Query Translation for Mediators over Ontology-Based Information Sources

Yannis Tzitzikas[1], Nicolas Spyratos[2*], and Panos Constantopoulos[1]

[1] Department of Computer Science, University of Crete, Greece, and
Institute of Computer Science, ICS-FORTH
{tzitzik, panos}@csi.forth.gr
[2] Laboratoire de Recherche en Informatique, Universite de Paris-Sud, France
spyratos@lri.fr

Abstract. We propose a model for providing integrated and unified access to multiple information sources. Each source comprises: (a) an *ontology* i.e. a set of terms structured by a subsumption relation, and (b) a *database* that stores descriptions of objects using terms of the ontology. We assume that different sources may use different ontologies, i.e., different terminologies with terms that correspond to different natural languages or to different levels of granularity. Information integration is obtained through a mediator comprising two parts: (a) an *ontology*, and (b) a set of *articulations* to the sources, where an articulation to a source is a set of relationships between terms of the mediator and terms of that source. Information requests (queries) are addressed to the mediator whose task is to analyze each query into sub-queries, send them to the appropriate sources, then combine the results to answer the original query. We study the querying and answering process in this model and we focus on query translation between the mediator and the sources.

1 Introduction

The need for integrated and unified access to multiple information sources has stimulated the research on *mediators* (initially proposed in [1]). Roughly a mediator 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 expressed over the ontology of the mediator. Upon receiving a user query, the mediator has to query 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 has to appropriately combine the returned results and deliver the final answer to the user.

In this paper we consider information *sources* over a domain consisting of a denumerable set of objects. For example in the environment of the Web, the domain is the set of all Web pages, specifically, the set of all pointers to Web pages. Each source has an *ontology*, that is, a set of names, or *terms*, that are

* Work partially conducted while this author was visiting at the National Technical University of Athens, supported by the PENED/GGET project.

familiar to the users of the source, structured by a *subsumption* relation. In addition each source maintains a database storing objects that are of interest to its users. Specifically, each object in the database of a source is associated (indexed) with a description (conjunction of terms) over the ontology of that source. A user who looks for objects of interest can browse the ontology of a source until he reaches the desired objects, or he can query the source by submitting a boolean expression of terms. The source will then return the appropriate set of objects. Specifically, the general purpose catalogs of the Web, such as Yahoo! or Open Directory [1], the domain specific catalogs/gateways (e.g. for medicine, physics, tourism), as well as the personal bookmarks of the Web browsers can be considered as examples of such sources.

However, although several sources may carry information about the same domain, they usually employ different ontologies with terms that correspond to different natural languages, or to different levels of granularity, and so on. For example Figure 1 sketches graphically the ontologies of two sources S_1 and S_2 which provide access to electronic products. Now suppose that we want to provide unified access to the databases of these sources through an ontology such as the one shown in Figure 1.(c). A mediator is a system that can bridge these heterogeneities and provide a unified access to a set of such sources. Specifically, a mediator has an *ontology* with terminology and structuring that reflects the needs of its potential users, but does not maintain a database of objects. Instead, the mediator has a number of *articulations* to other sources. An articulation to a source is a set of relationships between the terms of the mediator and the terms of the source. These relationships are defined by the designer of the mediator. The mediator uses the articulations in order to translate queries over its own ontology to queries over the ontologies of the articulated sources. Figure 2 shows the general architecture of a mediator.

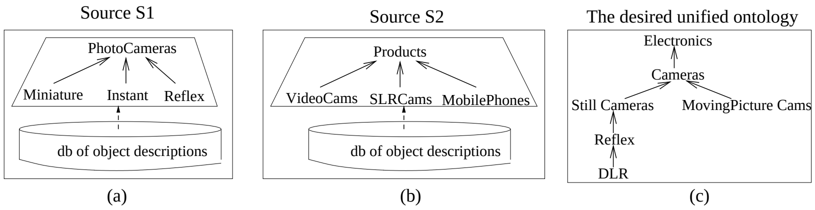


Fig. 1. Two sources providing access to electronic products and the desired unified ontology

In this paper we describe this mediator-based architecture and we present algorithms for source selection and query translation. The objective that governs these tasks is to minimize the "semantic difference" between the received query and the query finally answered by the mediator. Our approach can complement mediators over relational databases so that to support approximate translations of values that are partially ordered. Another essential feature that distinguish our work is that our mediators can operate differently for those users (or information

[1] http://dmoz.org

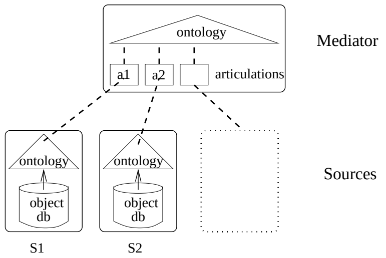


Fig. 2. The mediator architecture

needs) that focus on recall and those that focus on precision. Our model can be used for defining user views over existing Web catalogs.

The paper is organized as follows: Section 2 describes the information sources, and Section 3 the mediators. Section 4 defines the query translation problem, and Section 5 presents the algorithms for translating queries. Finally, Section 6 discusses related work and concludes the paper.

2 The Information Sources

Let *Obj* denote the set of all objects of a domain (e.g. the set of all pointers to Web pages). Each source has an *ontology*, i.e. 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 *covered* or *subsumed* by b if $a \preceq b$; we also say that b *covers* or *subsumes* a, e.g. `Databases` $\preceq$ `Informatics`, `Canaries` $\preceq$ `Birds`. We write $a \sim b$ if both $a \preceq b$ and $b \preceq a$ hold, e.g., `Computer Science` $\sim$ `Informatics`. Note that $\sim$ is an equivalence relation over T and that $\preceq$ is a partial order over the equivalence classes of T.

In addition each source has a stored *interpretation* I of its terminology, i.e. a function $I : T \to 2^{Obj}$ that associates each term of T with a set of objects [2]. Figure 3 shows an example of a source.

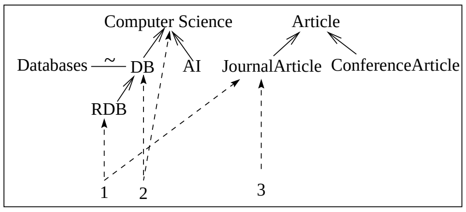


Fig. 3. Graphical representation of a source

Here the stored objects are denoted by the natural numbers 1,2 and 3, dashed oriented lines are used to connect each term t with the elements of $I(t)$, solid arrows indicate subsumption, and solid non-oriented lines equivalence.

Concerning queries, each source responds to queries over its own terminology:

[2] We use the symbol 2^{Obj} to denote the power set of *Obj*.

Def. 1. Let T be a terminology. A *query* over T is any string derived by the following grammar, where t is a term of T: $q ::= \ t \mid q \wedge q' \mid q \vee q' \mid q \wedge \neg q' \mid (q)$. We will denote by Q_T the set of all queries over the terminology T.

Any interpretation I of T can be extended to an interpretation $\hat{I}$ of Q_T, as follows: $\hat{I}(q \wedge q') = \hat{I}(q) \cap \hat{I}(q')$, $\hat{I}(q \vee q') = \hat{I}(q) \cup \hat{I}(q')$, $\hat{I}(q \wedge \neg q') = \hat{I}(q) \setminus \hat{I}(q')$, $\hat{I}(t) = I(t)$. For simplicity, hereafter we shall use the symbol I to denote both the interpretation and its extension over queries. Given two interpretations I, I' of T, we call I less or equal than 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 on interpretations.

Now, the interpretation that a source uses for answering queries must respect in some sense the structure of its ontology (i.e. the relation $\preceq$). For example, assume that a source has stored two sets of objects under the terms `Databases` and `AI`, and no objects under the term `Computer Science` - although the latter term subsumes the former two. However, such a stored interpretation is acceptable since we can "satisfy" $\preceq$ by defining the interpretation of `Computer Science` to be the union of the sets of objects associated with `Databases` and `AI`. In order to define this formally we introduce the notion of *model*.

Def. 2. An interpretation I of T is a *model* of $(T, \preceq)$ if for each t, t' in T, if $t \preceq t'$ then $I(t) \subseteq I(t')$.

Clearly, we can always extend an interpretation I to a model of $(T, \preceq)$, and we assume that each source answers queries from a model of its stored interpretation. However in order to respond to a query, a source must select one among several possible models, and as such we assume the minimal model which is greater than I, denoted by $\bar{I}$ (and it can be easily proved that there is always a unique minimal model). We shall refer to this model as the *answer model*. Thus if a source $[(T, \preceq), I]$ receives a query $q \in Q_T$ it returns the set of objects $\bar{I}(q)$.

For answering queries there are two distinct approaches. In the first approach the source computes and stores the answer model $\bar{I}$. However, whenever the ontology or the interpretation I changes, $\bar{I}$ must be appropriately updated. This requires an efficient method for handling updates since recomputing the whole $\bar{I}$ from scratch would be inefficient. In the second approach, only the interpretation I is stored, and whenever a query q is received the source computes and returns $\bar{I}(q)$. This evaluation is based on the (easily proved) proposition that for any $t \in T$: $\bar{I}(t) = \bigcup\{I(s) \mid s \preceq t\}$. However, evaluating $\bar{I}(t)$ in this manner requires computing the transitive closure of $\preceq$. More about the implementation and the inference mechanisms of a source can be found in [2].

3 The Mediator

Suppose that we want to build a mediator M over a set of sources S_1,...,S_k. Even if all sources carry information about the same domain, they usually employ different ontologies, i.e. different terminologies with terms that correspond to different natural languages, or to different levels of granularity, and so on. A mediator M is a system that can bridge these heterogeneities and provide a unified access to a set of such sources. Specifically a mediator M has an ontology $(T, \preceq)$ with terminology and structuring that reflects the needs of its potential users. For achieving integration we enrich the mediator with a set of relationships that relate its terms with the terms of the sources. These relationships, or *articulations*, are defined by the designer of the mediator. Formally:

Def. 3. A mediator M over k sources $S_1=[(T_1, \preceq_1), I_1]$,..., $S_k=[(T_k, \preceq_k), I_k]$ consists of:
1) an ontology $(T, \preceq)$, and
2) a set of *articulations* a_i, one for each source S_i. Each a_i is a subsumption relation over $T \cup T_i$

For example, consider the sources S_1 and S_2 shown in Figure 4 and suppose that we want to provide access to these sources through a mediator M with ontology as shown this figure. For achieving integration we enrich the mediator with two articulations a_1 and a_2:

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

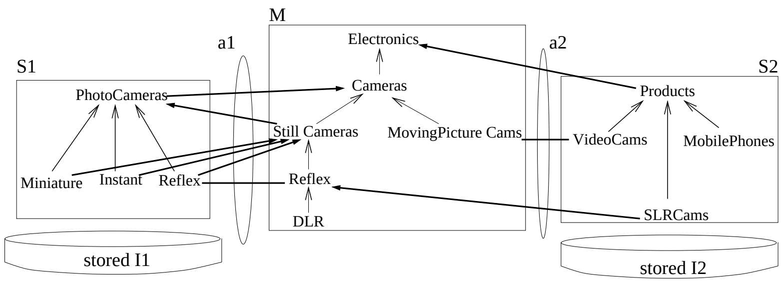


Fig. 4. A mediator over two catalogs of electronic products

3.1 The Answer Model of the Mediator

In this section we define the interpretation that M uses for answering queries. Given an articulation a_i, we will use R_i to denote those terms of T_i that appear in a_i (clearly $R_i \subseteq T_i$). Now, let $G = (T_G, \preceq_G)$ denote the ontology defined as follows:

$$T_G = T \cup R_1 \cup ... \cup R_k \quad \text{and} \quad \preceq_G \, = \, \preceq \cup \, a_1 \cup ... \cup a_k$$

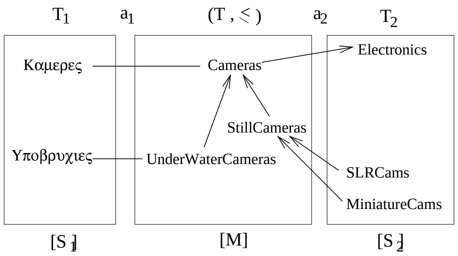
Roughly, the terminology T_G is made up from the mediator terminology T augmented by the sets R_i of source terms that the mediator knows (through the articulations a_i), while the subsumption $\preceq_G$ is the mediator subsumption $\preceq$ augmented by the articulations a_i to the sources. Note that if two terms in two sources have the same name e.g. `DB`, then by default they are considered different

($\texttt{DB}_i \not\sim \texttt{DB}_j$). This is reasonable as the same term can have different interpretations (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. Two terms are considered equivalent, e.g. $\texttt{DB}_i \sim_G \texttt{DB}_j$, only if they belong to the same equivalence class of G, e.g. if there is a term $t \in T$ such that $t \sim_G \texttt{DB}_i$ and $t \sim_G \texttt{DB}_j$.

The interpretation that the mediator uses in answering queries is defined with respect to the ontology $G = (T_G, \preceq_G)$ just defined. We call *mediator interpretation* the function $I_G : T_G \to 2^{Obj}$ defined as follows:

$$I_G(t) = \begin{cases} \emptyset & \text{if } \ t \in T \\ \bar{I}_i(t) & \text{if } \ t \in R_i \end{cases}$$

Recall that $\bar{I}_i$ denotes the answer model of the source S_i. This means that the interpretation of a term t is empty if t belongs to the terminology of the mediator, otherwise it is the set of objects that will be returned if we query the source that owns the term t. We can now extend I_G to a model of G and by $\bar{I}_G$ we will denote the minimal model which is larger than I_G (which again is unique). Upon reception of a query q, the mediator returns $\bar{I}_G(q)$, i.e., $\bar{I}_G$ is the *answer model* of M. Figure 5.(a) shows an example of a mediator with two articulations, while the table of Figure 5.(b) shows the (current) answer models $\bar{I}_1$ and $\bar{I}_2$ of the sources S_1 and S_2, and the induced answer model $\bar{I}_G$ of the mediator M.



(a)

T_G	$\bar{I}_1$	$\bar{I}_2$	$\bar{I}_G$
$K\acute{\alpha}\mu\epsilon\rho\epsilon\varsigma_1$	{1,2}		{1,2, **3,4**}
$Y\pi o\beta\rho\acute{\upsilon}\chi\iota\epsilon\varsigma_1$	{2}		{2}
Electronics$_2$		{1,3,4}	{1,**2**,3,4}
SLRCams$_2$		{3,4}	{3,4}
MiniatureCams$_2$		{1}	{1}
Cameras			**{1,2,3,4}**
Still Cameras			**{1,3,4}**
UnderWaterCameras			**{2}**

(b)

Fig. 5. A mediator with two articulations to sources S_1 and S_2

3.2 Answering Queries at the Mediator

Answering queries at the mediator is done based on the answer model $\bar{I}_G$, and we can easily see that: $\bar{I}_G(t) = \bigcup\{I(s) \mid s \preceq_G t\}$. For example, suppose that the mediator shown in Figure 4 receives the query $q = \texttt{Cameras}$. The answer to this query is defined as follows:

$$\begin{aligned} \bar{I}_G(\texttt{Cameras}) = & \ I_G(\texttt{Cameras}) \cup I_G(\texttt{StillCameras}) \cup I_G(\texttt{MovingPictureCams}) \cup \\ & I_G(\texttt{Reflex}) \cup I_G(\texttt{DLR}) \cup I_G(\texttt{PhotoCameras}_1) \cup I_G(\texttt{Miniature}_1) \cup \\ & I_G(\texttt{Instant}_1) \cup I_G(\texttt{Reflex}_1) \cup I_G(\texttt{VideoCams}_2) \cup I_G(\texttt{SLRCams}_2) \\ = & \ \emptyset \cup \emptyset \cup \emptyset \cup \emptyset \cup \emptyset \cup \\ & \bar{I}_1(\texttt{PhotoCameras}_1) \cup \bar{I}_1(\texttt{Miniature}_1) \cup \bar{I}_1(\texttt{Instant}_1) \cup \bar{I}_1(\texttt{Reflex}_1) \cup \\ & \bar{I}_2(\texttt{VideoCams}_2) \cup \bar{I}_2(\texttt{SLRCams}_2) \end{aligned}$$

However query answering cannot be performed as in the individual sources, because the interpretation I_G is not stored at the mediator. For evaluating a query the mediator has to query the underlying sources and then to combine the returned answers. Suppose first that the query is just a term t of T. In order to evaluate $\bar{I}_G(t)$, the mediator has to retrieve from each source S_i the set of objects $\bigcup\{ \bar{I}_i(s) \mid s \in R_i$ and $s \preceq_G t\}$. If we define

$$q_i(t) = \bigvee\{s \in R_i | s \preceq_G t\}$$

then for evaluating $\bar{I}_G(t)$ the mediator sends the query $q_i(t)$ to each source S_i and then it takes the union of the returned answers. That is:

$$\bar{I}_G(t) = \bigcup_{i=1..k} \bar{I}_i(q_i(t)) \tag{1}$$

In our example, we have:

$q_1(\texttt{Cameras}) = \texttt{PhotoCameras}_1 \vee \texttt{Miniature}_1 \vee \texttt{Instant}_1 \vee \texttt{Reflex}_1$

$q_2(\texttt{Cameras}) = \texttt{VideoCams}_2 \vee \texttt{SLRCams}_2$

Now, if the query q is not just a term of T then the mediator can evaluate the set $\bar{I}_G(q)$ by combining, through set operations, the interpretations of the terms that appear in q. This time, however, the evaluation presents certain problems as we will show next.

Consider, for simplicity, the case where M has decided to query only one particular source S_i. Since M will query only one source, it does not have to combine results from multiple sources. Instead, M just sends (at most) one query to S_i and then delivers the answer returned by S_i to the user. For simplicity hereafter we shall write S instead of S_i, and R instead of R_i. If M receives a query q over T, M actually sends a query q_R over R to the source S. The following question arises:

What is the relationship between the original query q and the query q_R with respect to the ontology G ?

Let us introduce a definition prior to answering this question. If q, q' are queries over G, we write $G \models q \preceq q'$, or $q \preceq_G q'$, if $I(q) \subseteq I(q')$ in every model I of G. Moreover, we write simply $q \preceq q'$ instead of $G \models q \preceq q'$, when G is understood. Two queries q, q' are called *equivalent*, denoted $q \sim q'$, if $q \preceq q'$ and $q' \preceq q$. Returning to our question, note that it may will be that $q \preceq q_R$, or $q_R \preceq q$, or both (i.e. $q_R \sim q$), or none of the three. If $q \preceq q_R$, we will call q_R *including*, if $q_R \preceq q$, we will call q_R *included*, and if $q \sim q_R$, we will call q_R *perfect*.

In the case where q is a single term $t \in T$, then the query q_R is actually the query $q_i(t) = \bigvee\{s \in R | s \preceq_G t\}$, and clearly $q_R \preceq q$. For example consider the mediator shown in Figure 4 and assume only the source S_1. If $q = \texttt{Cameras}$ then $q_R = \texttt{PhotoCameras}_1 \vee \texttt{Miniature}_1 \vee \texttt{Instant}_1 \vee \texttt{Reflex}_1$. Clearly here we have $q_R \preceq q$, thus q_R is an included translation.

Let us now consider the query $q = \mathtt{Cameras} \wedge \neg \mathtt{DLR}$. Assume that we derive the query q_R by replacing each term t appearing in q by the query $q_1(t)$. This means that $q_R = q_1(\mathtt{Cameras}) \wedge \neg q_1(\mathtt{DLR}) =$
$(\mathtt{PhotoCameras}_1 \vee \mathtt{Miniature}_1 \vee \mathtt{Instant}_1 \vee \mathtt{Reflex}_1) \wedge \neg\ \epsilon$, where ϵ denotes the empty query and clearly $\bar{I}(\epsilon) = \emptyset$ in every source. Thus, the query finally answered is the query $\mathtt{PhotoCameras}_1 \vee \mathtt{Miniature}_1 \vee \mathtt{Instant}_1 \vee \mathtt{Reflex}_1$ and notice that this is not an included query, meaning that the answer returned by M may contain objects about DLR although the user does not want them. However, note that there is actually an included query, i.e the query $q'_R = (\mathtt{PhotoCameras}_1 \vee \mathtt{Miniature}_1 \vee \mathtt{Instant}_1 \vee \mathtt{Reflex}_1) \wedge \neg \mathtt{Reflex}_1$, for which it holds: $q'_R \preceq q$, and which the mediator did not evaluate. This example shows that if we derive q_R by replacing each term appearing in q by the query $q_1(t)$, then we do not always get an included query. This problem is the subject of the subsequent section.

4 The Query Translation Problem

In this section we study the problem of *query translation*: how, for a given query q, the mediator can choose the "best" translation of q that can be answered by the underlying sources.

Roughly, among the many possible translations the mediator should select those with the "minimum" change of "semantics". Ideally, we want to find a query q_R "equivalent" to q, that is, a perfect translation. If a perfect translation is not possible, then the mediator should be able to compute the "biggest included" and the "smallest including" query, if any of these queries exist. The former is appropriate for those users (or information needs) that focus on *precision*, while the latter for those that focus on *recall*. For example consider the mediator shown in Figure 4 and assume that a user submits the query StillCameras. If the user is interested in precision, that is, if he does not want to retrieve objects which are not StillCameras, then he may prefer an answer to the query $\mathtt{Miniature}_1 \vee \mathtt{Instant}_1 \vee \mathtt{Reflex}_1$. On the other hand, if the user is interested in recall, that is, if he does not want to miss objects which are StillCameras, then he may prefer the answer to the query $\mathtt{PhotoCameras}_1$. Also note that a user interested in precision may ask an answer to the smallest including query, if the biggest included query yielded no results (e.g. if $q = \mathtt{DLR}$), or if the query yielded less objects than those wanted.

Let us now define formally the criteria for identifying the preferred translations. Assume that M receives a query q (over T) and suppose that M has decided to query only one particular source S_i. For brevity we shall hereafter write S instead of S_i and R instead of R_i. Among the possibly many including translations of q we prefer the "*smallest*", i.e. the queries in the set $up(q, R)$ defined as follows:

$$up(q, R) = glb\{q_R \in Q_R \mid q \preceq q_R\}$$

Among the possibly many included translations of q we prefer the "*biggest*", i.e. the queries in the set $down(q, R)$ defined as follows:

$$down(q, R) = lub\{q_R \in Q_R \mid q_R \preceq q\}$$

If there is a query q_R such that $q_R \in up(q,R)$ and $q_R \in down(q,R)$, then $q \sim q_R$, thus q_R is a *perfect* translation.

Concerning including translations, we can easily see that the set $\{q_R \in Q_R | q \preceq q_R\}$ may be empty, (e.g. if $q = \texttt{Electronics}$ in Figure 4). If not empty then it is infinite and $glb\{q_R \in Q_R | q \preceq q_R\}$ exists, specifically, it is the set of queries equivalent to the query:

$$\bigwedge \{q_R \in Q_R | q \preceq q_R\}$$

Analogously, the set $\{q_R \in Q_R | q_R \preceq q\}$ may be empty (e.g. if $q = \texttt{DLR}$), but if not empty then $lub\{q_R \in Q_R | q_R \preceq q\}$ exists, specifically, it is the set of queries equivalent to the query:

$$\bigvee \{q_R \in Q_R | q_R \preceq q\}$$

In conclusion, given a query q over T we would like the mediator to be able to compute a query in $down(q,R)$, and a query in $up(q,R)$, if these exist.

Now assume that M receives a query q, and decides to query a *set* of sources S_1,...,S_k. This set can be the set of all sources, or it may be the set of those sources which are online currently, or it may be the set of sources that have been selected on the basis of some criterion or cost fuction. Recall that the mediator will return an answer which will be the combination of the answers of the queries sent to the underlying sources. Thus the answer of the mediator will be an answer to a query over R where $R = R_1 \cup ... \cup R_k$. This again raises the question about the relation between q and q_R, and, as in the single source case, we would like a q_R such that $q_R \sim q$, or $q_R \in down(q,R)$, or $q_R \in up(q,R)$. Thus in order to find the biggest included or the smallest including (or the perfect) translation of q, the mediator needs the same translation mechanism as in the single source case. The only difference is that here $R = R_1 \cup ... \cup R_k$.

5 Translating and Evaluating Queries

In this section we describe the algorithms for computing the *up* and *down* of a query. Hereafter we will use $up(q,R)$ to denote any query contained in the set $up(q,R)$, and $down(q,R)$ to denote any query contained in the set $down(q,R)$. Let us first consider the case where q is a single term $t \in T$. In this case one can easily see that

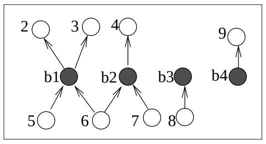
$$up(t,R) \sim \bigwedge \{r \in R | t \preceq r\} \quad \text{and} \quad down(t,R) \sim \bigvee \{r \in R | r \preceq t\}$$

The queries $up(t,R)$ and $down(t,R)$ can be derived by algorithms which traverse G and "collect" the appropriate terms of R. However if $\{r \in R | t \preceq r\} = \emptyset$ then the algorithm that derives $up(t,R)$ will return *Nil*. Analogously, if $\{r \in R | r \preceq t\} = \emptyset$ then the algorithm that derives $down(t,R)$ will return *Nill* too.

Let us now generalize to the general case where q is a query in Q_T. The following propositions allow us to derive the *up* and *down* of a query q, by synthesizing the *up* and *down* of the terms that appear in q. For brevity we shall write $up(q)$ (and $down(q)$) instead of $up(q,R)$ (and $down(q,R)$).

1. $up(q \wedge q') = up(q) \wedge up(q')$
2. $up(q \vee q') = up(q) \vee up(q')$
3. $up(q \wedge \neg q') = up(q) \wedge \neg down(q')$
4. $down(q \wedge q') = down(q) \wedge down(q')$
5. $down(q \vee q') = down(q) \vee down(q')$
6. $down(q \wedge \neg q') = down(q) \wedge \neg up(q')$

Thus in order to produce $down(q)$ or $up(q)$ the mediator parses the query q and composes the *up* or *down* of the terms that appear in q according to the above propositions. Due to limitations of space the proofs of these propositions are not included in this paper. Figure 6 gives some examples of translations where the elements of R are denoted by white circles.



$down(b1) = t_5 \vee t_6$ $\quad up(b1) = t_2 \wedge t_3$

$down(b2) = t_6 \vee t_7$ $\quad up(b2) = t_4$

$down(b1 \wedge \neg b2) = (t_5 \vee t_6) \wedge \neg t_4$ $\quad up(b1 \wedge \neg b2) = (t_2 \wedge t_3) \wedge \neg(t_6 \vee t_7)$

(a) (b)

Fig. 6. Examples of translations

However, notice that we cannot derive the desired translated query if the *up* or *down* of one subterm of q is empty (*Nil*). Nevertheless, there are cases where we can overcome this problem, and for doing so we introduce the special symbols $\top$ and $\perp$. If for a term t, we have $up(t) = \emptyset$, then we set $up(t) = \top$, and if $down(t) = \emptyset$, then we set $down(t) = \perp$. However the query q_R that we want to construct should not contain any of these special symbols. We can reduce all or some of these symbols according to the following rules:

(a) Delete the substrings "$\wedge\top$" or "$\top\wedge$". Example: $up(b_1 \wedge b_3) = (t_2 \wedge t_3) \wedge \top = t_2 \wedge t_3$

(c) Delete the substrings "$\vee\perp$" or "$\perp\vee$" . Example: $down(b_1 \vee b_4) = (t_5 \vee t_6) \vee \perp = t_5 \vee t_6$

(b) Delete the substrings "$\wedge\neg\perp$" . Example: $up(b_2 \wedge \neg b_4) = t_4 \wedge \neg\perp = t_4$

So, the process for translating a query q consists of the following steps:

(1) Parse q and synthesize the *up* and *down* of the terms that appear in q using propositions 1 to 6, seen earlier. Let q_t be the resulting query.
(2) Delete the symbols $\top$ and $\perp$ that can be reduced using rules (a), (b) and (c) seen earlier. Let $q_{t'}$ be the resulting query.
(3) If $q_{t'}$ does not contain any of the symbols $\top$ and $\perp$ then $q_R = q_{t'}$, else $q_R = Nil$.

Below we give some examples of translations assuming the ontology of Figure 6.

$down(b3) = t_8$	$up(b3) = \top = Nil$
$down(b4) = \perp = Nil$	$up(b4) = t_9$
$down(b1 \wedge b3) = (t_5 \vee t_6) \wedge t_8$	$up(b1 \wedge b3) = (t_2 \wedge t_3) \wedge \top = t_2 \wedge t_3$
$down(b1 \wedge b4) = (t_5 \vee t_6) \wedge \perp = Nil$	$up(b1 \wedge b4) = (t_2 \wedge t_3) \wedge t_9$
$down(b1 \vee b4) = (t_5 \vee t_6) \vee \perp = t_5 \vee t_6$	$up(b1 \vee b4) = (t_2 \wedge t_3) \vee t_9$
$down(b3 \vee b4) = t_8 \vee \perp = t_8$	$up(b3 \vee b4) = \top \vee t_9 = Nil$
$down(b2 \wedge \neg b4) = (t_6 \vee t_7) \wedge \neg t_9$	$up(b2 \wedge \neg b4) = t_4 \wedge \neg\perp = t_4$
$down(b3 \wedge \neg b4) = t_8 \wedge \neg t_9$	$up(b3 \wedge \neg b4) = \top \wedge \neg\perp = Nil$

Let us now summarize how the mediator operates. Whenever the mediator receives a query q, at first it computes the translation $up(q)$ or $down(q)$ (according to the user's desire), then it evaluates each term that appears in the translated query (as described in section 3.2), and finally, it combines through set operations the obtained results in order to compute the final answer.

6 Related Work - Concluding Remarks

We proposed a model for building mediators over sources which index their objects using terms from ontologies. The ontologies that we consider although simple, they fit to the content-based organizational structure of Web catalogs and portals, keyword hierarchies and personal bookmarks. Besides most of the ontologies that are used for indexing and retrieving objects are term hierarchies ([3], [4], [5]). Concerning the functionality offered by our mediators, the objective that governs the selection of the sources to be queried (and the formulation of the queries to be sent to each source) is to minimize the "semantic difference" between the received query and the query finally answered by the mediator. We defined the desired translations and we described the algorithms for computing these translations.

The concept of mediator is not new. After the introduction of the mediator concept by Wiederhold [1], many different approaches have been proposed and developed in order to build mediators over relational databases (e.g. see [6–9]), SGML documents (e.g. see [10]), information retrieval systems (e.g. see [11–15]) and Web-based sources (e.g. see [16, 17]). The techniques for building relational mediators are appropriate for rendering the structural (schema) heterogeneities of the sources transparent to the users (e.g. see the systems TSIMMIS [18], [7], HERMES [19], Information Manifold [6]). Our work can complement these techniques so that to support approximate translations of values that are partially ordered. The difference with the approach presented in [20] is that in this approach the reasoning services for supporting translations have exponential complexity (it employs very expressive description logics), as opposed to the complexity of our mediators which is clearly polynomial. This is very important because usually the ontologies (e.g. those employed by Web catalogs) contain very large numbers of terms, e.g. the catalog of Open Directory contains 300.000 terms. Our work differs from other approaches that support approximate translations. The difference with [17] is that we also support negation in queries, while the difference with the system presented in [21], [22], is that the described system merges the ontologies of all underlying sources. We propose articulation instead of merging, because merging the ontologies of all underlying sources would introduce storage and performance overheads. In addition, full integration is a laborious task which in many cases does not pay-off because the integrated ontology becomes obsolete when the involved ontologies change.

An alternative approach for query translation which offers more operation modes is given in [23], while the optimization of query evaluation, i.e. the minimization of the number of queries that the mediator has to send to the sources in order to evaluate a user query, is described in [24].

One can easily see that our approach allows the users of the Web to define views over the existing Web catalogs: by defining a mediator the user can use his own terminology in order to access and query several Web catalogs, specifically those parts of the catalogs that is of interest to him. We plan to use our approach for building mediators over sources such as Google [3]. Google allows (1) browsing through the hierarchical catalog of Open Directory, and (2) searching through natural language queries. 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 documents that fall in the category `Sciences/CS/DataStructures` in the catalog of Open Directory. A mediator over such sources will allow the user to use the ontology of the mediator in order to browse those parts of the catalogs that is of interest to him. Moreover he will be able to query the databases of these sources by natural language queries. However, this implies that the mediator will send two kinds of queries to the sources: queries evaluated over the catalog and queries which are evaluated over the contents of the pages. In this case, since each source will return an ordered set of objects, we also need a method (e.g. [14]) for fusing these orderings in order to derive the ordering to be delivered to the user.

Acknowledgements. Many thanks to Anastasia Analyti for proof reading the paper, and to Agiolina Dellaporta.

References

1. G. Wiederhold. "Mediators in the Architecture of Future Information Systems". *IEEE Computer*, 25:38–49, 1992.
2. 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.
3. Nicola Guarino. "Formal Ontology and Information Systems". In *Proceedings of FOIS'98*, Trento, Italy, June 1998. Amsterdam, IOS Press.
4. Deborah L. McGuinness. "Ontological Issues for Knowledge-Enhanced Search". In *Proceedings of FOIS'98*, Trento, Italy, June 1998. Amsterdam, IOS Press.
5. Alexander Pretschner. "Ontology Based Personalized Search". Master's thesis, Department of Electrical Engineering and Computer Science - University of Kansas, 1999.
6. 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.
7. 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.
8. Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *"Database System Implementation"*, chapter 11. Prentice Hall, 2000.

[3] www.google.com

9. R. Yerneni, Chen Li, H.Garcia-Molina, and J.Ullman. "Computing capabilities of mediators". In *Proceedings of ACM SIGMOD'99*, Philadelphia, 1999.
10. 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.
11. 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.
12. L. Gravano and H. Garcia-Molina. "Generalizing GlOSS To Vector-Space Databases and Broker Hierarchies". In *Proc 21st VLDB Conf.*, Zurich, Switzerland, 1996.
13. Norbert Fuhr. "A Decision-Theoretic Approach to Database Selection in Networkded IR". *ACM Transactions on Information Systems*, 17(3), July 1999.
14. Yannis Tzitzikas. "Democratic Data Fusion for Information Retrieval Mediators". In *ACS/IEEE International Conference on Computer Systems and Applications*, Beirut, Lebanon, June 2001.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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).
20. 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.
21. 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.
22. 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.
23. 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.
24. Yannis Tzitzikas, Nicolas Spyratos, and Panos Constantopoulos. "Mediators over Ontology-based Information Sources", December 2001. (submitted for publication in journal).