity heterogeneities between the conceptual models of the sources. Given this conceptual modeling approach, the mediator does not have to tackle complex structural differences between the sources (as it happens in relational mediators). Moreover, it allows the integration of *schema* and *data* in a uniform manner. For example consider a source S having the conceptual model shown in Figure 2.3.(a), and a source S' having the conceptual model shown in Figure 2.3.(b), and suppose that both sources are implemented in the relational model. In source S the concept `wood` will be represented in the data level (it would be an element of the domain of an attribute), while in S' it would be a relation.

- We consider that each source can answer queries where a query is a boolean expression of its terms. However we can also handle sources which do not respond to queries of this form by either constructing a wrapper for each such source, or by extending the definition of articulations. For example, consider a source S_i implemented in the relational model (as described in section 2.6) and suppose that this source can answer only pure SQL queries. In this case the articulation a_i may contain relationships of the form `Cameras` $\preceq_{a_i} q_i$ where $q_i = \Pi_{object}(\sigma_{term-name="Cameras"}(ODB \bowtie TERMINOLOGY))$.

Summarizing, the advantages and novelties of our approach are the following:

- Easy construction

  A mediator can be constructed very easily even by ordinary Web users. Indeed, the simple conceptual modeling approach that we adopt makes the definition of the mediator's ontology and articulations very easy.

- Query Relaxation

  In many cases a query to a mediator yields no answer. The sure and the possible answers of sources, as well as the several modes of operation of a mediator, offer a solution to this problem.

- Efficient Query Evaluation

  The time complexity of query translation at the mediator is linear with respect to the size of the subsumption relations of the mediator.

- Scalability

  Articulation (instead of merging) enables a very natural, incremental evolution of a network of sources. Note that the ontologies employed by Web catalogs contain very large numbers of terms (e.g. the ontology of Open Directory contains 300.000 terms). Therefore the *articulation* of ontologies has many advantages compared to ontology *merging*, as such merging would introduce storage and performance overheads. In addition, full merging is a laborious task which in many cases does not pay-off because the integrated ontology becomes obsolete when the ontologies involved change. Another problem with full merging is that it usually requires full consistency, which may be hard to achieve in practice, while articulation can work on locally consistent parts of the ontologies involved. Note that the ontologies considered in this chapter present no consistency problems. There may only be long cycles of subsumption relationships, which induce big classes of equivalent terms.

- Applicability

Our approach provides a flexible and formal framework for integrating data from several sources, and/or for personalizing the contents of one or more sources. One can easily see how our approach allows users of the Web to define *personal views* over existing Web catalogs. Indeed, the ontologies considered fit quite well with the content-based organizational structure of Web catalogs (e.g. Yahoo!, Open Directory), keyword hierarchies (e.g. ACM's thesaurus) and personal bookmarks. By defining a mediator, the user can employ his own terminology in order to access and query several Web catalogs, specifically those parts of the catalogs that are of interest to him.

Moreover, as a mediator can also have a stored interpretation, our approach can drive to a network of articulated sources. Also recall that a mediator can translate the descriptions of the objects returned by the underlying sources. This implies that all (or some) of these objects can be straightforwardly stored in the mediator base (under terms of the mediator ontology).

Chapter 5

Articulating Ontologies

The mediators defined in Chapter 4 are based on articulations. An articulation between two ontologies can be defined by humans. In practice, ontology articulation is a laborious task especially when the involved ontologies consist of many terms. Therefore it is worth investigating methods that allow the automatic, or semi-automatic, construction of an articulation. This automation or assistance is possible in some specific cases. For instance, if the ontologies involved are *materialized*, i.e. if there are objects which are indexed under the terms of two or more ontologies, then we can exploit the stored objects in order to assist the designer in articulating the ontologies.

In this chapter we describe a method for assisting a human in articulating the ontologies of two sources. The algorithm that we present is based on the objects that are indexed under *both* ontologies. The distinctive features of the proposed method are:

(a) it is *independent* of the nature of the objects, i.e. the objects may be images, audio, video, etc.,

(b) it can be used in order to articulate terms and *queries*, and

(c) it can be implemented efficiently by a communication protocol, thus the involved sources can be *distant*.

The remaining of this chapter is organized as follows: Section 5.1 discusses related work and the motivation for our approach. Section 5.2 gives a short example illustrating the approach. Section 5.3 presents the definitions needed, and Sections 5.4 and 5.5 describe the proposed method for articulating two ontologies. Section 5.6 discusses the role of a training set, and Section 5.7 discusses applications. Finally, Section 5.8 concludes the chapter.

5.1 Ontology Integration / Matching

In general, ontology integration is an open research topic which includes issues such as the usefulness of integration, the semantics of interontology links, and the methods/processes for obtaining such links (e.g. see [102, 92, 22, 98]). Usually, ontology integration is done by domain experts (e.g. in the case of multilingual thesauri), with or without the aid of software tools.

Finding semantic mappings between ontologies is a also key challenge in building the *Semantic Web* [13] that has received relatively little attention. Given the de-centralized nature of the development of the Semantic Web, there will be an explosion in the number of ontologies (see for example [26]). Many of these ontologies will describe similar domains, but using different terminologies, and others will have overlapping domains. To integrate or exchange data from disparate ontologies, we must know the *semantic correspondences* between the elements of the ontologies.

According to [8] the approaches for linking two ontologies (in particular thesauri) can be broadly classified as either *model*-driven or *data*-driven. The model-driven approach starts with a (theoretical) model of how the two ontologies are constructed and how they are used. Based on this model, techniques can be developed for linking the two ontologies. However, a 1-1 matching (articulation) of the two models would generally require that the two ontologies have the same level of specificity. This is often not the case. Additionally, the model-driven approach has to address compatibility issues between the two ontologies, structural differences and semantic heterogeneity. Finally, the model-driven approach requires us to provide a semantic interpretation for the terms in the ontologies which may differ from how the terms are interpreted by the indexer of the corresponding source. Software tools are usually employed in order to assist the designer in articulating two ontologies. These tools usually rely on lexical resources. For instance, SKAT [91] is a system for articulating ontologies (within the integrated environment ONION [93]) in which the domain expert provides a set of intentional links between the ontologies (they call these links expert rules) and the system exploits semantic lexicons (e.g. WordNet [29]) in order to locate and propose to the expert other articulations for verification. A method for integrating thesauri based on the similarity of names and the structure of the terms is given in [116], while an approach for merging RDF ontologies and thesauri in order to derive a richer ontology is described in [7].

An alternative to the model-driven approach is the *data-driven* approach. In this approach, the relationships between terms in the two ontologies are *discovered* by examining how these terms

are used in indexing the objects. The advantage of such an approach is that it does not make any assumptions on how the two ontologies are constructed, or how they are used. All it requires is the presence of two databases that contain several objects in common. Valid links are discovered on the basis of how the terms are used in the individual databases. However, the data-driven approach does have inherent difficulties. First, unless one has a large collection of objects that have been indexed using both ontologies, spurious correlation can result in inappropriate linking. Second, if a term is not assigned to any of the common objects, one cannot establish a link for that term. Third, rarely occurring terms can result in statistically insignificant links. Finally, the validation of data-driven approaches can only be statistical in nature. In spite of these inherent difficulties, data-driven approaches can be formalized and automated. The problem of spurious links can be largely alleviated if there is a large collection of common objects. Most of the data-driven approaches that can be found in the literature are applicable only if the domain is a set of documents (texts) (e.g. [8, 59, 34, 72, 111]). For instance, a data-driven approach for linking thesauri is described in [8]. The proposed method employs statistical analysis and term clustering in order to relate each term of one thesaurus with a set of terms from another thesaurus. Another method for automatic articulation that relies on the co-occurrence of terms in a parallel corpus of texts is described in [59]. A further method can be found in [34]. Given two ontologies, this method finds, for each term in one ontology, the most similar term in the other ontology, based on probabilistic metrics.

Below we present a method which is independent of the nature of the objects, i.e. the objects may be images, audio, video, etc, and can be used in order to articulate the desired parts of an ontology.

5.2 Our Approach in Brief

Consider two ontology-based sources $S_1 = [(T_1, \preceq_1), I_1]$ and $S_2 = [(T_2, \preceq_2), I_2]$ as shown in Figure 5.1. Suppose that we want to articulate some (or all) terms of S_1 with terms of S_2 in order to define an articulation denoted $a_{1,2}$. For example, suppose that we want to articulate the term `green`. One reasonable way to articulate this term is to store, in the articulation $a_{1,2}$, the following two relationships:

- `green` $\succeq$ `cabbages`
  because the interpretation of `green` in S_1 is a superset of the interpretation of `cabbages` in S_2, i.e. $\{1, 2, 3\} \supseteq \{1, 2\}$, and

- `green` $\preceq$ `cabbages` $\vee$ `tomatoes`

  because the interpretation of `green` in S_1 is a subset of the interpretation of `cabbages` $\vee$ `tomatoes` in S_2, i.e. $\{1,2,3\} \subseteq \{1,2,3,4\}$

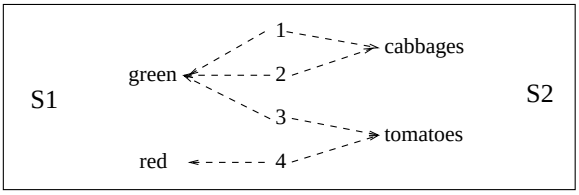


Figure 5.1: An example of two sources over a common domain

Specifically, in order to articulate a term a of S_1 we can follow the following steps: The first step is to compute the interpretation of a in S_1. The second step is to compute to "smallest" query of S_2 whose interpretation contains the set $I_1(a)$, call it b^+, and the "biggest" query of S_2 whose interpretation is contained in the set $I_1(a)$, call it b^-. The third, and last, step is to establish one or more relationships between the term a and the queries b^+, b^- by comparing the interpretation of a (in S_1) with the interpretations of b^+ and b^- (in S_2).

If, however, the two sources are distant, then in order to establish such relationships, we need a "protocol" describing what S_1 should send to S_2 how S_2 should reply, and what S_1 should understand from the response of S_2.

The next section describes how a source computes the queries that are needed for this protocol, while Section 5.4 presents the protocol as a whole.

5.3 Preliminaries

Let $[(T, \preceq), I]$ be a source. We use Q_T to denote the (infinite) set of all queries that can be formed using terms of T, the logical connectives $\wedge, \vee$, and parentheses. We exclude negation for reasons that will become clear in the sequel. Clearly Q_T is an infinite set.

Def 5.1 Let I be an interpretation over Obj. The *active domain* of I, denoted by $adom(I)$, is the set of all objects with nonempty index in I , that is:

$$adom(I) = \{\, o \in Obj \mid D_I(o) \neq \emptyset \}$$

For example, the active domain of the sources S_1 and S_2 shown in Figure 5.1 is the set $\{1,2,3,4\}$.

Def 5.2 Let $S = [(T, \preceq), I]$ be a source, and let K be a nonempty subset of $adom(I)$. The *upper closure* and the *lower closure* of K with respect to Q_T, denoted by K^+ and K^- respectively, are defined as follows:

$$\begin{aligned} K^+ &= \{q \in Q_T \mid K \subseteq I^-(q)\} \\ K^- &= \{q \in Q_T \mid K \supseteq I^-(q)\} \end{aligned}$$

For the source shown in Figure 5.2, if $K = \{1, 2\}$ then K^+ includes the queries `tomatoes`, `vegetables, tomatoes` $\vee$ `bananas, foods, foods` $\wedge$ `tomatoes`, etc. Notice that $K \subseteq I^-(\top)$, for every K, thus the top element $\top$ belongs to K^+, for every K. Therefore, K^+ is always nonempty. It follows that K^+ is an infinite set as $\top$, $\top \vee \top$, $\top \wedge \top \wedge \top$, and so on, belong to K^+.

In the source of the same figure, if $K = \{1, 2, 3\}$ then K^- includes the queries `tomatoes`, `vegetables, tomatoes` $\vee$ `vegetables, tomatoes` $\wedge$ `vegetables`, etc. Notice that $I^-(\bot) \subseteq K$, for every K, thus the bottom element $\bot$ belongs to K^-, for every K. Therefore, K^- is always nonempty. It follows that K^- is an infinite set as $\bot$, $\bot \vee \bot$, $\bot \wedge \bot \wedge \bot$, and so on, belong to K^-.

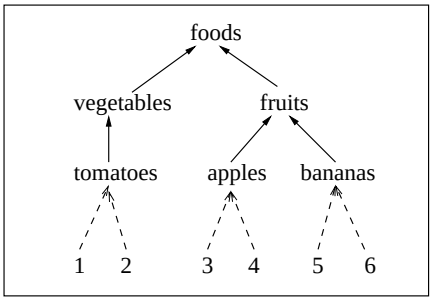


Figure 5.2: Example of a source

As our objective is the creation of an articulation, we consider queries which contain conjunctions and disjunctions and no negations, because we prefer storing "positive" information in an articulation. Furthermore, if we had considered queries with negation then K^+ and K^- would contain long queries which do not seem very useful for our purposes. For example if $K = \{1, 2\}$ then K^+ and K^- would include the queries `tomatos` $\wedge\neg$ `apples`, `tomatos` $\wedge\neg$ `(apples` $\vee$ `bananas)`, etc.

Let q and q' be two queries in Q_T. We write $q \leq q'$ if $I(q) \subseteq I(q')$ in every model I of $(T, \preceq)$. We write $q \sim q'$, if both $q \leq q'$ and $q' \leq q$ hold. Note that $\sim$ is an equivalence relation over Q_T. Let EQ_T denote the set of equivalence classes induced by $\sim$ over Q_T. We can extend the

relation $\leq$ over the set EQ_T as follows: for all c, c' in EQ_T, $c \leq c'$ if there is a $q \in c$ and a $q' \in c'$ such that $q \leq q'$. Note that $\leq$ is a partial order over EQ_T.

The partially ordered set $(EQ_T, \leq)$ is a *lattice* as every pair of elements c, c' of EQ_T have a greatest lower bound (for short glb) and a least upper bound (for short lub). Specifically the glb of two elements c, c' is the element $c \wedge c'$ and we know that it certainly exists by the construction of Q_T. Similarly, the lub of two elements c, c' is the element $c \vee c'$. Also notice that the set EQ_T is finite. This holds because each equivalence class of EQ_T corresponds to exactly one set of objects: the common interpretation of all queries in the class. As Obj is a finite set there is a finite set of subsets, hence EQ_T is finite. This means that the partially ordered set $(EQ_T, \leq)$ is a complete lattice.

Def 5.3 Let $S = [(T, \preceq), I]$ be a source, and let K be a nonempty subset of $adom(I)$. The *upper name* and the *lower name* of K with respect to S, denoted by $name^+$ and $name^-$ respectively, are defined as follows:

$$\begin{aligned} name^+(K) &= glb(K^+) \\ name^-(K) &= lub(K^-) \end{aligned}$$

As $(EQ_T, \leq)$ is a complete lattice, $glb(K^+)$ and $lub(K^-)$ exist, thus we have the following proposition:

Prop. 5.1 Every subset K of $adom(I)$ has an *upper* and a *lower name*, namely:

$$\begin{aligned} name^+(K) &= \bigwedge_{c \in K^+} c \\ name^-(K) &= \bigvee_{c \in K^-} c \end{aligned}$$

Now, as the query $\bigwedge_{c \in K^+} c$ is of infinite lenght, we need a method to compute a query of finite lenght which is equivalent to the query $name^+(K)$. For the same reason, we need a method to compute a query of finite lenght which is equivalent to the query $name^-(K)$.

Recall from Chapter 2 that the index of an object o with respect to an interpretation I, denoted by $D_I(o)$, is defined as follows:

$$D_I(o) = \{t \in T \mid o \in I(t)\}$$

Hereafter we shall also use $D_I(o)$ to denote the conjunction $\bigwedge\{t \in T \mid o \in I(t)\}$.

Theorem 5.1 $\bigvee_{o \in K} D_I(o) \sim name^+(K)$

The following two propositions prove this theorem.

Prop. 5.2 The query $\bigvee_{o \in K} D_I(o)$ is a lower bound of K^+.
Proof:

Let x denote the query $\bigvee_{o \in K} D_I(o)$. We will prove that x is a lower bound of the set K^+, i.e. we will prove $x \leq y$ for each $y \in K^+$.

Since $K \subseteq I^-(y)$, for each $o \in K$, $o \in I^-(y)$. Recall that each $o \in K$ is indexed under the set of terms $D_I(o)$. This implies that it must be $y \geq D_I(o)$ otherwise o would not be an element of $I^-(y)$. Thus $y \geq \bigvee_{o \in K} D_I(o)$, i.e. $y \geq x$.

$\diamond$

Prop. 5.3 $\bigvee_{o \in K} D_I(o) \in K^+$
Proof:

Each $o \in K$ is an element of $I(t)$ for each $t \in D_I(o)$. Thus o is an element of $I(D_I(o))$. This implies that $K \subseteq \bigcup\{I(D_I(o)) \mid o \in K\} = I(\bigvee_{o \in K} D_I(o))$. Since $I \sqsubseteq I^-$, we infer that $I(\bigvee_{o \in K} D_I(o)) \subseteq I^-(\bigvee_{o \in K} D_I(o))$, thus $\bigvee_{o \in K} D_I(o) \in K^+$.

$\diamond$

Since the query $\bigvee_{o \in K} D_I(o)$ is a lower bound of K^+ and it is an element of K^+, it follows that this query is the glb of K^+. It is also clear that this query has finite lenght, hence it is the query that we are looking for. For this purpose, herafter we use $name^+(K)$ to denote the query $\bigvee_{o \in K} D_I(o)$.

For the source shown in Figure 5.2 we have:

$$\begin{aligned}
name^+(\{1\}) &= \texttt{tomatoes} \wedge \texttt{vegetables} \wedge \texttt{foods} \\
name^+(\{1,2\}) &= (\texttt{tomatoes} \wedge \texttt{vegetables} \wedge \texttt{foods}) \vee (\texttt{tomatoes} \wedge \texttt{vegetables} \wedge \texttt{foods}) \\
name^+(\{1,2,3\}) &= (\texttt{tomatoes} \wedge \texttt{vegetables} \wedge \texttt{foods}) \vee (\texttt{tomatoes} \wedge \texttt{vegetables} \wedge \texttt{foods}) \vee \\
&\quad (\texttt{apples} \wedge \texttt{fruits} \wedge \texttt{foods})
\end{aligned}$$

Note that it is equivalent to consider that $D_I(o) = \bigwedge\ min(\{t \in T \mid o \in I(t)\})$, instead of $D_I(o) = \bigwedge\ \{t \in T \mid o \in I(t)\}$. It is also equivalent to use $\bigvee_{o \in K}\ max(D_I(o))$ instead of $\bigvee_{o \in K} D_I(o)$.

Doing these "simplifications", we have:

$$\begin{aligned}
name^+(\{1\}) &= \texttt{tomatoes} \\
name^+(\{1,2\}) &= \texttt{tomatoes} \\
name^+(\{1,2,3\}) &= \texttt{tomatoes} \vee \texttt{apples} \\
name^+(\{3,4,5,6\}) &= \texttt{apples} \vee \texttt{bananas}
\end{aligned}$$

Let us now try to find a finite length query which is equivalent to $name^-(K)$.

Theorem 5.2 $\bigvee\{\ D_I(o) \mid o \in K,\ I(D_I(o)) \subseteq K\} \sim name^-(K)$

The following two propositions prove this theorem.

Prop. 5.4 The query $\bigvee\{\ D_I(o) \mid o \in K,\ I(D_I(o)) \subseteq K\}$ is an upper bound of K^-.
Proof:

Let x denote the query $\bigvee\{\ D_I(o) \mid o \in K,\ I(D_I(o)) \subseteq K\}$. We will prove that x is an upper bound of the set K^-, i.e. we will prove $x \geq y$ for each $y \in K^-$.

Suppose that there is an object $o \in I^-(y)$ such that $o \notin I^-(x)$. Since o is indexed under the set of terms $D_I(o)$, it must be $y \geq D_I(o)$ otherwise o would not be an element of $I^-(y)$. If $I(D_I(o)) \subseteq K$ then certainly o would be an element of $I^-(x)$. So, let us suppose that $I(D_I(o)) \not\subseteq K$. In this case $y \geq D_I(o) \Leftrightarrow I^-(y) \supseteq I(D_I(o))$. As $I(D_I(o)) \not\subseteq K$ we infer that $I^-(y) \not\subseteq K$ which is a contradiction. Thus the hypothesis $o \notin I^-(x)$ is not valid, hence x is an upper bound of K^-.

$\diamond$

Prop. 5.5 $\bigvee\{\ D_I(o) \mid o \in K,\ I(D_I(o)) \subseteq K\} \in K^-$
Proof:

If $I(D_I(o)) \subseteq K$ then $\bigcup\{I(D_I(o)) \mid o \in K,\ I(D_I(o)) \subseteq K\} \subseteq K$. Thus $\bigvee\{\ D_I(o) \mid o \in K,\ I(D_I(o)) \subseteq K\} \in K^-$.

$\diamond$

Since the query $\bigvee\{\ D_I(o) \mid o \in K,\ I(D_I(o)) \subseteq K\}$ is an upper bound of K^- and it is an element of K^-, it follows that this query is the lub of K^-. It is also clear that this query has finite lenght, hence it is the query that we are looking for. For this purpose, hereafter we use $name(K)^-$ to denote the query $\bigvee\{\ D_I(o) \mid o \in K,\ I(D_I(o)) \subseteq K\}$.

If the set $\{o \in K,\ I(D_I(o)) \subseteq K\}$ is empty then we consider that $name(K)^- = \bot$.

For the source shown in Figure 5.2 we have:

$$\begin{aligned} name^-(\{1\}) &= \bot \\ name^-(\{1,2\}) &= (\texttt{tomatoes} \wedge \texttt{vegetables}) \vee (\texttt{tomatoes} \wedge \texttt{vegetables}) \\ name^-(\{1,2,3\}) &= (\texttt{tomatoes} \wedge \texttt{vegetables}) \vee (\texttt{tomatoes} \wedge \texttt{vegetables}) \vee (\texttt{tomatoes} \wedge \texttt{vegetables}) \end{aligned}$$

By doing analogous simplifications, as with $name^+$, we now have:

$$\begin{aligned} name^-(\{1\}) &= \bot \\ name^-(\{1,2\}) &= \texttt{tomatoes} \\ name^-(\{1,2,3\}) &= \texttt{tomatoes} \\ name^-(\{4,5,6\}) &= \texttt{bananas} \end{aligned}$$

If we have multiple classification, i.e. if an object can be indexed under more than one terms, then the upper and lower name of a set of objects can be a more complex query, specifically a disjunction of conjunctions. Some examples from the source shown in Figure 5.3 follow:

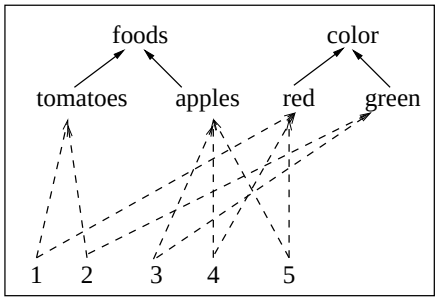


Figure 5.3: Example of a source with multiple classification

$$
\begin{aligned}
name^{+}(\{1,3\}) &= (\texttt{tomatoes} \wedge \texttt{red}) \vee (\texttt{apples} \wedge \texttt{green}) \\
name^{-}(\{1,3\}) &= (\texttt{tomatoes} \wedge \texttt{red}) \vee (\texttt{apples} \wedge \texttt{green}) \\
name^{+}(\{1,3,5\}) &= (\texttt{tomatoes} \wedge \texttt{red}) \vee (\texttt{apples} \wedge \texttt{green}) \vee (\texttt{apples} \wedge \texttt{red}) \\
name^{-}(\{1,3,5\}) &= (\texttt{tomatoes} \wedge \texttt{red}) \vee (\texttt{apples} \wedge \texttt{green})
\end{aligned}
$$

5.4 Term-to-Query Articulation

Let S_1 and S_2 be two sources where $S_1 = [(T_1, \preceq_1), I_1]$ and $S_2 = [(T_2, \preceq_2), I_2]$. For brevity, we shall use E_1 to denote I_1^-, E_2 to denote I_2^-, Obj_1 to denote $adom(I_1)$, and Obj_2 to denote $adom(I_2)$.

Clearly, if the active domains of the interpretations of the sources are disjoint, i.e. $Obj_1 \cap Obj_2 = \emptyset$, then we cannot infer any relationship between the terms of the two ontologies.

Let us first consider the sources S_1 and S_2 shown in Figure 5.4 and suppose that we want to articulate the term t of S_1 with the term t' of S_2. The articulation $a_{1,2}$ may contain one of the following relationships:

- $t \succeq t'$ because $E_1(t) \supseteq E_2(t') \cap Obj_1$, i.e. $\{1,2\} \supseteq \{2\}$,
- $t \preceq t'$ because $E_1(t) \cap Obj_2 \subseteq E_2(t')$, i.e. $\{2\} \subseteq \{2,3\}$,
- $t \sim t'$ because $E_1(t) \cap Obj_2 = E_2(t') \cap Obj_1$, i.e. $\{2\} = \{2\}$.

For similar reasons, the articulation $a_{2,1}$ may contain the relationship $t' \succeq t$, $t' \preceq t$, or $t' \sim t$. Our approach is that an articulation should contain relationships that hold in the common domain of the two sources, i.e. in the set $Obj_1 \cap Obj_2$. This means that in our example we will have $a_{1,2} = a_{2,1} = \{t \sim t'\}$.

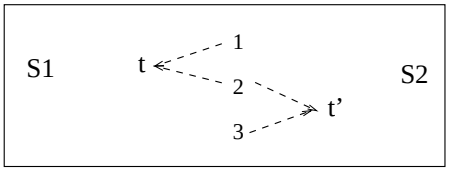


Figure 5.4: Example of two sources with overlapping domains

Consider the case where S_1 wants to articulate a term (or query) a of its ontology with a query of S_2. Figure 5.5 describes the appropriate protocol for articulating the term a. We use $S_1 \stackrel{m}{\rightarrow} S_2$ to denote that S_1 sends the message m to S_2. We will also denote $S_1 \stackrel{m}{\rightarrow} S_2$ by $S_1 \rightarrow S_2 : m$.

Term-to-Query
Input: $a \in Q_{T_1}$
Output: subsumption relationships between a and elements of Q_{T_2}

S_1 $\quad S_1 \rightarrow S_2 : E_1(a)$

S_2 $\quad$ If $E_1(a) \cap Obj_2 \neq \emptyset$ then
$\quad\quad b^- = name_2^-(E_1(a) \cap Obj_2)$
$\quad\quad b^+ = name_2^+(E_1(a) \cap Obj_2)$
$\quad\quad S_2 \rightarrow S_1 : (b^-, E_2(b^-)), (b^+, E_2(b^+))$
$\quad$ else $S_2 \rightarrow S_1 : (\perp, \emptyset), (\perp, \emptyset)$

S_1 $\quad$ If $b^- \neq \perp$ then
$\quad\quad$ If $E_1(a) \supseteq E_2(b^-) \cap Obj_1$ then set $a \succeq b^-$
$\quad\quad$ If $E_1(a) \subseteq E_2(b^-) \cap Obj_1$ then set $a \preceq b^-$

$\quad$ If $b^+ \neq \perp$ then
$\quad\quad$ If $E_1(a) \supseteq E_2(b^+) \cap Obj_1$ then set $a \succeq b^+$
$\quad\quad$ If $E_1(a) \subseteq E_2(b^+) \cap Obj_1$ then set $a \preceq b^+$

Figure 5.5: The protocol for term-to-query articulation

The protocol works as follows:

- At first S_1 sends to S_2 the interpretation of a, i.e. the set of objects $E_1(a)$.
- Upon reception of this message, S_2 performs the intersection of the received set of objects

and the active domain of S_2. If the intersection turns out to be empty, then no articulation is derived, therefore S_2 sends to S_1 the bottom query $\perp$ and the empty set.

Otherwise (if the intersection is not empty), then S_2 computes the *lower* and *upper name* of the received set of objects with respect to S_2. Specifically, S_2 computes the lower and upper name of the set $E_1(a) \cap Obj_2$, because the received set of objects $E_1(a)$ may not be subset of the active domain of S_2. Subsequently, S_2 sends to S_1 these names and their interpretation in E_2.

- In turn, S_1 by comparing the set $E_1(a)$ with the sets $E_2(b^-)$ and $E_2(b^+)$, infers zero, one, or more relationships between a and b^-, b^+.

As an example, consider the sources shown in Figure 5.6. Suppose that S_1 wants to articulate its terms with queries of S_2.

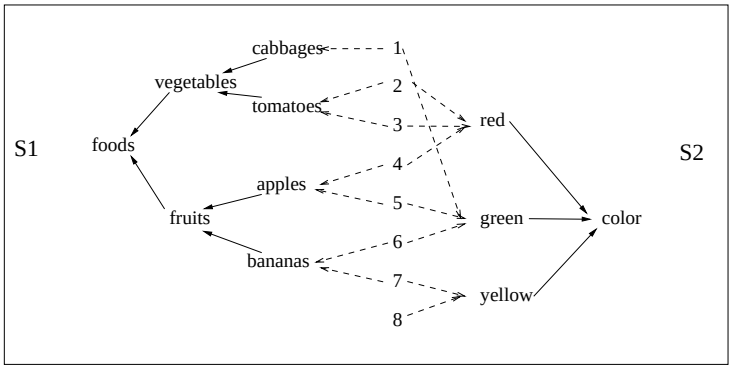


Figure 5.6: An example of two sources S_1 and S_2

The steps for articulating the term `cabbages` follow:

$S_1 \rightarrow S_2$: {1}
$S_2 \rightarrow S_1$: ($\perp, \emptyset$),(`green`, {1,5,6})
S_1 : `cabbages` $\preceq$ `green`

The steps for articulating the term `apples` follow:

$S_1 \rightarrow S_2$: {4, 5}
$S_2 \rightarrow S_1$: ($\perp, \emptyset$), (`red` $\vee$ `green`,{1,2,3,4,5,6})
S_1 : `apples` $\preceq$ `red` $\vee$ `green`

The steps for articulating the term `foods` follow:

$S_1 \rightarrow S_2$: {1,2,3,4,5,6,7}
$S_2 \rightarrow S_1$: (`red` $\vee$ `green`,{1,2,3,4,5,6}), (`red` $\vee$ `green` $\vee$ `yellow`,{1,2,3,4,5,6,7,8})
S_1 : `foods` $\succeq$ `red` $\vee$ `green`,
`foods` $\preceq$ `red` $\vee$ `green` $\vee$ `yellow`

If S_1 runs the protocol for each term of its ontology, it will infer the following relationships:

$$\begin{aligned}
\texttt{cabbages} &\preceq \texttt{green} \\
\texttt{tomatoes} &\preceq \texttt{red} \\
\texttt{apples} &\preceq \texttt{red} \vee \texttt{green} \\
\texttt{bananas} &\preceq \texttt{green} \vee \texttt{yellow} \\
\texttt{vegetables} &\preceq \texttt{green} \vee \texttt{red} \\
\texttt{fruits} &\preceq \texttt{red} \vee \texttt{green} \vee \texttt{yellow} \\
\texttt{foods} &\succeq \texttt{red} \vee \texttt{green} \\
\texttt{foods} &\preceq \texttt{red} \vee \texttt{green} \vee \texttt{yellow}
\end{aligned}$$

If S_2 runs this protocol for each term of its ontology, it will infer the following relationships:

$$\begin{aligned}
\texttt{red} &\succeq \texttt{tomatoes} \\
\texttt{red} &\preceq \texttt{tomatoes} \vee \texttt{apples} \\
\texttt{green} &\succeq \texttt{cabbages} \\
\texttt{green} &\preceq \texttt{cabbages} \vee \texttt{apples} \vee \texttt{bananas} \\
\texttt{color} &\succeq \texttt{cabbages} \vee \texttt{tomatoes} \vee \texttt{apples} \vee \texttt{bananas}
\end{aligned}$$

The protocol can be used not only for articulating simple terms to queries, but also for articulating queries to queries. For example, the steps for articulating the query $\texttt{apples} \vee \texttt{bananas}$ follow:

$S_1 \rightarrow S_2$: {4, 5, 6, 7}
$S_2 \rightarrow S_1$: ($\texttt{red} \vee \texttt{green} \vee \texttt{yellow}$, {1,2,3,4,5,6,7,8})
S_1 : $\texttt{apples} \vee \texttt{bananas} \preceq \texttt{red} \vee \texttt{green} \vee \texttt{yellow}$

The steps for articulating the query $\texttt{fruits} \wedge \neg\, \texttt{bananas}$ follow:

$S_1 \rightarrow S_2$: {4, 5}
$S_2 \rightarrow S_1$: ($\texttt{red} \vee \texttt{green}$, {1,2,3,4,5,6})
S_1 : $\texttt{fruits} \wedge \neg\, \texttt{bananas} \preceq \texttt{red} \vee \texttt{green}$

As another example, consider the sources S_1 and S_2 shown in Figure 5.7, and suppose that S_1 wants to articulate its terms. By running the protocol it will infer the following relationships:

$$\begin{aligned}
\texttt{tomatoForSalad} &\sim \texttt{tomatoes} \wedge \texttt{red} \\
\texttt{favoriteApple} &\sim \texttt{apples} \wedge \texttt{green} \\
\texttt{myFoods} &\preceq (\texttt{tomatoes} \wedge \texttt{red}) \vee (\texttt{apples} \wedge \texttt{green})
\end{aligned}$$

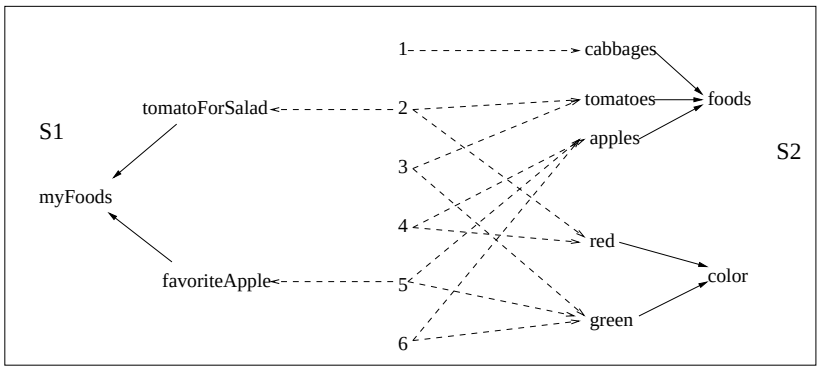


Figure 5.7: An example of two sources S_1 and S_2

5.5 Terms-to-Term Articulation

Suppose that we do not want to articulate terms with queries, but terms with *single terms* only.

For the pusposes of the *term-to-term* articulation, we can define the upper and lower closure of a set K as follows:

$$\begin{aligned} K^+ &= \{t \in T \mid K \subseteq I^-(t)\} \\ K^- &= \{t \in T \mid K \supseteq I^-(t)\} \end{aligned}$$

Note that now $glb(K^+)$ and $lub(K^-)$ do not always exist. For example, consider the source shown in Figure 5.8.(a). Note that $\{1\}^+ = \{t, t'\}$ and that $glb(\{t, t'\})$ does not exist. For the source shown in Figure 5.8.(b) note that $\{1, 2\}^- = \{t, t'\}$ and that $lub(\{t, t'\})$ does not exist.

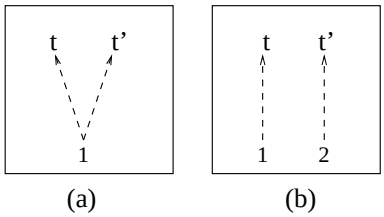


Figure 5.8: An example of two sources

This means that we cannot define the upper and lower name as we did before. However, we can define the upper and lower *names* of a set K as follows:

$$\begin{aligned} names^+(K) &= min(K^+) \\ names^-(K) &= max(K^-) \end{aligned}$$

Consider for example the source shown in Figure 5.9. Here we have:

$$\begin{aligned} \{1,2,3\}^{+} &= \{b,a\} \\ \{1,2,3\}^{-} &= \{c,d,e\} \\ names^{+}(\{1,2,3\}) &= \{b\} \\ names^{-}(\{1,2,3\}) &= \{c,d\} \end{aligned}$$

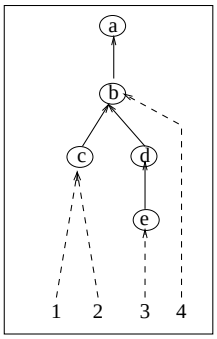


Figure 5.9: An example

A protocol for articulating simple terms with simple terms is quite similar to the protocol for articulating terms with queries that was shown in Figure 5.5. Recall that in that protocol, for articulating a term $a \in T_1$, S_2 sends to S_1 at most two queries, namely the queries b^+ and b^- (and their interpretation). However here, for articulating a term $a \in T_1$, S_2 may send several terms (and their interpetation) to S_1, namely the set of terms $names^+(K)$ and $names^-(K)$.

Figure 5.10 shows the protocol for articulating a term $a \in T_1$ with simple terms from T_2.

Certainly, the relationships obtained by the term-to-term articulation are less expressive than the relationships obtained by the term-to-queries articulation. For instance, suppose that we want to articulate the terms of the source S_1 in each one of the three examples that are shown in Figure 5.11. Table 5.1 shows the articulation $a_{1,2}$ that is derived by the *term-to-term* articulation and the *term-to-queries* articulation in each of these three examples.

5.6 The Role of a Training Set

Consider that we want to articulate the ontologies of two sources S_1 and S_2. We can run the presented protocol on a *training set* in order to : (a) *enhance the accuracy* of the resulting articulation, and/or (b) to *enhance efficiency*.

Term-to-Term
Input: $a \in T_1$
Output: subsumption relationships between a and elements of T_2

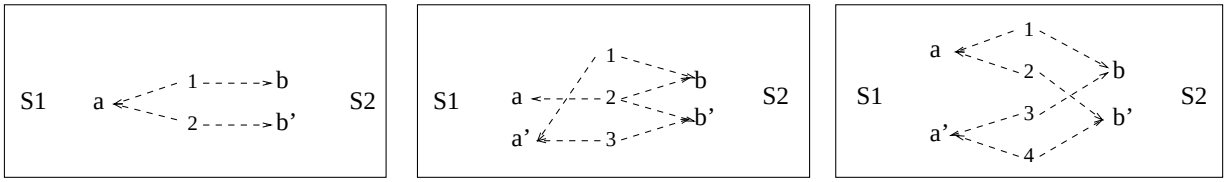
S_1 $S_1 \rightarrow S_2 : E_1(a)$

S_2 If $E_1(a) \cap Obj_2 \neq \emptyset$ then
$blist^- = names_2^-(E_1(a) \cap Obj_2)$
$blist^+ = names_2^+(E_1(a) \cap Obj_2)$
for each $b \in blist^-$
$S_2 \rightarrow S_1 : (b, E_2(b))$
for each $b \in blist^+$
$S_2 \rightarrow S_1 : (b, E_2(b))$
else $S_2 \rightarrow S_1 : (\perp, \emptyset), (\perp, \emptyset)$

S_1 for each $b \in blist^-$
If $b \neq \perp$ then
If $E_1(a) \supseteq E_2(b) \cap Obj_1$ then set $a \succeq b$
If $E_1(a) \subseteq E_2(b) \cap Obj_1$ then set $a \preceq b$

for each $b \in blist^+$
If $b \neq \perp$ then
If $E_1(a) \supseteq E_2(b) \cap Obj_1$ then set $a \succeq b$
If $E_1(a) \subseteq E_2(b) \cap Obj_1$ then set $a \preceq b$

Figure 5.10: The protocol for articulating a term of S_1 in the *term-to-term* articulation



(a) (b) (c)

Figure 5.11: Three examples

Example	$a_{1,2}$	
	term-to-term articulation	*term-to-query* articulation
Figure 5.11.(a)	$a \succeq b$ $a \succeq b'$	$a \sim b \vee b'$
Figure 5.11.(b)	$a \preceq b$ $a \preceq b'$	$a \sim b \wedge b'$ $a' \preceq b \vee b'$
Figure 5.11.(c)		$a \preceq b \vee b'$ $a' \preceq b \vee b'$

Table 5.1: *Term-to-term* vs *term-to-query* articulation

A training set is a set of objects C which is known to both sources, i.e. $C \subseteq Obj_1 \cap Obj_2$. Running the protocol on the training set C means that the sources S_1 and S_2 instead of using $E_1(a)$ and $E_2(b)$, they use $E_1(a) \cap C$ and $E_2(b) \cap C$ respectively.

If the training set C corresponds to a well known, thus well-indexed set of objects then it can improve the quality of the obtained articulations. For example in the case where S_1 and S_2 are bibliographic sources, C can be a set of 100 famous papers in computer science.

A training set can also enhance the efficiency of the protocol, which is very important, especially in the case where the involved sources are distant. In this way, the articulation can be obtained more efficiently since a smaller number of objects go back and forth. For example, consider two portals each offering an ontology-based catalog of Web resources that is of interest to their users. Assume that the portals want to cooperate, that is, they have agreed that if one portal cannot answer a user's query, then it must forward the query to the other portal. For this reason they may run our protocol for articulating their ontologies. The obtained articulations are then stored and used for translating queries across the ontologies in the future. In particular, by running the protocol, S_1 will obtain an articulation $a_{1,2}$, and S_2 will obtain an articulation $a_{2,1}$. However the interpretations of the terms of such catalogs are commonly very big (millions of objects). For this reason running the protocol is time and bandwidth consuming. In order to reduce the cost, the sources can agree to run the protocol on a *training set* C.

5.7 Applications

5.7.1 Ontology Integration/Articulation

Consider a domain expert who wants to integrate two ontologies in order to derive a new one. For deriving the integrated ontology he copies the two ontologies in a new space and then he uses our protocol as follows: he selects a term (or query) from one ontology and the protocol returns a semantically close term (or query) from the other. If he agrees with the resulting relationship he stores it in the integrated ontology.

Let $(T_1, \preceq_1)$ and $(T_2, \preceq_2)$ be the original ontologies. Let $a_{1,2}$ denote the articulation between the elements of T_1 and queries over Q_{T_2}, and let $a_{2,1}$ denote the articulation between the elements of T_2 and queries over Q_{T_1}. The integrated ontology is obtained by taking the union of the original ontologies and the two articulations $a_{1,2}$ and $a_{2,1}$. Note that the resulting ontology is not captured by the definition 2.1 (of Chapter 2) because it may contain subsumption relationships between terms and *queries*. In the case where we do not want such relationships in the integrated ontology, the term-to-term articulation approach that was presented in Section

5.5 can be employed.

Ontology integration however requires having a big number of common objects. If the objects of the domain have a textual content then a machine learning approach (such as the one presented in [34]) can be employed in order to "extend" the set of common objects.

5.7.2 Mediators

Consider a mediator over at least one source, i.e. a mediator $M = (T, \preceq, a_1, ..., a_{k-1})$, where $k \geq 2$, and suppose that the designer of the mediator wants to define an articulation a_k to a new source $S_k = [(T_k, \preceq_k), I_k]$. Specifically, suppose that we want to articulate a term t of the mediator with the ontology of the source S_k.

For doing so, we can use the articulation protocol described in this chapter, which requires the mediator to send to S_k the term t and its interpretation. Since the mediator does not (necessarily) have a stored interpretation of its terminology T, it can *query* the sources $S_1, ...S_{k-1}$ (as described in Chapter 4) in order to compute the interpretation of the term t. Furthermore, recall that a mediator may also have a stored interpretation I of its terminology, which can be also exploited for computing the interpretation of t.

Thus, the articulation protocol can assist the designer of a mediator during the articulation of a new source, as shown in Figure 5.12.

Mediators Revisited

By using the articulation protocol the mediator can infer subsumption relationships between terms of its own ontology and queries of the sources. However, according to Chapter 4 an articulation a_i consists of subsumption relationships between terms only. Therefore, in this section we extend the definition of an articulation to include subsumption relationships between terms and queries as well.

Def 5.4 Let $(T, \preceq)$ be the ontology of a mediator and let $(T_i, \preceq_i)$ be the ontology of source S_i. An *articulation* a_i is a subsumption relation over $T \cup Q_{T_i}$.

This extension is very useful, because now the designer of the mediator can define articulations containing more complex relationships, as in the following examples:

- $\texttt{Electronics}_M \succeq (\texttt{TV}_i \vee \texttt{Mobiles}_i \vee \texttt{Radios}_i)$
- $\texttt{DBArticles}_M \sim (\texttt{Databases}_i \wedge \texttt{Articles}_i)$

In the first example, the users of the mediator can use the term `Electronics` instead of a long disjunction of terms at source S_i (benefit: brevity), while in the second they can use the term

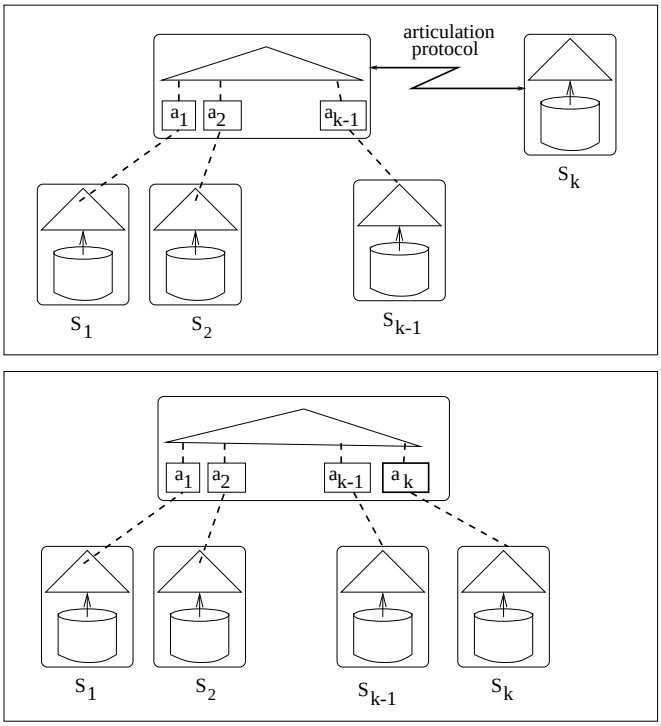


Figure 5.12: A mediator using the articulation protocol

`DBArticles` instead of the conjunction of two terms at source S_i (note, however, that this is useful only if S_i supports multiple classification).

Let us now discuss the consequences of this extension with regard to the functionality of the mediators. For each term $t \in T$ the *tail* and *head* of t with respect to a_i can be defined as follows:

Def 5.5 Given a term $t \in T$ and articulation a_i we define

$$tail_i(t) = \{s \in Q_{T_i} \mid sa_it\} \quad \text{and} \quad head_i(t) = \{u \in Q_{T_i} \mid ta_iu\}$$

Note that now the tail and head of a term are not sets of terms of T_i, but sets of *queries* over T_i.

The *lower* and *upper* approximation of t with respect to a_i are defined as in Chapter 4. The four interpretations and the eight answer models of the mediator are defined in the same way too.

In this framework the concept of compatibility is now redefined as follows:

Def 5.6 A source S_i is *compatible* with the mediator M if for any queries s, t in Q_{T_i}, if sa_it then $s \preceq_i t$.

As we mentioned in Section 4.4, maintaining compatibility is not an easy task. The mediator should (periodically) check the compatibility of its sources, e.g. by submitting to them queries allowing to check whether $t \preceq_i t'$. However, t and t' are now queries, thus the sources should support subsumption checking over queries.

5.7.3 Agent Communication

If we consider each source as an autonomous information agent, then our algorithm can be used for defining a protocol which can be adopted by two agents for the purpose of "learning" the language (terminology) of each other (e.g. see [96], [41] for a review). In this view, we can consider that the set Obj stands for the real world, the set of objects that are indexed under both ontologies stands for the part of the world which is known to both agents, and the ontology of each agent stands for the terminology (language) that the agent uses in order to refer to the world. The agents can run this protocol in order to infer relationships that hold between their terminologies, and they can store these relationships as articulations for using them in subsequent communication sessions.

5.8 Summary and Conclusion

In this chapter we presented a data-driven approach for articulating ontologies. This method can be used for the automatic or semi-automatic construction of an articulation between two or more materialized ontologies. A distinctive feature of this method is that it is *independent* of the nature of the objects, i.e. the objects may be images, audio, video, etc, and that it can be implemented efficiently by a *communication protocol*, thus the involved ontologies (sources) can be *distant*.

Our method can also be employed in order to assist integration/articulation of ontologies which contain attributes and relations. This is because successfully matching the subsumption hierarchies would greatly aid matching the remaining parts of the ontologies.

A different approach to the same problem can be found in [34]. The approach presented there is based on probabilistic metrics and tries to find matchings for all terms of the involved ontologies. This requires computing a similiarity matrix $|T_1| \times |T_2|$, thus this method is appropriate for integrating the entire ontologies. Note that this is useful only if both ontologies concern the same domain. By contrast, our method can be used in order to articulate only the desired parts of an ontology. Our method yields relationships only if they are justified by the instances. Thus our

method can be used in order to articulate ontologies of different domains. Therefore, we believe that our method is more flexible. It can also be implemented efficiently by a communication protocol, thus the involved ontologies may be distant. Moreover our method is more general since it can link queries - not just terms.

Chapter 6

Result Fusion by a Voting Process

Chapter 4 presented a model for building mediators over ontology-based sources. Here we present a model for building mediators over *retrieval sources*. The combination of these two models allows building mediators over hybrid sources.

Building a mediator over retrieval sources is different from building a mediator over ontology-based sources:

- an ontology-based source accepts queries over a controlled vocabulary (its terminology) and returns a set of objects, while
- a retrieval source accepts natural language queries and returns an ordered set of objects.

Thus a mediator over retrieval sources does not have to translate user queries as each such source accepts the same set of queries. When a user submits a query to the mediator, the mediator just forwards the query to each underlying source. However, since retrieval sources return ordered sets of objects, the mediator should return ordered sets of objects too. This requires having a method for *fusing* (merging) the results in order to derive the ordering to be returned to the user.

We propose a technique for fusing (merging) the results of the underlying sources which is based solely on the results actually returned by each source for each query. The final ordering is derived by aggregating the orderings of each source by a *voting process*. In particular, we view the fusion as an *election* where

- the objects correspond to the *candidates*,
- the sources correspond to the *electors*, and
- each ordering corresponds to a *voting ticket*.

In addition, the fused ordering is accompanied by a *level of agreement*. This factor can be

construed by the user as the level of confidence of the answer returned.

A distinctive characteristic of the proposed method is that it does *not* rely on any prior knowledge about the underlying sources. This characteristic makes this technique applicable to evolving environments in which there are several sources whose functionality evolves unpredictably over time, thus it is appropriate for the environment of the Web. Moreover this technique does not introduce time or bandwidth costs, nor does require it of the underlying sources to support any special communication protocol.

The remaining of this chapter is organized as follows: Section 6.1 reviews related work. Section 6.2 formulates the problem. Section 6.3 describes the proposed fusion technique. Section 6.4 introduces a measure of distance over orderings, and Section 6.5 describes the method for deriving the level of agreement. Section 6.6 discusses implementation, Section 6.7 shows how this technique can be applied for building mediators over retrieval, ontology-based and hybrid sources, and finally, Section 6.8 concludes the chapter and identifies issues for further research.

6.1 Related Work

The problem of result fusing (or merging) is being studied mostly in the area of text information retrieval, specifically in the area of meta searching. In general, metasearchers (for example MetaCrawler [115], SavvySearch [60], Profusion [40]) merge results from multiple search sources into a single ranked list using some *results fusing* strategy.

Fusing strategies can be divided into two categories [138]: *integrated methods* and *isolated methods*. Integrated methods require the sources to provide special information for use in fusing, while isolated fusing methods can be applied without any special information from the sources. We are interested in isolated methods because the existing integrated methods present some important drawbacks, apart from having narrow applicability. In order to perform server selection and result merging a mediator usually takes into account retrieval effectiveness measures of the underlying sources, such as *Precision/Recall curves* (see for example the system described in [43]). However, in our opinion, these measures are ill-defined. They are based on the assumption that in a given collection O of documents, and for a given query q, there is a subset R ($R \subseteq O$) of relevant documents, while the rest of the documents ($O \setminus R$) are non-relevant. If the system which is being evaluated processes the information request q and generates a document answer set A, then recall and precision are defined as follows:

$$Recall = \frac{|A \cap R|}{|R|} \qquad\qquad Precision = \frac{|A \cap R|}{|A|}$$

However, the Information Retrieval problem is based on the following assumption:

for a given information need, some documents are just more relevant *than others*

Thus the set R cannot be specified exactly: the specification of R amounts to specifying an appropriate relevance or cardinality threshold. For example, the upper part of Figure 6.1 shows the partition of a document set to those documents that are relevant to q and those that are irrrelevant to q. On the other hand, the lower part of this Figure shows the ordering of the documents with respect to their relevance to q. Document 3 is the most relevant while document 2 is the least relevant. Note that documents 5 and 6 are equally relevant.

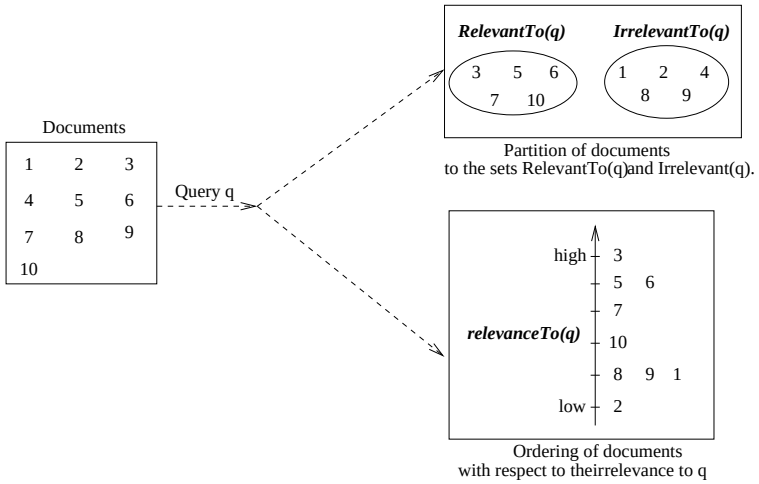


Figure 6.1: Distinction of documents according to relevance

Moreover, there is the problem of subjectiveness: who judges and how whether a document is relevant to a query, or more relevant than another document ? Certainly there is no all-knowing or widely accepted human or source. This means that in a heterogeneous environment like the Web, taking precision and recall measures into account, or comparing the measures of different sources, is an ill-founded approach and may result to low retrieval effectiveness. For instance, the approach for server selection proposed in [43] presumes that the mediator knows the number of documents in each underlying source that is relevant to the interests of the users of the mediator! Some other approaches (e.g. [138], [139]) exploit the results of past queries, or training queries, for estimating the number of relevant documents of each underlying source. However such approaches are rather unrealistic in the context of the Web, given the autonomy and continuous evolution of Web sources.

Certain merging approaches just interleave the returned orderings (i.e. [43]) while others assume that the degrees of relevance returned by each source are comparable, and use them for ordering the results (i.e. [138], [139]). An interesting approach that fits to this category can be

found in [39]. The proposed approach combines fuzzy information from multiple systems which are not necessarily text retrieval systems. It assumes that some systems return sets of objects while others return sorted lists, in particular graded sets (fuzzy sets). The proposed approach could be used for building a meta search engine only if the degrees of relevance returned by the underyling systems were comparable. In [32] two isolated techniques for merging search results are introduced. These techniques require downloading the document contents and they also employ a set of relevance collection statistics. The drawbacks of these techniques are that (a) downloading document contents incurs time and bandwidth costs, and (b) the use of relevance collections statistics presupposes that the underlying sources are known in advance.

However, the availability of multiple sources presents a new opportunity: it allows us to derive *aggregate measures of relevance* by viewing the problem of results fusing as a *group decision problem*. We believe that such an approach fits better to the Web environment and the technique presented in this chapter implements this approach.

6.2 Problem Formulation

We consider a domain, i.e. a collection O of objects, such as documents, Web pages, etc, and a set of sources S over that domain. Specifically, let $O = \{o_1, ..., o_n\}$ and let $S = \{S_1, ..., S_k\}$. We also consider a mediator over the sources of S, that we denote by M or by $[S_1, ..., S_k]$. Upon receiving a query q the mediator forwards q to each source in S. When a source S_i receives a query q, it returns a linear ordering of the set O. We denote this ordering by $(O)_i(q)$, or just by $(O)_i$, if the query q is clear from context. After retrieving the orderings returned by the underlying sources, the mediator M has to "fuse" them in order to derive a single ordering, denoted by $(O)_0$. The problem that we study in this chapter is precisely how to do this fusing.

6.3 Fusion by Voting

Consider a mediator $[S_1, ..., S_k]$ who has forwarded a query q to each of its underlying sources, and let $(O)_1, ..., (O)_k$ be the returned orderings of O.

Def 6.1 Let $(O)_i$ be an ordering of O and let o be an object in O. We denote by $r_i(o)$ the *position*, from the left, of o in $(O)_i$.

For instance, if $(O)_i =< o_1, o_2 >$ then $r_i(o_1) = 1$ and $r_i(o_2) = 2$.

Def 6.2 The *votes of* an object o over a set of sources S, denoted by $V_S(o)$, is defined as follows:

$$V_S(o) = \sum_{S_i \in S} r_i(o)$$

That is, V_S is a function ($V_S : O \rightarrow Int$) which for each object o returns the sum of the positions of o in the orderings returned by the sources in S. Clearly, if S consists of only one source, e.g. the source S_1, then $V_S(o) = V_{\{S_1\}}(o) = r_1(o)$ for each object o. When the set S will be clear from context, we will denote the function V_S by just V.

We view the fusion problem as an *election*, where the objects correspond to the *candidates*, the sources correspond to the *electors*, and each ordering $(O)_i$ corresponds to the *voting ticket* of S_i. More specifically, $r_i(o_j)$ is the vote of S_i for the object o_j. We assume that the "small" votes (numbers) are given to the preferred objects, while the "big" votes are given to the non-preferred. From this perspective, the quantity $V_S(o_j)$ is the sum of the votes that o_j took by the sources in S. This means that the object o which has the smallest $V_S(o)$, is the *winner* of the elections. Thus we can derive the ordering $(O)_0$, by ordering the objects in decreasing order with respect to V_S.

Some examples are given below:

- Example (A)

$$\begin{aligned} (O)_1 &= < o_1, o_2 > \\ (O)_2 &= < o_2, o_1 > \\ V(o_1) &= 1 + 2 = 3 \\ V(o_2) &= 2 + 1 = 3 \end{aligned}$$

  Observe that the objects o_1, o_2 took the same votes. This implies that we cannot derive a single linear ordering of O. Thus we can write: $(O)_0 = \{o_1, o_2\}$. We will return to the equally-voted objects later in this section.

- Example (B)

$$\begin{aligned} (O)_1 &= < o_1, o_2, o_3 > \\ (O)_2 &= < o_1, o_2, o_3 > \\ (O)_3 &= < o_1, o_2, o_3 > \\ V(o_1) &= 1 + 1 + 1 = 3 \\ V(o_2) &= 2 + 2 + 2 = 6 \\ V(o_3) &= 3 + 3 + 3 = 9 \\ (O)_0 &= < o_1, o_2, o_3 > \end{aligned}$$

  Here each source returned the same ordering of O. In such cases we say that we have

homophony. Clearly, if we have homophony we can always derive a single linear ordering of O.

- Example (C)

$$\begin{aligned}
(O)_1 &= < o_1, o_2, o_3 > \\
(O)_2 &= < o_1, o_2, o_3 > \\
(O)_3 &= < o_1, o_3, o_2 > \\
V(o_1) &= 1 + 1 + 1 = 3 \\
V(o_2) &= 2 + 2 + 3 = 7 \\
V(o_3) &= 3 + 3 + 2 = 8 \\
(O)_0 &= < o_1, o_2, o_3 >
\end{aligned}$$

Here although we have not homophony, we are still able to derive a single linear ordering of O.

- Example (D)

$$\begin{aligned}
(O)_1 &= < o_1, o_2, o_3 > \\
(O)_2 &= < o_1, o_3, o_2 > \\
(O)_3 &= < o_3, o_1, o_2 > \\
V(o_1) &= 1 + 1 + 2 = 4 \\
V(o_2) &= 2 + 3 + 3 = 9 \\
V(o_3) &= 3 + 2 + 1 = 6 \\
(O)_0 &= < o_1, o_3, o_2 >
\end{aligned}$$

- Example (E)

$$\begin{aligned}
(O)_1 &= < o_1, o_2, o_3 > \\
(O)_2 &= < o_1, o_3, o_2 > \\
(O)_3 &= < o_2, o_1, o_3 > \\
(O)_4 &= < o_2, o_3, o_1 > \\
(O)_5 &= < o_3, o_1, o_2 > \\
(O)_6 &= < o_3, o_2, o_1 > \\
V(o_1) &= 1 + 1 + 2 + 3 + 2 + 3 = 12
\end{aligned}$$

$$
\begin{aligned}
V(o_2) &= 2+3+1+1+3+2 = 12 \\
V(o_3) &= 3+2+3+2+1+1 = 12 \\
(O)_0 &= \{o_1, o_2, o_3\}
\end{aligned}
$$

Here the mediator received the set of all possible orderings of O.

In the previous discussion, we assumed that each source returns a linear ordering of O. However there are cases where the answer of a source is not a linear ordering. For instance, if a source returns objects accompanied by degrees of relevance, then there may be two or more objects with the same degree of relevance. For this purpose hereafter we consider that an answer is a linear ordering of a *set* of objects. For example, consider a source that returns objects accompanied by their degree of relevance and suppose that this source returns the following answer:

$$\{(o_1, 0.8), (o_2, 0.8), (o_3, 0.7), (o_4, 0.65), (o_5, 0.65), (o_6, 5)\}$$

We view this answer as: $< \{o_1, o_2\}, \{o_3\}, \{o_4, o_5\}, \{o_6\} >$ or $< \{o_1, o_2\}, o_3, \{o_4, o_5\}, o_6 >$ for notational simplicity.

This view is also useful when fusing. As demonstrated in examples (A) and (E), the summation of votes may result in ties, i.e. equally voted objects. In such cases we cannot derive a single linear ordering. However with the revised definition of an answer, we can assume that all equally voted objects reside on the *same* position in the final ordering. In example (E) this means that $r_0(o_1) = r_0(o_2) = r_0(o_3) = 1$, and we can write $(O)_0 =< \{o_1, o_2, o_3\} >$. As another example consider a case where: $V(o_1) = 2$, $V(o_2) = 5$, $V(o_3) = 5$, and $V(o_4) = 7$, The fused ordering is $(O)_0 =< o_1, \{o_2, o_3\}, o_4 >$.

Hereafter we will use the *ordering* to denote a linear ordering of a set of objects.

Some examples where sources do not return linear orderings of objects follow:

- Example (F)

$$
\begin{aligned}
(O)_1 &= < \{o_1, o_2\}, o_3 > \\
(O)_2 &= < o_3, \{o_1, o_2\} > \\
V(o_1) &= 1+2 = 3 \\
V(o_2) &= 1+2 = 3 \\
V(o_3) &= 2+1 = 3 \\
(O)_0 &= < \{o_1, o_2, o_3\} >
\end{aligned}
$$

- Example (G)

$$\begin{aligned}(O)_1 &= <\{o_1, o_2\}, o_3> \\ (O)_2 &= <o_2, \{o_1, o_3\}> \\ V(o_1) &= 1+2=3 \\ V(o_2) &= 1+1=2 \\ V(o_3) &= 2+2=4 \\ (O)_0 &= <o_2, o_1, o_3>\end{aligned}$$

6.3.1 When Sources Return Ordered *Subsets* of O

In the context of the Web, the set O corresponds to the set of all pointers to Web pages, i.e. all URLs. However, the search engines of the Web do not return an ordering of the entire set O, but only an ordered *subset* of O. Moreover the meta search engines retrieve only a subset of the results of each underlying system, say the first 100 documents because retrieving the entire set of results, might be time prohibitive, since answers usually consist of thousands of pointers.

Here we modify the formulation of our problem to deal with the more general case which also fits to the characteristics of the search systems of the Web. In particular, we assume that each source returns an ordered *subset* of O. We will denote the ordering returned by a system S_i, by $(O_i)_i$ where $O_i \subseteq O$.

One can easily see that in this setting our method for deriving $(O)_0$ is not appropriate. For example suppose that there are 10 sources, i.e. $k = 10$, and consider an object o which belongs only to the set O_1 and assume that $r_1(o) = 10$. Now consider another object o' so that $r_i(o') = 2$ for each $i = 1..k$. This means that $V(o) = 10$ and $V(o') = 20$, thus $V(o) < V(o')$ and o will preceed o' in the final ordering, although it should be the opposite.

In order to overcome this problem it suffices to redefine the function r_i, for $i = 1, ..k$. First, let F be the maximum number of objects that were retrieved by any one source, i.e.:

$$F = max\{|O_1|, ..., |O_k|\}$$

Observe that if $o_j \in O_i$ then $r_i(o_j) \leq F$. Now, we redefine each function r_i as follows:

$$r_i(o_j) = \begin{cases} \text{position of } o_j \text{ in } O_i & \text{if } o_j \in O_i \\ F+1 & \text{otherwise} \end{cases} \tag{6.1}$$

that is, if $o \in O \setminus O_i$ then $r_i(o) = F + 1$. Hereafter we will write r_i^F instead of r_i in order to avoid ambiguities. The summation of votes is done as before. The fused ordering $(O_0)_0$, where $O_0 = \bigcup_{i=1..k} O_i$, is again obtained by ordering the elements of O_0 in ascending order with respect to their votes. Some examples follow:

- Example (I)

$$\begin{aligned}
(O_1)_1 &= < o_1, o_2 > \\
(O_2)_2 &= < o_3, o_4 > \\
V(o_1) &= 1 + 3 = 4 \\
V(o_2) &= 2 + 3 = 5 \\
V(o_3) &= 3 + 1 = 4 \\
V(o_4) &= 3 + 2 = 5 \\
O_0 &= \{o_1, o_2, o_3, o_4\} \\
(O_0)_0 &= < \{o_1, o_3\}, \{o_2, o_4\} >
\end{aligned}$$

- Example (J)

$$\begin{aligned}
(O_1)_1 &= < o_1, o_2 > \\
(O_2)_2 &= < o_2, o_3 > \\
(O_3)_3 &= < o_4, o_3 > \\
V(o_1) &= 1 + 3 + 3 = 7 \\
V(o_2) &= 2 + 1 + 3 = 6 \\
V(o_3) &= 3 + 2 + 2 = 7 \\
V(o_4) &= 3 + 3 + 1 = 7 \\
O_0 &= \{o_1, o_2, o_3, o_4\} \\
(O_0)_0 &= < o_2, \{o_1, o_3, o_4\} >
\end{aligned}$$

- Example (K)

$$\begin{aligned}
(O_1)_1 &= < o_1, o_2, o_3 > \\
(O_2)_2 &= < o_5, o_4, o_3 >
\end{aligned}$$

$$
\begin{aligned}
(O_3)_3 &= < o_5, o_4, o_6 > \\
V(o_1) &= 1+4+4=9 \\
V(o_2) &= 2+4+4=10 \\
V(o_3) &= 3+3+4=10 \\
V(o_4) &= 4+2+2=8 \\
V(o_5) &= 4+1+1=6 \\
V(o_6) &= 4+4+3=11 \\
O_0 &= \{o_1, o_2, o_3, o_4, o_5, o_6\} \\
(O_0)_0 &= < o_5, o_4, o_1, \{o_2, o_3\}, o_6 >
\end{aligned}
$$

6.4 The Distance Between Two Orderings

In order to define the level of agreement of the fused ordering (in Section 6.5), we first define the "distance" between two orderings.

Def 6.3 Let $(O)_a, (O)_b$ be two orderings of O. The *distance* between $(O)_a$ and $(O)_b$, denoted as $dist((O)_a, (O)_b)$, is defined as:

$$dist((O)_a, (O)_b) = \sum_{o \in O} |r_a(o) - r_b(o)| \tag{6.2}$$

$\diamond$

Table 6.1 shows the distances between the orderings of the examples of section 6.3

An important question is whether the function *dist* is a *metric function* (for an introduction to metric spaces see [49]). Recall that given a non-empty set X, a distance function d on X, is called a *metric* for X, if it is a function which assigns to each pair of points a real number ($d : X \times X \rightarrow R$), and satisfies the following properties for all $x, y, z \in X$:

i $d(x, y) \geq 0$

ii $d(x, y) = 0$ if and only if $x = y$

iii $d(x, y) = d(y, x)$

iv $d(x, y) \leq d(x, z) + d(y, z)$, (triangle inequality)

(A)	$dist((O)_1, (O)_2) = 1 + 1 = 2$
(B)	$dist((O)_1, (O)_2) = 0$ $dist((O)_1, (O)_3) = 0$ $dist((O)_2, (O)_3) = 0$
(C)	$dist((O)_1, (O)_2) = 0$ $dist((O)_1, (O)_3) = 1 + 1 = 2$ $dist((O)_2, (O)_3) = 1 + 1 = 2$
(D)	$dist((O)_1, (O)_2) = 1 + 1 = 2$ $dist((O)_1, (O)_3) = 1 + 1 + 2 = 4$ $dist((O)_2, (O)_3) = 1 + 1 = 2$
(E)	$dist((O)_1, (O)_2) = 1 + 1 = 2$ $dist((O)_1, (O)_3) = 1 + 1 = 2$ $dist((O)_1, (O)_4) = 2 + 1 + 1 = 4$ $dist((O)_1, (O)_5) = 1 + 1 + 2 = 4$ $dist((O)_1, (O)_6) = 2 + 2 = 4$ $dist((O)_2, (O)_3) = 1 + 1 + 2 = 4$ $dist((O)_2, (O)_4) = 2 + 2 = 4$ $dist((O)_2, (O)_5) = 1 + 1 = 2$ $dist((O)_2, (O)_6) = 2 + 1 + 1 = 4$ $dist((O)_3, (O)_4) = 1 + 1 = 2$ $dist((O)_3, (O)_5) = 2 + 2 = 4$ $dist((O)_3, (O)_6) = 1 + 1 + 2 = 4$ $dist((O)_4, (O)_5) = 2 + 1 + 1 = 4$ $dist((O)_4, (O)_6) = 1 + 1 = 2$ $dist((O)_5, (O)_6) = 1 + 1 = 2$

Table 6.1: Examples of distances between orderings

In our case, X is the set of all linear orderings of the set O. Below we prove that the function *dist* is indeed a metric for X.

Prop. 6.1 The function *dist* as defined in Def. 6.3, is a metric function.
Proof:

Clearly, the function *dist* satisfies the properties [i], [ii] and [iii]. Below we prove that property [iv] is also satisfied by *dist*.

Let A, B, C be orderings of O. We have

$$\begin{aligned} dist(A,C) &= \sum_{o \in O} |r_A(o) - r_C(o)| \\ dist(A,B) &= \sum_{o \in O} |r_A(o) - r_B(o)| \\ dist(B,C) &= \sum_{o \in O} |r_B(o) - r_C(o)| \end{aligned}$$

Since all $r_A(o_i), r_B(o_i), r_C(o_i)$ are integers (actually positive integers), for any $o \in O$ it holds:

$$|r_A(o) - r_C(o)| \ \leq \ |r_A(o) - r_B(o)| \ + \ |r_B(o) - r_C(o)|$$

since the function $|x - y|$ is a metric for the set of integers. This implies that $dist(A,C) \leq dist(A,B) + dist(B,C)$.

We conclude that the function *dist* is a metric for the set of all orderings of O.

$\diamond$

Here an interesting question arises: Let P denote the set of all orderings of the set O. Each ordering can be viewed in the finite metric space $(P, dist)$.

Let p be a point in P and K a subset of P. We can define the distance of p from K, denoted by $d(p,K)$, as follows:

$$d(p,K) = \frac{\sum_{u \in K} dist(p,u)}{|K|}$$

Now, consider a mediator which receives a set of answers K, where is clearly a subset of P. One might argue that the mediator should return an ordering which corresponds to a point $p_m \in P$ that satisfies the following condition:

$$d(p_m, K) \leq d(p,K) \text{ for each } \ p \in P \tag{6.3}$$

This is shown graphically in Figure 6.2. As the set P is finite, the point(s) which satisfy (6.3) can be found as follows: at first we compute $d(p,K)$ for each $p \in P$ and then we select the point(s) with the smallest distance from K. This requires computing the set of all orderings of

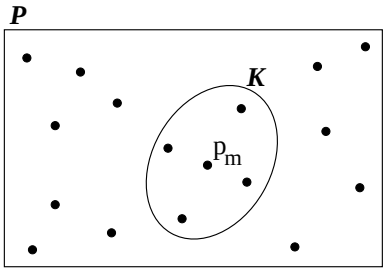


Figure 6.2: The metric space $(P, dist)$.

O. Thus the complexity of finding p_m in this way is exponential with respect to the size of O. Hence, this approach is not appropriate if O is a large set of objects.

On the other hand, the democratic fusion technique does not require computing the set of all orderings of O and the complexity of this technique is linear with respect to the size of O. Now, let p_0 be the point that corresponds to the ordering $(O)_0$ which is derived by the democratic fusion technique. The question is whether the point p_0 satisfies condition (6.3). Unfortunately, we have not yet managed to prove or disprove this, hence this is an issue for further research.

6.4.1 The Distance Between two Ordered Subsets

Consider two sets O_a, O_b such that $O_a \subseteq O$ and $O_b \subseteq O$, and two orderings of these sets denoted by $(O_a)_a$ and $(O_b)_b$. The distance between these orderings, is defined by the following definition:

Def 6.4 Let $(O_a)_a, (O_b)_b$ be two ordered subsets of O. The *distance* between $(O_a)_a$ and $(O_b)_b$, denoted by $dist((O_a)_a, (O_b)_b)$, is defined as:

$$dist((O_a)_a, (O_b)_b) = \sum_{o \in O_a \cup O_b} |r_a^F(o) - r_b^F(o)|$$

where $F = max\{|O_a|, |O_b|\}$.
$\diamond$

The functions r_a^F, r_b^F are as defined in equation (6.1). We will also denote $dist((O_a)_a, (O_b)_b)$ by $dist_F((O_a)_a, (O_b)_b)$ to avoid ambiguities concerning the value of the constant F.

6.5 The Level of Agreement of the Fused Ordering

Apart from deriving the fused ordering, we would also like to derive a factor expressing the *level of agreement* (or confidence level) of the fused ordering. This factor could be given to the user in

order to allow him to distinguish final orderings which exhibit large degree of agreement, from those with small degree of agreement.

Having defined the distance between two orderings of O, we can now compute the distance between the ordering returned by each source S_i, and the final (fused) ordering, $(O)_0$, that is $dist((O)_0, (O)_i)$. These distances for the examples of section 6.3 are shown in Table 6.2.

(A)	$dist((O)_0, (O)_1) = 0 + 1 = 1$ $dist((O)_0, (O)_2) = 1 + 0 = 1$
(B)	$dist((O)_0, (O)_1) = 0$ $dist((O)_0, (O)_2) = 0$ $dist((O)_0, (O)_3) = 0$
(C)	$dist((O)_0, (O)_1) = 0$ $dist((O)_0, (O)_2) = 0$ $dist((O)_0, (O)_3) = 0 + 1 + 1 = 2$
(D)	$dist((O)_0, (O)_1) = 0 + 1 + 1 = 2$ $dist((O)_0, (O)_2) = 0$ $dist((O)_0, (O)_3) = 1 + 0 + 1 = 2$
(E)	$dist((O)_0, (O)_1) = 0 + 1 + 2 = 3$ $dist((O)_0, (O)_2) = 0 + 2 + 1 = 3$ $dist((O)_0, (O)_3) = 1 + 0 + 2 = 3$ $dist((O)_0, (O)_4) = 2 + 0 + 1 = 3$ $dist((O)_0, (O)_5) = 1 + 2 + 0 = 3$ $dist((O)_0, (O)_6) = 2 + 1 + 0 = 3$
(F)	$dist((O)_0, (O)_1) = 0 + 0 + 1 = 1$ $dist((O)_0, (O)_2) = 1 + 1 + 0 = 2$
(G)	$dist((O)_0, (O)_1) = 0 + 1 + 1 = 2$ $dist((O)_0, (O)_2) = 0 + 0 + 1 = 1$
(I)	$dist_4((O_0)_0, (O_1)_1) = 0 + 0 + 4 + 3 = 7$ $dist_4((O_0)_0, (O_2)_2) = 4 + 3 + 0 + 0 = 7$
(J)	$dist_4((O_0)_0, (O_1)_1) = 1 + 1 + 3 + 3 = 8$ $dist_4((O_0)_0, (O_2)_2) = 3 + 0 + 0 + 3 = 6$ $dist_4((O_0)_0, (O_3)_3) = 3 + 4 + 0 + 1 = 8$
(K)	$dist_6((O_0)_0, (O_1)_1) = 2 + 2 + 1 + 5 + 6 + 2 = 18$ $dist_6((O_0)_0, (O_2)_2) = 4 + 3 + 1 + 0 + 6 + 2 = 16$ $dist_6((O_0)_0, (O_3)_3) = 4 + 3 + 3 + 0 + 0 + 2 = 12$

Table 6.2: The distances from the final ordering

We can now define the *distance* of the fused ordering $(O)_0$ from the the set of orderings $\{(O)_i \mid i = 1..k\}$.

Def 6.5 The *distance* of the ordering $(O)_0$ from the set of orderings $\{(O)_i \mid i = 1..k\}$, denoted by $Dem((O)_0, \{(O)_1, ..., (O)_k\})$ is given by:

$$Dem((O)_0, \{(O)_1, ..., O_k\}) = \frac{\sum_{i=1}^{k} dist((O)_0, (O)_i)}{k} \tag{6.4}$$

$\diamond$

Thus it is the average distance between the final ordering $(O)_0$ and each one of the underlying orderings. Table 6.3 shows the distances of the fused orderings of our examples.

(A)	$Dem((O)_0, \{(O)_1, (O)_2\}) = 2/2$
(B)	$Dem((O)_0, \{(O)_1, (O)_2, (O)_3\}) = 0/3$
(C)	$Dem((O)_0, \{(O)_1, (O)_2, (O)_3\}) = 2/3$
(D)	$Dem((O)_0, \{(O)_1, (O)_2, (O)_3\}) = 4/3$
(E)	$Dem((O)_0, \{(O)_1, ..., (O)_6\}) = 18/6$
(F)	$Dem((O)_0, \{(O)_1, (O)_2\}) = 3/2$
(G)	$Dem((O)_0, \{(O)_1, (O)_2\}) = 3/2$
(I)	$Dem((O_0)_0, \{(O_1)_1, (O_2)_2\}) = 14/2$
(J)	$Dem((O_0)_0, \{(O_1)_1, (O_2)_2, (O_3)_3\}) = 22/3$
(K)	$Dem((O_0)_0, \{(O_1)_1, (O_2)_2, (O_3)_3\}) = 46/3$

Table 6.3: The distances of the fused orderings

Remark: Notice that although the distance of the final orderings of the Examples (F) and (G) are equal (=3/2), the final ordering in (F) is a set, while the final ordering in (G) is a linearly ordered set.

However recall that our objective is to derive a factor standing for the *level of confidence* (or agreement) of the final ordering. This factor can be given to the user in order to distinguish the final orderings which exhibit a large degree of agreement from those with a small degree of agreement. Clearly if the distance is large then the confidence factor should be small. Thus we can derive the confidence factor by appropriately transforming the distance. Let us denote this factor by $CF((O)_0, \{(O)_1, ..., O_k\})$. For notational simplicity we shall use Dem for denoting the distance and CF for denoting the confidence factor of the final ordering. Two transformations for deriving CF from Dem are presented below:

One idea is to employ a *linear transformation* of the form

$$CF = V - Dem \tag{6.5}$$

where V is a fixed positive value. Note that V should be big enough if we want CF to be always positive. Thus we can set V equal to the maximum possible distance. Note that in case of

absolute agreement, i.e. homophony, we get $CF = V$, while in case of absolute disagreement, we get $CF = 0$. If we want CF to range in the interval [0,1], which would be more clear, we can derive CF by the following formula:

$$CF = \frac{V - Dem}{V} \tag{6.6}$$

Here, in case of absolute agreement we get $CF = 1$, while in case of absolute disagreement we get $CF = 0$.

Another idea, is to employ an *inversion transformation* of the form

$$CF = V^{-Dem} \tag{6.7}$$

for some fixed value $V > 1$ such as $V = 2$ or $V = e$. This transformation provides a measure with a sharp peak at $Dem = 0$ gradually sloping away towards 0 as Dem becomes larger. For instance consider the Example (E) where we have $Dem = 3$, and assume that $V = 2$. In this case we have $CF = 2^{-3}$. Also note that formula (6.7) relieves us from having to compute the maximum possible distance, which depends on the number of objects and the number of sources.

6.6 Implementation

For implementing the democratic data fusion, the mediator needs a table of the form:

OBJECTS(obj:ID, V:Int)

In the case of the Web where O is the set of all Web pages, the identity of a page is its URL. Initially, this table is empty, unless the domain O is fixed and a priori known to the mediator. In this case the table may by construction contain a row for each object of O and the cells of the column V will have the value 0. Upon reception of an ordering $(O)_i$ the mediator executes the following statement:

For each o_j in $(O)_i$ do
$o_j.V := o_j.V + r_i(o_j)$

6.7 Applications

6.7.1 Mediators over Retrieval Sources

If the objects of the domain are documents and the sources are text retrieval systems ([113], [70],[10]), then our technique can be employed for building meta-retrieval systems, or meta-search

engines (see [10] for a brief introduction). Thus we can have meta-retrieval systems which are independent from the indexing and the matching methods employed by the underlying sources, i.e. we can have mediators over information retrieval systems that employ the boolean model, the vector space model [114], the probabilistic model [108], the inference network model [125], the belief network model [107], or any other information retrieval model. In such a scenario, the only constraint that we impose is that each source accepts the same form of queries, be they bags of words, natural language queries, boolean expressions of terms, etc. This is true in the environment of the Web since most of the search engines of the Web accept natural language queries.

6.7.2 Mediators over Ontology-based Sources

Recall that each ontology-based source returns a subset of Obj. Each such set can be viewed as an ordered set where all of its elements reside on the first position, i.e. $\mathrm{V}(o) = 1$ for each object o in the answer. Consequently, a mediator can use the fusion technique for ordering the objects returned by ontology-based sources too. The mediator can order the objects of the answer in decreasing order with respect to their votes. In this way the most *popular*, or widely-known, objects appear first. Note that the maximum number of votes of an object in an answer equals the number of sources, while the minimum number of votes of an object equals 1.

6.7.3 Mediators over Hybrid Sources

Suppose that we want to build a mediator over a set of hybrid sources. The user can use the ontology of the mediator in order to browse or query those parts of the sources that are of interest to him. Moreover he is able to query the databases of these sources using natural language queries. This implies that the mediator will send two kinds of queries to the sources: (a) queries which are evaluated based on the indexing of the objects with respect to the ontology of the source, and (b) queries which are evaluated based on the contents of the objects (pages). Figure 6.3 describes graphically this architecture.

Note that this application scenario exploits the translation facilities of the mediators presented in Chapter 4 and the result fusing method described in this chapter.

6.8 Summary and Conclusion

In this chapter we presented a result fusing (merging) technique appropriate for environments which comprise heterogeneous and evolving sources. According to this technique, the fused

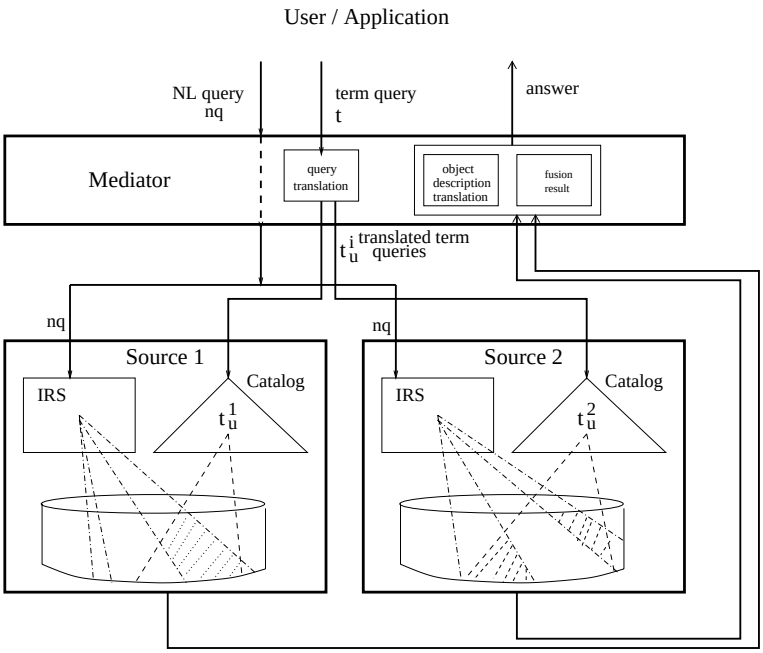


Figure 6.3: Building mediators over hybrid sources

ordering is derived by aggregating the orderings of the underlying sources using a mathematically sound method which is based solely on the actual results returned by each source. The technique is also enriched by a method for deriving the confidence factor of the fused ordering. Given this factor, the user can draw conclusions about the degree of agreement of the results that were returned by the underlying sources as a response to his query. A high factor may drive the user to read only the very first objects of the fused ordering (since they probably are the most relevant to his query), while a low factor may drive him to read more objects.

Chapter 7

Conclusion and Future Research

The main contribution of this dissertation consists of the following:

- A novel scheme for indexing and retrieving objects according to multiple aspects or *facets*. The proposed scheme is a faceted scheme enriched with a method for specifying the combinations of terms that are valid. We give a model-theoretic interpretation to this model and we provide mechanisms for *inferring the valid combinations* of terms. This inference service can be exploited for preventing errors during the indexing process, which is very important especially in the case where the indexing is done collaboratively by many users, and for deriving "complete" navigation trees suitable for browsing through the Web. The proposed scheme has several advantages by comparison to the hierarchical classification schemes that are currently employed by Web catalogs, namely, *conceptual clarity* (it is easier to understand), *compactness* (it takes less space), *scalability* (the update operations can be formulated easier and be performed more efficiently).

- A flexible and efficient model for building *mediators* over ontology-based information sources. The proposed mediators support several modes of *query translation* and evaluation which can accommodate various application needs and *levels of answer quality*. The proposed model can be used for providing users with customized views of Web catalogs. It can also complement the techniques for building mediators over relational sources so as to support approximate translation of partially ordered values.

- A data-driven method for *articulating* ontologies. This method can be used for the automatic or semi-automatic construction of an articulation between two or more materialized ontologies. A distinctive feature of this method is that it is independent of the nature of the objects, i.e. the objects may be images, audio, video, etc, and that it can be implemented efficiently by a communication protocol, thus the involved ontologies (sources) can

be distant.

- A novel method for *fusing* the results of sources that return ordered sets of objects. The proposed technique is based solely on the actual results which are returned by each source for each query. The final (fused) ordering of objects is derived by aggregating the orderings of each source in a *democratic* manner. In addition, the fused ordering is accompanied by a *level of agreement* (alternatively construed as the level of confidence). The proposed method does not require any prior knowledge about the underlying sources, therefore it is appropriate for environments where the underlying systems are heterogeneous and autonomous, thus our method is appropriate for the Web.

These results allow creating a complex network of sources. The network can have primary sources and secondary sources. A primary source can be ontology-based, retrieval or hybrid. Information integration is obtained through secondary sources, i.e. mediators, that can provide uniform and integrated access to multiple primary or secondary sources.

The mediators over ontology-based sources are based on articulations and offer the translation facilities needed in order to cross different ontologies. For example, Figure 7.1 shows a network consisting of four primary ontology-based sources and three mediators. An arrow from a source S to a mediator M means that M is a mediator over the source S, and circles denote articulations. Note that mediator $M1$ can be also considered as a primary source, because it has a stored interpretation.

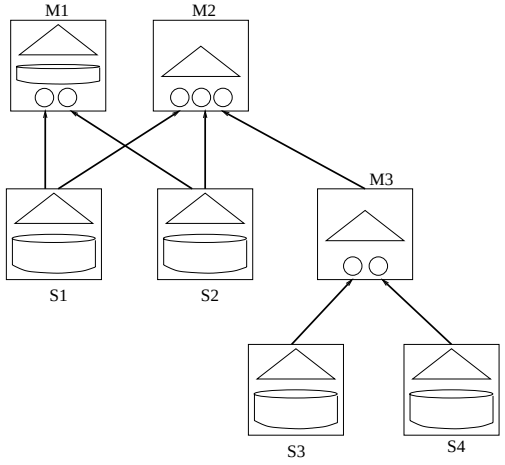


Figure 7.1: A network consisting of primary and secondary ontology-based sources

An advantage of our approach is that one can use it to create a complex network of sources in a natural and straightforward manner. Indeed,

- in order to add a mediator to such a network one has to (a) select the sources to be mediated, (b) design the mediator ontology $(T, \preceq)$ based on the ontologies $(T_i, \preceq_i)$ of the selected sources, and (c) design the articulations a_i based on the known/observed relations between terms of the mediator and terms of the selected sources;
- in order to remove a mediator from the network one just has to disconnect the mediator from the network;
- moreover, in order to add a source to the network all one has to do is (a) select a mediator in the network and (b) design an articulation between the source and that mediator;
- finally, to remove a source one simply has to remove the corresponding articulation from the mediator and disconnect the source.

A significant consequence of this approach is that network evolution can be *incremental*. Indeed, new relationships between terms of the mediator and terms of the sources can be added with minimum effort as soon as they are observed, and relationships that are seen to be no more valid can be removed just as easily: simply add/remove the relationships at the appropriate articulation in the mediator database storing the articulation. Also note that our approach allows having *mutually articulated* sources as shown in Figure 7.2. In this case we cannot longer distinguish sources into primary and secondary.

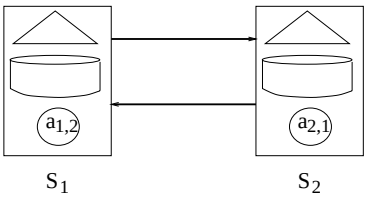


Figure 7.2: Mutually articulated sources

The articulations can be defined manually and/or by the data-driven method that we proposed. The advantages of the extended faceted ontologies that we introduced can certainly ease the task of ontology articulation.

In addition, we can have mediators over retrieval sources which provide aggregate measures of relevance. Figure 7.3 shows a mediator over three primary sources, one ontology-based, one retrieval and one hybrid. Here M is a hybrid mediator. Note that since $S2$ is a retrieval source, M does not need an articulation to $S2$.

Figure 7.4 shows how users can employ mediators for defining personal views over the catalogs of the Web. Note that user $U3$ has a mediator which also has a stored interpretation. This

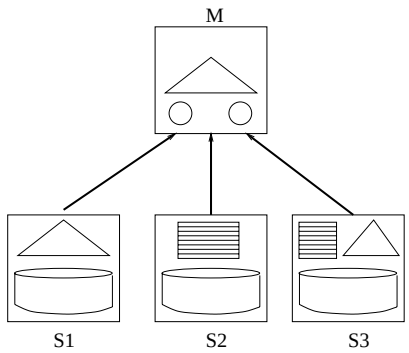


Figure 7.3: A mediator over an ontology-based, a retrieval and a hybrid source

resembles the bookmarks facility of Web browsers, however the addition of articulations now enables the user to browse and query remote catalogs.

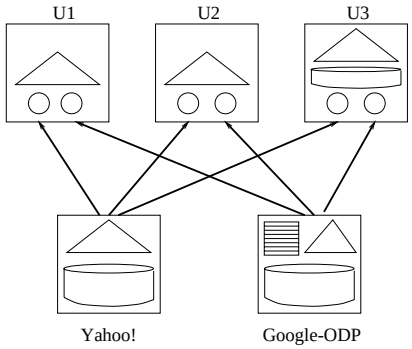


Figure 7.4: Using mediators for personalizing the catalogs of the Web

These results allow creating a complex information network where different users can use their own terminologies in order to index, access and query the objects of a domain.

7.1 Further Research

There are several issues for further research. Below we list some of them organized by topic.

- *Extended Faceted Ontologies*

  An issue for further research is to study how the updates on a faceted ontology $\mathcal{F}$ of a $PEFO < \mathcal{F}, P >$ (or $NEFO < \mathcal{F}, N >$) should affect/update the descriptions which are stored in P (or N).

  Another issue is to investigate the possibility of having both "positive" and "negative" descriptions, i.e. an extended faceted ontology of the form $< \mathcal{F}, P, N >$ equipped with

rules for resolving possible conflicts.

A further issue is to study descriptions which are ordered sets of terms.

- *Democratic Result Fusion*

  Suppose that we have democratic mediators over other democratic mediators. The following questions arise: (a) how the confidence factor of the answers returned by the underlying mediators should affect the derivation of the fused ordering, and (b) how we should compute the confidence factor of the fused ordering.

Broader issues that we think are worth further research include:

- *Algebra over ontologies and ontology-based sources*

  It would be very useful to have an *algebra* with operators for *articulating, integrating, personalizing* and *querying* ontologies and ontology-based sources. This algebra should formalize and extend the ideas of [141], so as to provide a well-founded method for (a) composing and personalizing ontologies (thus reducing the cost of developing them from scratch), and (b) integrating sources and personalizing/adapting them. For doing this, one idea is to investigate whether the results of *formal concept analysis* [45], and Galois connections, can be exploited.

- *Update operations in a network of sources*

  An issue for further research is to study the updates in a network of articulated sources. Since many sources will be actually views of other sources (virtual or materialized), the issue of *view updating* arises. It would be interesting to investigate whether we can extend the work of [11] in order to apply it in this context. We also plan to design and study algorithms for *propagating* the updates so that the entire network be up to date and stable.

Bibliography

[1] " Aristotle's Theory of Substance ". (http://socrates.berkeley.edu/ fl3min4/25A/aristotlesontology.html).

[2] "INSPEC Thesaurus". (http://www.iee.org/publish/Support/inspec/Document/Class/).

[3] "Knowledge Interchange Format". Draft proposed American National Standard (dpANS) NCITS.T2/98-004, (http://logic.stanford.edu/kif/dpans.html).

[4] "Library of Congress Subject Headings (LCSH)". (http://lcweb.loc.gov/cds/lcsh.html).

[5] "Medical Subject Headings (MeSH)". (http://www.nlm.nih.gov/mesh/meshhome.html).

[6] Christine Froidevaux Alain Bidault and Brigitte Safar. "Repairing Queries in a Mediator Approach". In *Proceedings of the ECAI'02*, Lyon, France, 2002.

[7] Bernd Amann and Irini Fundulaki. "Integrating Ontologies and Thesauri to Build RDF Schemas". In *Proceedings of the Third European Conference for Digital Libraries ECDL'99*, Paris, France, 1999.

[8] S. Amba. "Automatic Linking of Thesauri". In *Proceeding of SIGIR'96*, Zurich, Switzerland, 1996. ACM Press.

[9] José Luis Ambite, Naveen Ashish, Greg Barish, Craig A. Knoblock, Steven Minton, Pragnesh J. Modi, Ion Muslea, Andrew Philpot, and Sheila Tejada. Ariadne: a system for constructing mediators for Internet sources. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 561–563, 1998.

[10] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *"Modern Information Retrieval"*. ACM Press, Addison-Wesley, 1999.

[11] F. Bancilhon and N. Spyratos. "Update Semantics of Relational Views". *ACM Transactions on Database Systems*, December 1981.

[12] V. Richard Benjamins and Dieter Fensel. "Community is Knowledge! in $(KA)^2$". Submitted to KAW'98.

[13] Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web". *Scientific American*, May 2001.

[14] Magnus Boman, Janis A. Bubenko, Paul Johannesson, and Benkt Wangler. *"Conceptual Modelling"*. Prentice-Hall, 1997.

[15] George Boolos. *"Logic, Logic and Logic"*. Harvard University Press, 1998.

[16] E. Borger. *"Computability, Complexity, Logic"*. Amsterdam: North-Holland, 1989.

[17] Ronald Brachman and James Schmolze. "An Overview of the KL-ONE Knowledge Representation System". *Cognitive Science*, 9(2), 1985.

[18] Dan Brickley and R. V. Guha. "Resource Description Framework (RDF) Schema specification: Proposed Recommendation, W3C", March 1999. http://www.w3.org/TR/1999/PR-rdf-schema-19990303.

[19] P. Briebricher, N. Fuhr, G. Knorz, G. Lustig, and M. Schwantner. "The Automatic Indexing System AIR/PHYS-from research to application". In *11th International Conference on Research and Development in Information Retrieval*, pages 333–342, Grenoble, France, 1988. Presses Universitaires de Grenoble.

[20] Mario Bunge. *Scientific Research I. The Search for Systems*, 3(1), 1967.

[21] D. Calvanese, G. de Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. "Description Logic Framework for Information Integration". In *Proceedings of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98)*, 1998.

[22] H. Chalupsky. "Ontomorph: A Translation System for Symbolic Knowledge". In *Principles of Knowledge Representation and Reasoning, Procs of the Seventh International Conference KR-2000*, San Francisco, CA, 2000.

[23] Chen-Chuan K. Chang and Héctor García-Molina. "Mind Your Vocabulary: Query Mapping Across Heterogeneous Information Sources". In *Proc. of the ACM SIGMOD*, pages 335–346, 1999.

[24] Chen-Chuan K. Chang and Héctor García-Molina. "Approximate query mapping: Accounting for translation closeness". *VLDB Journal*, 10(2-3), 2000.

[25] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. "The TSIMMIS project: Integration of Heterogeneous Information Sources". In *Proceedings of IPSJ*, Tokyo, Japan, October 1994.

[26] Peter Clark. "Some Ongoing KBS/Ontology Projects and Groups", 2002. http://www.cs.utexas.edu/users/mfkb/related.html.

[27] Sophie Cluet, Claude Delobel, Jérôme Siméon, and Katarzyna Smaga. "Your mediators need data conversion!". In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1998.

[28] E. F. Codd. "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM*, 13(6):377–387, 1970.

[29] Princeton University Cognitive Science Laboratory. "WordNet: A Lexical Database for the English Language". (http://www.cogsci.princeton.edu/ wn).

[30] Panos Constantopoulos, Manos Theodorakis, and Yannis Tzitzikas. "Developing Hypermedia Over an Information Repository ", 1996. 2nd WorkShop on Open Hypermedia Systems at Hypertext'96, Washington, DC, USA,March 16-20, 1996.

[31] Panos Constantopoulos and Yannis Tzitzikas. "Context-Driven Information Base Update". In Panos Constantopoulos, John Mylopoulos, and Yannis Vassiliou, editors, *Advanced Information Systems Engineering - Proceedings of the 8th International Conference, CAiSE'96*, pages 319–344, Heraklion, Crete, Greece, May 1996. Springer-Verlag.

[32] N. Craswell, D. Hawking, and P. Thistlewaite. "Merging Results from Isolated Search Engines". In *Proceedings of the Tenth Australasian Database Conference*, 1999.

[33] S. Decker, M. Erdmann, D. Fensel, and R. Studer. "Ontobroker: Ontology based Access to Distributed and Semi-Structured Information ". In *Semantic Issues in Multimedia Systems*. Kluwer Academic Publisher, 1999.

[34] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. "Learning to Map between Ontologies on the Semantic Web". In *Proceedings of the World-Wide Web Conference (WWW-2002)*, 2002.

[35] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. *"Reasoning in Description Logics"*, chapter 1. CSLI Publications, 1997.

[36] Elizabeth B. Duncan. "A Faceted Approach to Hypertext", 1989. in Ray McAleese eds., HYPERTEXT: theory into practice, BSP,1989.

[37] Oliver M. Duschka and Michael R. Genesereth. "Answering Recursive Queries Using Views". In *Proceedings of AAAI, 1997.*, 1997.

[38] Oliver M. Duschka and Michael R. Genesereth. "Query Planning in Infomaster". In *Proceedings of the Twelfth Annual ACM Symposium on Applied Computing, SAC'97*, San Jose, February 1997.

[39] Ronald Fagin. "Combining Fuzzy Information From Multiple Systems". *Journal of Computer and System Sciences*, 58, 1999.

[40] Yizhong Fan and Susan Gauch. "Adaptive Agents for Information Gathering from Multiple, Distributed Information Sources". In *1999 AAAI Symposium on Intelligent Agents in Cyberspace*, Stanford University, March 1999.

[41] Jacques Ferber. *"Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence"*. Addison-Wesley, 1999.

[42] T. Finin, Richard Fritzson, Don McKay, and Robin McEntire. "KQML as an Agent Communication Language". In *3rd International Conference on Information and Knowledge Management, CIKM94*. ACM Press, December 1994.

[43] Norbert Fuhr. "A Decision-Theoretic Approach to Database Selection in Networkded IR". *ACM Transactions on Information Systems*, 17(3), July 1999.

[44] Antony Galton. *"Logic for Information Technology"*. John Wiley & Sons, 1990.

[45] B. Ganter and R. Wille. *"Formal Concept Analysis: Mathematical Foundations"*. Springer-Verlag, Heidelberg, 1999.

[46] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey Ullman, Vasilis Vassalos, and Jennifer Widom. "The TSIMMIS Approach to Mediation: Data Models and Languages". In *Proceedings of IPSJ*, Tokyo, Japan, October 1994.

[47] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *"Database System Implementation"*, chapter 11. Prentice Hall, 2000.

[48] M. R. Genesareth, A. M. Keller, and O. Duschka. "Infomaster: An Information Integration System". In *Proceedings of 1997 ACM SIGMOD Conference*, May 1997.

[49] J. R. Giles. *Introduction to the Analysis of Metric Spaces*. Cambridge University Press, 1987.

[50] George Gratzer. *"Lattice Theory: First Concepts and Distributive Lattices"*. W.H.Freeman and Company, 1971.

[51] L. Gravano and H. Garcia-Molina. "Generalizing GlOSS To Vector-Space Databases and Broker Hierarchies". In *Proc 21st VLDB Conf.*, Zurich, Switzerland, 1996.

[52] Thomas R. Gruber. "A Translation Approach to Portable Ontologies". *Knowledge Acquisition*, 5(2):199–220, 1993.

[53] Thomas R. Gruber. "Towards Principles for the Design of Ontologies Used for Knowledge Sharing ". In *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers, 1994.

[54] Nicola Guarino. "Understanding, Building and Using Ontologies: A Commentary to "Using Explicit Ontologies in KBS Development"". *International Journal of Human and Computer Studies*, 46:293–310, 1997.

[55] Nicola Guarino. "Formal Ontology and Information Systems". In *Proceedings of FOIS'98*, Trento, Italy, June 1998. Amsterdam, IOS Press.

[56] Nicola Guarino. "Some Ontological Principles for Designing Upper Level Lexical Resources". In *Proceedings of first int. Conf. on Language Resources and Evaluation*, Granada, Spain, May 1998.

[57] Nicola Guarino, Claudio Masolo, and Guido Vetere. "OntoSeek: Content-based Access to the Web". *IEEE Inteligent Systems*, pages 70–80, May,June 1999.

[58] Jeff Heflin, James Hendler, and Sean Luke. "Coping with Changing Ontologies in a Distributed Environment". In *Proceedings of AAAI-99 Workshop on Ontology Management*, 1999.

[59] Heiko Helleg, Jurgen Krause, Thomas Mandl, Jutta Marx, Matthias Muller, Peter Mutschke, and Robert Strogen. "Treatment of Semantic Heterogeneity in Information Retrieval". Technical Report 23, Social Science Information Centre, May 2001. (http://www.gesis.org/en/publications/reports/iz_working_papers/).

[60] A. Howe and D. Dreilinger. "SavvySearch: A MetaSearch Engine that Learns Which Search Engines to Query". *AI Magazine*, 18(2), 1997.

[61] Richard Hull and Roger King. "Semantic Database Modeling". *ACM Computing Surveys*, 19(3):202–260, September 1987.

[62] Getty Research Institute. "Art & Architecture Thesaurus". (http://www.getty.edu/research/tools/vocabulary/aat/).

[63] Getty Research Institute. "The Getty Thesaurus of Geographic Names (TGN)". http://shiva.pub.getty.edu/tgn_browser/.

[64] J. Jannink, S. Pichai, D. Verheijen, and G. Wiederhold. "Encapsulation and composition of ontologies". In *Proceedings of AAAI Workshop on AI & Information Integration, 1998*, 1998.

[65] G. Karvounarakis, V. Christophides, D. Plexousakis, and S. Alexaki. Querying RDF Descriptions for Community Web Portals. In *The National French Conference on Datadases (BDA'01)*, Agadir, Maroco, October 2001.

[66] Vipul Kashyap and Amit Sheth. "Semantic and Schematic Similarities between Database Objects: A Context-based Approach". *VLDB Journal*, 5(4), 1996.

[67] Vipul Kashyap and Amit Sheth. "Semantic Heterogeneity in Global Information Systems: the Role of Metadata, Context and Ontologies ". In *Cooperative Information Systems: Trends and Directions*. Academic Press, 1998.

[68] Raili Kauppi. "Einfuhrung in die Theorie der Begriffssysteme". *Acta Universitatis Tamperensis, Ser A, Vol 15, University of Tampere*, 1967.

[69] C. Knoblock, Yigal Arens, and Chun-Nan Hsu. "Cooperating Agents for Information Retrieval". In *Procedings of the Second International Conference on Cooperative Information Systems*, Toronto, Ontario, Canada, 1994.

[70] Robert R. Korfhage. *"Information Storage and Retrieval"*. John Wiley & Sons, 1997.

[71] Information Systems Laboratory. "The Semantic Index System (SIS)". Institute of Computer Science Foundation for Research and Technology Hellas. (http://zeus.ics.forth.gr/forth/ics/isl/r-d-activities/semantic_index_system.html).

[72] M. Lacher and G. Groh. "Facilitating the Exchange of Explicit Knowledge Through Ontology Mappings". In *Proceedings of the 14th Int. FLAIRS Conference*, 2001.

[73] Veronique Lattes and Marie-Christine Rousset. "The use of CARIN languagage and algorithms for Information Integration: the PISCEL project". In *Proceedings of Second International and Interdisciplinary Workshop on Intelligent Information Integration*, Brighton Centre, Brighton, UK, August 1998.

[74] Fritz Lehmann. "Semantic Networks". In *Semantic Networks in Artificial Intelligence*, pages 1–50. Pergamon Press, 1992.

[75] D. B. Lenat. "CYC: A Large-Scale Investment in Knowledge Infrastructure". *Communications of the ACM*, 38(11), 1995.

[76] Alon Y. Levy. "Answering Queries Using Views: A Survey", 1996.

[77] Alon Y. Levy, Divesh Srivastava, and Thomas Kirk. "Data Model and Query Evaluation in Global Information Systems". *Journal of Intelligent Information Systems*, 5(2), 1995.

[78] P. H. Lindsay and D. A. Norman. *Human Information Processing*. Academic press, New York, 1977.

[79] Sean Luke, Lee Spector, David Rager, and Jim Hendler. "Ontology-based Web Agents". In *Proceedings of First International Conference on Autonomous Agents*, 1997. (http://www.cs.umd.edu/projects/plus/SHOE/).

[80] Aimilia Maganaraki, Sofia Alexaki, Vassilis Christophides, and Dimitris Plexousakis. "Benchmarking RDF Schemas for the Semantic Web". In *Proceedings of ICSW'2002*, Sardinia, Italy, June 2002.

[81] D. Maluf and G. Wiederhold. *"Abstraction of Representation for Interoperation"*. 1997.

[82] Amanda Maple. "Faceted Access: A Review of the Literature", 1995. http://theme.music.indiana.edu/tech_s/mla/facacc.rev.

[83] Zygmunt Mazur. "Models of a Distributed Information Retrieval System Based on Thesauri with Weights". *Information Processing and Management*, 30(1):61–77, 1994.

[84] G. McCalla, F. Searwar, J. Thomson, J. Collins, Y. Sun, and B. Zhou. "Analogical User Modelling: A Case Study in Individualized Information Filtering". In *Proceedings of the 5th International Conference on User Modeling*, Kailuna-Kona, Hawaii, USA, January 1996.

[85] Deborah L. McGuinness. "Ontological Issues for Knowledge-Enhanced Search". In *Proceedings of FOIS'98*, Trento, Italy, June 1998. Amsterdam, IOS Press.

[86] Carlo Meghini and Umberto Straccia. "A Relevance Terminological Logic for Information Retrieval". In *Proceedings of SIGIR'96*, Zurich, Switzerland, August 1996.

[87] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. "OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Preexisting Ontologies.". In *Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS'96)*, Brussels, Belgium, June 1996. IEEE Computer Society Press.

[88] Hafedh Mili and Roy Rada. "Merging Thesauri: Principles and Evaluation". *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 10(2), March 1988.

[89] Simon Milton, Ed. Kazmierczak, and Chris Keen. "An Ontological Study of Data Modelling Languages using the Chisholm's Ontology ". In *11th European-Japanese Conference on Information Modelling and Knowledge Bases*, Maribor, Slovenia, May 2001.

[90] Marvin Minsky. "A Framework for Representing Knowledge". In *Mind Design*. The MIT Press, 1981.

[91] P. Mitra, G. Wiederhold, and J. Jannink. "Semi-automatic Integration of Knowledge sources". In *Proc. of the 2nd Int. Conf. On Information FUSION*, 1999.

[92] Prasenjit Mitra, Gio Wiederhold, and J. Jannink. "Semi-automatic Integration of Knowledge Sources". In *Proceedings of Fusion'99*, 1999.

[93] Prasenjit Mitra, Gio Wiederhold, and Martin Kersten. " A Graph-oriented Model for Articulation of Ontology Interdependencies". *Lecture Notes in Computer Science*, 1777, 2000.

[94] G. Modica, A. Gal, and H.M. Jamil. "The Use of Machine-Generated Ontologies in Dynamic Information Seeking". In *Proceedings of the Sixth International Conference on Cooperative Information Systems (CoopIS-2001)*, Trento, Italy, September 2001.

[95] John Mylopoulos. "Information Modeling in the Time of the Revolution". *Information Systems*, 23(3):127–155, 1998.

[96] Nils J. Nilsson. *"Artificial Intelligence - A New Synthesis"*. Morgan Kaufmann, 1998.

[97] Henric Nottelmann and Norbert Fuhr. "MIND: An Architecture for Multimedia Information Retrieval in Federated Digital Libraries". In *DELOS Workshop on Interoperability in Digital Libraries*, Darmstadt, Germany, September 2001.

[98] N.F. Noy and M.A. Musen. "PROMPT: Algorithm and Tool for Automated Ontology Merging and Allignment". In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2000.

[99] Esko Nuutila. *"Efficient Transitive Closure Computation in Large Digraphs"*. PhD thesis, Acta Polytechnica Scandinavica, Helsinki, 1995. (url = http://www.cs.hut.fi/~enu/thesis.html).

[100] C. Paice. "A Thesaural Model of Information Retrieval". *Information Processing and Management*, 27(5):433–447, 1991.

[101] B. Peterson, W.A. Andersen, and J. Engel. "Knowledge Bus: Generating Application-focused Databases from Large Ontologies". In *Proceedings of the 5th Workshop KRDB-98*, Seattle, WA, USA, May 1998.

[102] H. Sofia Pinto, A. Gomez-Perez, and Joao Martins. "Some Issues on Ontology Integration". In *Proc. of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5)*, Stockholm, Sweden, 1999.

[103] Ruben Prieto-Diaz. "Classification of Reusable Modules". In *Software Reusability. Volume I*, chapter 4, pages 99–123. acm press, 1989.

[104] Ruben Prieto-Diaz. "Implementing Faceted Classification for Software Reuse". *Communications of the ACM*, 34(5), 1991.

[105] Roy Rada, L. Darden, and J. Eng. "Relating two Knowledge Bases: The Role of Identity and Part-whole". *The Role of Language in Problem Solving*, 2(2):71–91, March 1987.

[106] S. R. Ranganathan. "The Colon Classification". In Susan Artandi, editor, *Vol IV of the Rutgers Series on Systems for the Intellectual Organization of Information*. New Brunswick, NJ: Graduate School of Library Science, Rutgers University, 1965.

[107] Berthier A. Ribeiro-Neto and Richard Muntz. "A Belief Network Model for IR". In *SIGIR'96*, Zurich, Switzerland, 1996.

[108] S. E. Robertson and K. Sparck Jones. "Relevance Weighting of Search Terms". *Journal of the American Society for Information Sciences*, 27(3), 1976.

[109] J. Robinson. "A Machine-Oriented Logic Based on the Resolution Principle". *Journal of the Association for Computing Machinery*, 12(1):23–41, 1965.

[110] Thomas Rolleke and Norbert Fuhr. "Retrieval of Complex Objects Using a Four-Valued Logic". In *Proceedings of SIGIR'96*, Zurich, Switzerland, August 1996.

[111] I. Ryutaro, T. Hideaki, and H. Shinichi. "Rule Induction for Concept Hierarchy Allignment". In *Proceedings of the 2nd Workshop on Ontology Learning at the 17th Int. Conf. on AI (IJCAI)*, 2001.

[112] Giovanni M. Sacco. "Dynamic Taxonomies: A Model for Large Information Bases". *IEEE Transactions on Knowledge and Data Engineering*, 12(3), May 2000.

[113] G. Salton. *"Introduction to Modern Information Retrieval"*. McGraw-Hill, 1983.

[114] G. Salton and M. E. Lesk. "Computer Evaluation of Indexing and Text Processing". *Journal of the ACM*, 15(1), January 1968.

[115] E. Selberg and O. Etzioni. "Multi-Service Search and Comparison Using the MetaCrawler". In *Proceedings of the 1995 World Wide Web Conference*, December 1995.

[116] Marios Sintichakis and Panos Constantopoulos. "A Method for Monolingual Thesauri Merging". In *Proceedings of 20th International Conference on Research and Development in Information Retrieval, ACM SIGIR'97*, Philadelphia, PA, USA, July 1997.

[117] Nicolas Spyratos. "The Partition Model: A Functional Approach". Technical Report No. 430, INRIA, 1985.

[118] Nicolas Spyratos. "The Partition Model: A Deductive Database Model". *ACM Transactions on Database Systems*, 12(1):1–37, 1987.

[119] Nicolas Spyratos and Christophe Lecluse. "Incorporating Functional Dependencies in Deductive Query Answering". In *Proceedings of the Third International Conference on Data Engineering, ICDE-87*, February 1987.

[120] International Organization For Standardization. "Documentation - Guidelines for the establishment and development of monolingual thesauri", 1988. Ref. No ISO 2788-1986.

[121] V. S. Subrahmanian, S. Adah, A. Brink, R. Emery, A. Rajput, R. Ross, T. Rogers, and C. Ward. "HERMES: A Heterogeneous Reasoning and Mediator System", 1996. (www.cs.umd.edu/projects/hermes/ overview/paper).

[122] J. H. Sugarman. *"The development of a classification system for information storage and retrieval purposes based upon a model of scientific knowledge generation"*. PhD thesis, School of Education, Boston University, 1981.

[123] Bill Swartout, Ramesh Patil, Kevin Knight, and Tom Russ. "Towards Distributed Use of Large-Scale Ontologies", 1996. USC/Information Sciences Institute, CA, September 27, 1996, (http://www.isi.edu/isd/banff_paper/ Banff_final_web/Banff_96_final_2.html).

[124] Hideaki Takeda, Kenji Iino, and Toyoaki Nishida. "Ontology-supported Agent Communication". In *AAAI Spring Symposium on Information Gathering*, 1995.

[125] H. R. Turtle and B. W. Croft. "Evaluation of an Inference Netword-Based Retrieval Model". *ACM Transactions on Information Systems*, 9(3):187–222, 1991.

[126] Yannis Tzitzikas. "Exploiting Semantic Networks to Generate Hypermedia", 1996. 3rd Doctoral Consortium on Advanced Information Systems Engineering, Heraklion, Crete, Greece (20-21 May 1996).

[127] Yannis Tzitzikas. "Democratic Data Fusion for Information Retrieval Mediators". In *ACS/IEEE International Conference on Computer Systems and Applications*, Beirut, Lebanon, June 2001.

[128] Yannis Tzitzikas, Nicolas Spyratos, and Panos Constantopoulos. "Deriving Valid Expressions from Ontology Definitions". In *11th European-Japanese Conference on Information Modelling and Knowledge Bases*, Maribor, Slovenia, May 2001.

[129] Yannis Tzitzikas, Nicolas Spyratos, and Panos Constantopoulos. "Faceted Ontologies for Web Portals". Technical Report TR-293, Institute of Computer Science-FORTH, October 2001.

[130] Yannis Tzitzikas, Nicolas Spyratos, and Panos Constantopoulos. "Mediators over Ontology-based Information Sources". In *Second International Conference on Web Information Systems Engineering, WISE 2001*, Kyoto, Japan, December 2001.

[131] Yannis Tzitzikas, Nicolas Spyratos, and Panos Constantopoulos. "Mediators over Ontology-based Information Sources", December 2001. (submitted for publication in journal).

[132] Yannis Tzitzikas, Nicolas Spyratos, and Panos Constantopoulos. "Query Evaluation for Mediators over Web Catalogs". In *27th International Conference on Information and Communication Technologies and Programming, ICT&P'02*, Primorsko, Boulgaria, June 2002.

[133] Yannis Tzitzikas, Nicolas Spyratos, and Panos Constantopoulos. "Query Translation for Mediators over Ontology-based Information Sources". In *Second Hellenic Conference on Artificial Intelligence, SETN-2002*, Thessaloniki, Greece, April 2002.

[134] Yannis Tzitzikas, Nicolas Spyratos, Panos Constantopoulos, and Anastasia Analyti. "Extended Faceted Ontologies". In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering, CAiSE-02*, Toronto, Canada, May 2002. (short paper).

[135] Jeffrey D. Ullman. "Information integration using logical views". In *In Proc. of the 6th Int. Conf. on Database Theory (ICDT-97)*, 1997.

[136] Frank van Harmelen and Dieter Fensel. "Practical Knowledge Representation for the Web". In *Workshop on Intelligent Information Integration, IJCAI'99*, 1999.

[137] B. C. Vickery. "Knowledge Representation: A Brief Review". *Journal of Documentation*, 42(3):145–159, 1986.

[138] E. Vorhees, N. Gupta, and B. Johnson-Laird. "The Collection Fusion Problem". In *Proceedings of the Third Text Retrieval Conference (TREC-3)*, Gaithersburg, MD, 1995.

[139] Ellen Vorhees. "Multiple Search Engines in Database Merging", 1997. DL 97.

[140] G. Wiederhold. "Mediators in the Architecture of Future Information Systems". *IEEE Computer*, 25:38–49, 1992.

[141] Gio Wiederhold. "An Algebra for Ontology Composition". In *Proceedings of 1994 Monterey Workshop on Formal Methods*, September 1994.

[142] W. A. Woods. "Understanding Subsumption and Taxonomy". In *Principles of Semantic Networks*, chapter 1. Morgan Kaufmann Publishers, 1991.

[143] Yiming Yang and Christofer G.Chute. "An Example-based Mapping Method for Text Categorization and Retrieval". *ACM Transactions on Information Systems*, 12(3):252–277, July 1994.

[144] R. Yerneni, Chen Li, H.Garcia-Molina, and J.Ullman. "Computing capabilities of mediators". In *Proceedings of ACM SIGMOD'99*, Philadelphia, 1999.