Query Evaluation in Peer-to-Peer Networks of Taxonomy-based Sources

Yannis Tzitzikas [1] Carlo Meghini

Istituto di Scienza e Tecnologie dell' Informazione [ISTI]
Consiglio Nazionale delle Ricerche [CNR], Pisa, Italy
Email : {tzitzik,meghini}@isti.cnr.it

Abstract. We consider the problem of query evaluation in Peer-to-Peer (P2P) systems that support semantic-based retrieval services. We confine ourselves to the case where the peers employ *taxonomies* for describing the contents of the objects, and *articulations*, i.e. inter-taxonomy mappings, for bridging the inevitable naming, granularity and contextual heterogeneities that may exist between the taxonomies of the sources. We identify two basic query evaluation approaches: one based on *query rewriting*, the other based on *direct* query evaluation. For each approach we present a *centralized* and a *decentralized* algorithm for carrying out the query evaluation task. Finally, we present a qualitative comparison of these algorithms and discuss further optimizations. Correctness of the algorithms presented is based on a mathematical analysis of the problem.

1 Introduction

In recent years there has been a growing interest in information integration, whose objective is to access, relate and combine data from multiple sources. This need has stimulated the research on *mediators* (initially proposed in [22]). A model for building mediators over taxonomy-based sources of the kind of Web catalogs has been proposed in [21]. According to this model, each source consists of a taxonomy and an object base, i.e. a database that indexes the objects of the domain under the terms of the taxonomy. A mediator consists of a taxonomy plus a number of *articulations* to the other sources of the network, where an articulation is actually a mapping between the terms of the mediator and the terms of the sources. In this paper we extend this model and study the more general scenario where we have a network of articulated sources. In such a network we can no longer distinguish sources to primary and secondary (i.e. mediators) as we may have mutually articulated sources, thus we actually have a peer-to-peer (P2P) system, i.e. a distributed system in which participants rely on one another for service, rather than solely relying on dedicated and often centralized infrastructure.

[1] Work done during the postdoctoral studies of the author at CNR-ISTI as an ERCIM fellow.

Many examples of P2P systems have emerged recently, most of which are wide-area, large-scale systems that provide content sharing [4], storage services [13, 18], or distributed "grid" computation [2, 1]. Smaller-scale P2P systems also exist, such as federated, serverless file systems [7, 5] and collaborative workgroup tools [3]. Existing peer-to-peer (P2P) systems have focused on specific application domains (e.g. music files) or on providing file-system-like capabilities. These systems do not yet provide semantic-based retrieval services. In most of the cases, the name of the object (e.g. the title of a music file) is the only means for describing the contents of the object. The advantage of the approach proposed in this paper, is that it also allows describing the contents of the objects with respect to taxonomies. Consequently, we can have enhanced semantic-based retrieval services.

We describe the architecture and the functioning of such a system and we focus on the query evaluation process. We carry out a mathematical analysis of the problem, which leads to the identification of the terms whose interpretation is to be considered in order to evaluate a global query. This result is exploited in the context of two different approaches to query evaluation. In the first approach, which is a kind of query rewriting, a given query is first transformed into a set of local queries, which are then executed in an optimal way to obtain the answer to the original query. In the latter approach, the query is evaluated directly, without any prior consideration of the involved local queries. For each approach, we present a centralized and a decentralized algorithm, capturing the two basic different styles for carrying out a distributed computation. The correctness of the presented algorithms directly stems from the mathematical analysis of the problem.

The remaining of this paper is organized as follows: Section 2 describes the building blocks of a network of articulated sources, i.e. sources, mediators and articulated sources; introduces the notion of network query and answer; and carries out the mathematical analysis of the problem, on which all the algorithms hinge. Section 3 focuses on query evaluation and describes several query evaluation approaches and the corresponding algorithms. It also presents a qualitative comparison of these algorithms and discuses further optimizations. Section 4 describes related work, and finally, Section 5 concludes the paper and identifies issues for further research. For reason of space, proofs are omitted, but the technical ideas behind the most important results (such as Proposition 3) are provided, along with illustrating examples.

2 The Network

Let Obj denote the set of all objects of a domain common to several information sources. A typical example of such a domain is the set of all pointers to Web pages. A network of articulated sources over Obj is a set of sources $N = \{S_1, ...S_n\}$ where each S_i falls into one of the following categories:

- *Simple sources*: they consist of a taxonomy and an object base, i.e. a database that indexes objects of Obj under the terms of the taxonomy. A simple

source accepts queries over its taxonomy and returns the objects whose index "matches" the query.
- *Mediators*: they consist of a taxonomy plus a number of articulations to other sources of the network. Again, a mediator accepts queries over its taxonomy but as it does not maintain an object base, query answering requires sending queries to the underlying sources and combining the returned results.
- *Articulated sources*: they are both simple sources and mediators, i.e. they consist of a taxonomy, an object base and a number of articulations to other sources of the network. An articulated source can behave like a simple source, like a mediator, or like a mediator which in addition to the external sources can also use its own simple source during query answering.

Clearly, simple sources and mediators are special cases (or roles) of articulated sources. In order to minimize the number of notions to be introduced, and thus attain clarity, we will first introduce simple sources, and define query-answering on them. Then, articulated sources will be defined, as an intuitive extension, having mediators as special cases.

2.1 Simple Sources

We consider a taxonomy-based conceptual modeling approach. Taxonomies is probably the oldest and most widely used conceptual modeling tool. The advantages of this conceptual modeling approach for the P2P paradigm are discussed in [20].

A simple source consists of two parts: a *taxonomy* and a stored *interpretation*.

Definition 1. A taxonomy is a pair $(T, \preceq)$ where T, the *terminology,* is a finite and non empty set of names, or *terms*, and $\preceq$ is a reflexive and transitive relation over T, modeling *subsumption* between terms.

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.

The stored interpretation I of a simple source is a function $I : T \rightarrow 2^{Obj}$ that associates each term T of the source terminology with a set of objects (we use the symbol 2^{Obj} to denote the powerset of Obj). Figure 1 shows an example of a source. In this and subsequent figures, objects are represented by natural numbers and their membership to the interpretation of a term is indicated by a dotted arrow from the object to that term; subsumption of terms is indicated by a continuous-line arrow from the subsumed term to the subsuming term, while equivalence is indicated by a continuous non-oriented line segment; finally, for readability, we do not represent the entire subsumption relation but its transitive reduction, in which reflexive and transitive arrows are suppressed. In Figure 1, objects 1 and 3 are members of the interpretation of the term

`Live`. As there are no other objects connected to `Live` with dotted arrows, $I(\texttt{Live}) = \{1, 3\}$. Moreover, the term `Metal` is subsumed by `Rock`, while the term `Rocky` is equivalent to the term `Rock`. Note that equivalence captures the notion of synonymy, and that each equivalence class simply contains alternative terms for naming a set of objects.

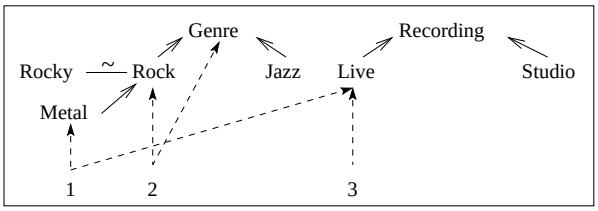

Fig. 1. Graphical representation of a source over a domain of audio files

A simple source responds to queries over its own terminology.

Definition 2. A *query* over a terminology T is any string derived by the following grammar, where t is a term of T: $q ::= \ t \mid q \wedge q' \mid q \vee q' \mid q \wedge \neg q' \mid (q) \mid \epsilon$. We will denote by Q_T the set of all queries over T.

We now proceed to define the notion of answer to a query. Clearly, a source answers queries based on the stored interpretation of its terminology. However, in order for answers to make sense, the interpretation that a source uses for answering queries must respect the structure of the source's taxonomy in the following intuitive sense: if $t \preceq t'$ then $I(t) \subseteq I(t')$. The notion of model, introduced next, captures well-behaved interpretations.

Definition 3. An interpretation I is a *model* of a taxonomy $(T, \preceq)$ if for all t, t' in T, if $t \preceq t'$ then $I(t) \subseteq I(t')$.

The interpretation of the source illustrated in Figure 1 is not a model of the source taxonomy, as $\texttt{Metal} \preceq \texttt{Genre}$, and yet $I(\texttt{Metal}) \not\subseteq I(\texttt{Genre})$. In order to overcome this problem without altering the contents of interpretations, we need to extend interpretations. However, in so doing, we want to achieve minimality, that is the extension should contain as much data is needed to be a model, and no more. For the term `Genre` in the source in Figure 1, this amounts to define the extended interpretation I' as $I'(\texttt{Genre}) = I(\texttt{Genre}) \cup I(\texttt{Rock}) \cup I(\texttt{Rocky}) \cup I(\texttt{Jazz}) \cup I(\texttt{Metal})$.

Definition 4. Given an interpretation I of T, the model of $(T, \preceq)$ *generated* by I, denoted $\bar{I}$, is given by: $\bar{I}(t) = \bigcup\{I(s) \mid s \preceq t)\}$

In order to show that the generated interpretation satisfies the minimality requirement expressed above, let us order the set of interpretations of a given terminology T by using pointwise set inclusion. Given two interpretations I, I' of T, I is less than or equal to I', in symbols $I \leq I'$, if $I(t) \subseteq I'(t)$ for each term $t \in T$. Note that $\leq$ is a partial order over interpretations. The following Proposition establishes the minimality of generated interpretations.

Proposition 1. If I is an interpretation of T then $\bar{I}$ is the unique minimal model of $(T, \preceq)$ which is greater than or equal to I.

By relying on generated interpretations, we can now define the notion of answer to queries posed against a simple source.

Definition 5. Given a simple source S_i with terminology T and interpretation I, and a query q over T, the *answer to q in S_i*, $ans_i(q)$, is inductively defined as follows: $ans_i(t) = \bar{I}(t)$, $ans_i(q \wedge q') = ans_i(q) \cap ans_i(q')$, $ans_i(q \vee q') = ans_i(q) \cup ans_i(q')$, $ans_i(q \wedge \neg q') = ans_i(q) \setminus ans_i(q')$.

The model defined so far has indeed a logical grounding, which can be disclosed by viewing terms as propositional letters. Then, each subsumption relationship $t \preceq t'$ becomes the conditional $t \rightarrow t'$, the whole subsumption relation $\preceq$ becomes a propositional theory, and the generated interpretation of each object o, $\bar{I}^{-1}(o)$, is the smallest propositional model including $I^{-1}(o)$ and satisfying $\preceq$[2]. A query is a propositional formula q, whose answer is the set of objects o whose propositional model $\bar{I}^{-1}(o)$ satisfies q.

2.2 Articulated Sources

An *articulated source* is a simple source that also maintains a number of *articulations* to other sources. An articulation to a source is a set of relationships between the terms of the articulated source and the terms of that source.

Definition 6. An *articulation* from a taxonomy $(T_i, \preceq_i)$ to a taxonomy $(T_j, \preceq_j)$, denoted by $\preceq_{ij}$, is any nonempty set of relationships $t_j \preceq_{ij} t_i$ where $t_i \in T_i$ and $t_j \in T_j$.

If $t_j \preceq_{ij} t_i$, we say that t_j *articulates* t_i. Articulations bridge the heterogeneities that may exist between two or more sources in order to provide a uniform query interface to these sources. The relationships making up an articulation are defined by the designer and are stored at the articulated source. They can be defined manually, but they can also be constructed automatically or semi-automatically in some specific cases, following a model-driven approach or a data-driven approach In this paper we do not focus on articulation design or construction. We treat this issue in [19].

Definition 7. An *articulated source* M over k sources $S_1, ..., S_k$ consists of: (1) a simple source, comprising a taxonomy $(T_M, \preceq_M)$ and a stored interpretation I_s of T_M, and (2) a set $\{a_{M,1}, ..., a_{M,k}\}$, where each $a_{M,i}$ is an *articulation* from $(T_M, \preceq_M)$ to $(T_i, \preceq_i)$.

An articulated source with an empty stored interpretation, i.e. $I(t) = \emptyset$ for all $t \in T_M$, is called a *mediator.* An articulated source with no articulations and with a nonempty interpretation is a *simple source.*

Figure 2 shows an example of a mediator over two sources that provide access to music files. The articulations $a_{M,1}$ and $a_{M,2}$ shown in this figure are given by:

[2] Note that I^{-1} and $\bar{I}^{-1}$ are functions from Obj to 2^T.

$\mathrm{a}_{M,1} = \{\texttt{Pop}_1 \preceq \texttt{Genre}, \texttt{AlternativeRock}_1 \preceq \texttt{Rock}, \texttt{HeavyMetal}_1 \preceq \texttt{Metal}\ \}$
$\mathrm{a}_{M,2} = \{\texttt{DanceMusic}_2 \preceq \texttt{Folk}, \texttt{DanceMusic}_2 \preceq \texttt{Greece}, \texttt{Cantriglie}_2 \preceq \texttt{Italy}\ \}$

$\mathrm{a}_{M,1}$ demonstrates how articulations can bridge the granularity heterogeneities that may exist between different terminologies, while $\mathrm{a}_{M,2}$ demonstrates how the articulations of the mediator can restore the context of the objects of the sources, here, the fact that the origin of all music files of S_2 is Greece, and that "Cantriglie" also originate from Italy.

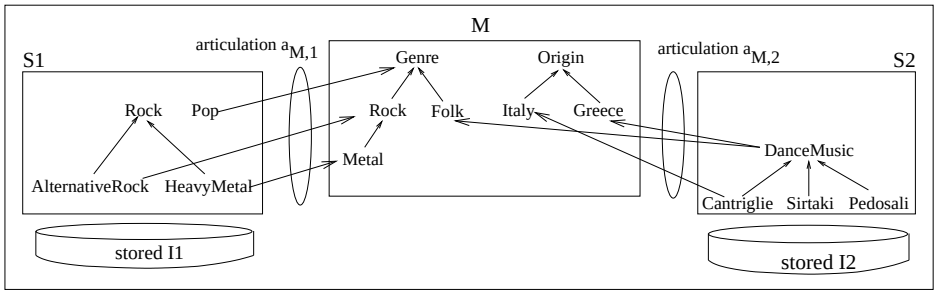

Fig. 2. A mediator over two music sources

Thus a network comprised of simple sources, mediators and articulated sources can be defined in terms of just articulated sources, as follows (two articulated sources are said to be disjoint if their terminologies are disjoint).

Definition 8. A *network of articulated sources,* or simply a *network*, N is a non-empty set of disjoint articulated sources $N = \{S_1, \ldots, S_n\}$, where each source S_i is articulated over some of the sources in $N \setminus \{S_i\}$.

Note that the disjointness of sources can be implemented in practice by, say, prefixing each term by the name of the source in which the term appears. Figure 3 shows an example of a network consisting of four sources $S_1, ..., S_4$; two simple sources (S_3 and S_4), one mediator (S_2) and one articulated source (S_1).

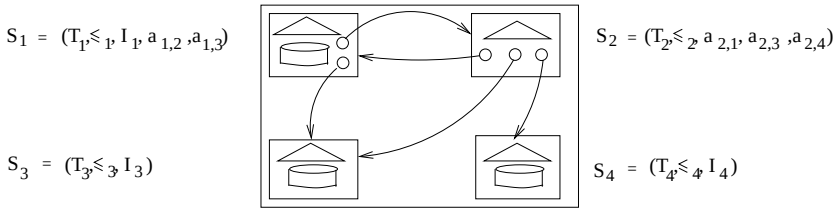

Fig. 3. A network of articulated sources

Networks are set up in order to let users extract information from the member sources. The following questions arise: (a) what queries should users be able to formulate, and (b) what answer should queries return? Next Section addresses these questions.

2.3 Network Queries and Answers

From a logical point of view, we can view an entire network $N = \{S_1, \ldots, S_n\}$ as a single simple source S_N, comprised by a terminology T, a subsumption relation $\sqsubseteq$ and a stored interpretation I, where:

- $T = \bigcup_{i=1}^{n} T_i$
- $I = \bigcup_{i=1}^{n} I_i$
- $\sqsubseteq = (\bigcup_{i=1}^{n} \sqsubseteq_i)^*$

where $\sqsubseteq_i$ is the *total subsumption* of the source S_i, given by the union of the subsumption relation $\preceq_i$ with all articulations of the source, that is: $\sqsubseteq_i = \preceq_i \cup a_{i,1} \ldots \cup a_{i,n}$ and A^* denotes the transitive closure of the binary relation A.

Accordingly, we can define a *network query* to be a query over T. By so doing, we have an answer to the first question: users of the network submit network queries. That is, users can formulate not only queries over the terminology of the articulated source nearest to them, but also queries over a remote terminology, or a combination of these two, freely mixing terms from different terminologies in a single query. In other words, as long as a term is *known* in the network[3], it can be used to extract information from anywhere in the network.

Following the model developed so far, the answer to a network query q, or *network answer,* is given by $ans_N(q)$, which relies, according to Definition 5, on the model of T generated by I, that is, of each term t in q :

$$\bar{I}(t) = \bigcup \{\ I(t') \mid t' \sqsubseteq t\} \tag{1}$$

2.4 Foundations for Answering Network Queries

The formulation of a network answer given in equation 1 does not immediately lend itself to computation, as it is expressed in terms of the network subsumption relation, which only exists in fragments. In order to derive a more operational definition of $\bar{I}(t)$, let us introduce entries.

Definition 9. Given an articulated network $N = \{S_1, \ldots, S_n\}$ and a term $t \in T$, the *entries of t in N*, $E(t)$, are the terms defined as follows:

$$E(t) = \{t\} \cup \ \{t' \in T \mid \exists t'' \in T :\ t' \preceq_{ij} t'' \sqsubseteq t \text{ for some } i, j\}.$$

The notation $t' \preceq_{ij} t'' \sqsubseteq t$ means that $t' \preceq_{ij} t''$ and $t'' \sqsubseteq t$. For each term t, the entries of t include t itself and those terms t' which articulate t in the various sources of the network, either directly or through an intermediate term t'' which is subsumed by t. For example, in the network shown in Figure 4.(a), the entries of the term c are given by: $E(c) = \{c, a3, b3, b1\}$ since terms *a3* and *b3* articulate c according to $\preceq_{31}$ and $\preceq_{32}$ respectively, and *b1* articulates *a3* (subsumed by c) according to $\preceq_{12}$.

In the following, if t is a term then we shall use $g(t)$ to denote the subscript of the terminology in which t belongs, e.g. if $t \in T_j$ then $g(t) = j$. As intuition

[3] This presupposes that each source knows (or can find) the source that owns every element of T, or that each term of each submitted query is accompanied by the identity (address) of its source.

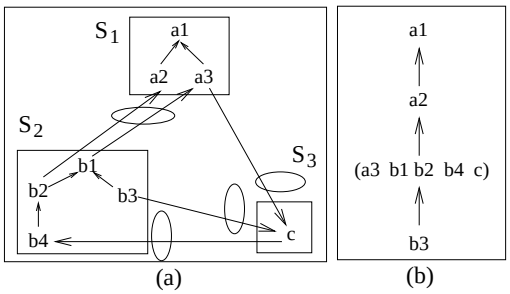

Fig. 4. A network of articulated sources

suggests and the following Proposition states, entries are crucial to compute network answers.

Proposition 2. For each term t of a network N, $ans_N(t) = \bigcup\{\ ans_{g(t')} \mid t' \in E(t)\}$.

This Proposition represents a first step towards a method for rewriting network queries, in that it maps a network query onto queries to the articulated sources of the network. However, in order to compute entries, knowledge of all articulations and taxonomies is required (*i.e.* $\sqsubseteq$ and $\preceq_{ij}$), and this is feasible only in a client-server architecture, or in a hybrid P2P distributed system (e.g. see [23]), where it is possible to assume that there exists a sort of server node holding this knowledge.

In order to devise a method for a pure P2P architecture, we refine entries to local entries. Our aim is to define equations which can be derived in a single articulated source S_i, by relying on the knowledge of the source subsumption relation $\preceq_i$ and articulations $\preceq_{ij}$.

Definition 10. Given a source S_i in an articulated network N and a term $t \in T_i$, the *local entries of* t *in* N, $e(t)$, are the terms defined as follows:

$$e(t) = \{t\} \cup \bigcup_{i=1}^{n} \{e(t') \mid t' \sqsubseteq_i t,\ t' \notin T_i\}.$$

By means of recursion, local entries break down the definition of entries into pieces each of which can be computed locally, *i.e.* at a source of the network. We can add subscripts and write $e_i(t) = \{t\} \cup \bigcup_{i=1}^{n} \{e_{g(t')}(t') \mid t' \sqsubseteq_i t,\ t' \notin T_i\}$ in order to indicate the source that performs the computation. To illustrate, let us consider the local entries of term c in the network shown in Figure 4.(a):

$$e(c) = \{c\} \cup e(b3) \cup e(a3) \tag{2}$$
$$e(b3) = \{b3\} \tag{3}$$
$$e(a3) = \{a3\} \cup e(b1) \tag{4}$$
$$e(b1) = \{b1\} \cup e(c)$$

Note that each such equations can be derived in a single articulated source S_i, by relying on the knowledge of the source subsumption relation $\preceq_i$ and articulations

$\preceq_{ij}$. Cycles in the network subsumption spanning at least one articulation are reflected into the definition of local entries. As an illustration, a few substitutions in equation 2 yield:

$$e(c) = \{c, b3, a3, b1\} \cup e(c) \tag{5}$$

reflecting the cycle $(a3, c, b4, b2, b1, a3)$ in the subsumption relation graph. This cycle would cause a similar effect on $e(a3)$ and $e(b1)$. Following the classical approach, $e(c)$ in equation 5 can be seen as a function F, mapping sets of terms into sets of terms, defined as: $F(X) \stackrel{\text{def}}{=} \lambda X.\{c, b3, a3, b1\} \cup X$. The fixpoints of this function are the solutions of the corresponding equation. F is clearly monotonic in the complete partial order $(2^T, \subseteq)$, thus it has unique least fixpoint, *i.e.* $\{c, b3, a3, b1\}$, and a unique greatest fixpoint, *i.e.* T. In addition, any set in between these two, according to the $\subseteq$ relation, is also a fixpoint. The selection of a particular fixpoint as the preferred one, is typically based on criteria reflecting the setting in which these equations arise. In our case, we want local entries to be the same as entries, *i.e.* for all t, $e(t)$ must be the same as $E(t)$. This leads to the selection of the least fixpoint as the preferred solution. In the following, we will generalize these ideas and provide an algorithm for computing local entries.

For convenience, let us define, for a term $t_i \in T_i$, the *index* of t_i as follows:

$$H(t_i) = \{t' \in T \mid t' \sqsubseteq_i t_i,\ t' \notin T_i\}.$$

As a consequence, the definition of the local entries of a term t_i consists, for each term $t_k \in E(t_i)$, of one equation:

$$e(t_k) = \{t_k\} \cup \bigcup \{e(t') \mid t' \in H(t_k)\}, \tag{6}$$

Below we give the index of each term involved in the definition of c in the last Example: $H(c) = \{b3, a3\}$, $H(b3) = \emptyset$, $H(a3) = \{b1\}$, $H(b1) = \{c\}$.

Let us call the *simple unfolding* of one such equations the transformation obtained by adding (in the set-theoretic sense) the definition of each occurring local entry, as given by the appropriate equation, to the right-hand side. For example, the simple unfolding of $e(c) = \{c\} \cup e(b3) \cup e(a3)$, is the addition of the definition of $e(b3)$ and of $e(a3)$ (as given by equations 3 and 4, respectively) to the right-hand side, and yields the equation: $e(c) = \{c, b3, a3\} \cup e(b3) \cup e(a3) \cup e(b1)$. The simple unfolding produces an equation equivalent to the original one, since the union operator is idempotent, that is $A = A \cup A$. In general, an equation of the form (6) can be written as:

$$e(t_k) = A_k \cup \bigcup \{e(t') \mid t' \in B_k\} \tag{7}$$

where A_k and B_k are suitable sets of terms. The simple unfolding of an equation in this form is given by:

$$e(t_k) = (A_k \cup B_k) \cup \bigcup \{e(t') \mid t' \in B_k \cup \bigcup_{t'' \in B_k} H(t'')\}.$$

Now let us call the *unfolding* of an equation, the equation generated by the iterative application of simple unfolding until it produces no change. Formally,

the generation of the unfolding can be represented by the evolution of the sets A_k and B_k. Using a superscript to indicate the number of the iteration step in the unfolding, we can derive the initial sets from the equations 6:

$$A_k^0 = \{t_k\}$$
$$B_k^0 = H(t_k)$$

while the (i+1)-th sets, for $i \geq 0$, are obtained by the simple unfolding of the $i-$th sets, and according to 8 are given by:

$$A_k^{i+1} = A_k^i \cup B_k^i$$
$$B_k^{i+1} = B_k^i \cup \bigcup \{H(t) \mid t \in B_k^i\}. \quad (8)$$

Existence and uniqueness of unfoldings is very simple to prove: at each iteration a few constants and a few recursions are added; since there are only finite terms in $E(t_i)$, the process is bound to converge after at most $|E(t_i)|$ simple unfoldings. Then, let us call $B_k^\star$ the set B_k^m, where m is the smallest number such that: $B_k^m = B_k^{m+1}$. By a simple induction argument, it can be shown that, for all terms t_k and $i \geq 0 : A_k^{i+1} = \{t_k\} \cup B_k^i$. It follows that $A_k^m \neq A_k^{m+1} = A_k^{m+2} = A_k^{m+3} = \ldots$, hence analogously to $B_k^\star$, we define $A_k^\star = A_k^{m+1} = \{t_k\} \cup B_k^\star$. It can be shown that:

Proposition 3. For each term t of a network N holds: $E(t) = LFP(e(t)) = A^\star$ where $LFP(F)$ denotes the least fixpoint of F on $(2^T, \subseteq)$.

Table 1 shows the computation of $A_k^\star$ for the term $t_k = c$ in the last Example. It can be verified that $m = 2$, thus $B_k^2 = B_k^\star$ and $A_k^3 = A_k^\star$.

i	A_k^i	B_k^i
0	$\{c\}$	$\{b3, a3\}$
1	$\{c, b3, a3\}$	$\{b3, a3, b1\}$
2	$\{c, b3, a3, b1\}$	$\{b3, a3, b1, c\}$
3	$\{c, b3, a3, b1\}$	$\{b3, a3, b1, c\}$

Table 1. Computing $A_k^\star$

The computation of local entries is an iterative process that mimics the unfolding of the equations 6. At the $i-$th iteration, the set B^i is computed, until a set B^m is found which is stable and thus is the sought $B^\star$. From $B^\star$, the set $A^\star$, hence $E(t)$, is obtained by just adding the term $\{t\}$. As equation 8 shows, B^i is obtained by adding the index $H(t)$ to B^{i-1}, for each term t in B^{i-1}.

3 Query Evaluation Approaches

We can distinguish query evaluation approaches according to two orthogonal criteria: (a) the number of *stages* of the query evaluation process, and (b) the number of *sources* that control this process.
According to the first criterion we can distinguish the following two approaches:

– The *Rewriting* (or double-stage) approach (denoted by *R*).
  Here, query evaluation is performed in two stages: in the first, all entries are collected, while in the second the collected entries are sent (probably after a planning stage) to the sources.
– The *Direct* approach (denoted by *D*).
  Here, both entries and local answers are collected in one stage.

According to the second criterion, assuming that each source can provide the index $H(t)$ for each term t of its taxonomy, we can distinguish two ways of computing B^i : either by sending indexes to a single place where the set B^i is accumulated, or by sending the partial B^i to the sources where indexes are being maintained. Correspondingly, there are two main approaches to the computation of $A^\star$:

– The *single controller* (or centralized) approach (denoted by *1*).
  Here, all entries and local answers are accumulated to one source (the original source).
– The *multiple controller* (or decentralized)approach (denoted by *m*).
  Here, the collection of entries and local answers is done collaboratively by several sources.

Clearly, the above cases can be combined resulting in four different query evaluation approaches, namely *R1, Rm, D1* and *Dm*. Each approach is discussed in a subsequent section. Of course, all these approaches are subject to several kinds of optimizations which, for reasons of space, are just sketched, in section 3.5.

3.1 The Rewriting - Single Controller (*R1*) Approach

In *R1* approach, any node of the network receiving a network query q must perform the following steps:

(Stage A)
  1. parsing the query in order to identify the target terms, *i.e.* the terms occurring in q;
  2. executing a single controller algorithm for collecting the entries of each target term;

(Stage B)
  3. sending the entries as queries to be evaluated to the appropriate sources (probably after a planning phase) and derivation of the network interpretation of the target terms by taking the union of the received answers. Then, the network answer is obtained according to Def. 5.

Step 1 is trivial from an algorithmic point of view. Step 3 is also simple although a planning phase can take place for further optimization. Now the algorithm for Step 2, i.e. for collecting the entries of a term t is presented in Figure 5.

It is straightforward to prove that algorithm *Alg_R1* is correct as it implements in a straightforward manner the method described in Section 2.4.

Algorithm: *Alg_R1*
Input: a term t
Output: the set $E(t)$
begin
(1) $A := \{t\}$; $C := \{t\}$;
(2) **repeat**
(3) change:=FALSE ;
(4) $B := \emptyset$;
(5) **for each** $t' \in C$ **do** $B := B \cup H_{g(t')}(t')$;
(6) **If** $B \not\subseteq A$ **then begin**
(7) $C := B \setminus A$;
(8) $A := A \cup B$;
(9) change := TRUE
(10) **end**
(11) **until** change = FALSE ;
(12) **return**(A);
end

Fig. 5. Computing local entries in the *R1* approach

3.2 The Rewriting - Multiple Controller (*Rm*) Approach

Query evaluation in the *Rm* approach can be done as in the *R1* approach with the only difference that here Step 2, i.e. the collection of entries, is done collaboratively by several sources. Specifically, a special data structure D is employed that keeps track of the state of this computation and that is being exchanged by the nodes involved in the process. In particular, the collection of entries is accomplished in the following steps:

1. construction of the data structure D for query evaluation;
2. insertion of D into the P2P network for evaluation;
3. waiting for the result;

For computing the network interpretation of a term t_i the data structure employed by this method is a triple whose first member is the name of the source S_o where the query is posed, followed by two sets of terms C and A explained below. To insert D into the P2P network, means to send this data structure to the source which owns the term t_i. Upon receiving a data structure D, an articulated source S_i executes the Algorithm *Alg_Rm*, illustrated in Figure 6. The set C is the set of the terms still to be processed, while A accumulates $A^\star$. Initially, the only term to be processed is t_i itself, while A is set to A^0. Upon processing a data structure D, the algorithm focuses on the terms to be processed (*i.e* in C) that are also in the local source terminology T_i. Each such term t is added to A and removed from C. In addition, the algorithm adds to C the terms in $H_i(t)$ *which have not yet been processed.* This last provision prevents redundant computations. Furthermore, it avoids non-termination in presence of cycles in the global subsumption relation. Otherwise, this kind of cycles would cause a phenomenon analogous to that of deadlocks in distributed databases and systems [17, 12]. The Algorithm may exit in two different ways: if there are no more

terms to be processed (*i.e* C is empty), the data structure D is returned (via *send*) to the originating source S_o, which will find the entries of t_i as the third element of the corresponding triple D. If there are terms still to be processed, the data structure is passed on to a source that owns at least one term in C.

Algorithm: *Alg_Rm* (for a source S_i)
Input: a data structure $D = (S_o, C, A)$.
Output: an updated data structure sent to another source
begin
(1) **for each** $t \in C \cap T_i$ **do begin**
(2) $A := A \cup \{t\}$;
(3) $C := (C \setminus \{t\}) \cup (H_i(t) \setminus A)$;
(4) **end**
(5) **if** $(C = \emptyset)$ **then** *send(S_o, C, A)*;
(6) **else begin**
(7) select a S_j s.t. $\exists\, t' \in C$ where $g(t') = j$;
(8) *send(S_j, C, A)*;
(9) **end**
end

Fig. 6. The algorithm for the *Rm* approach

Also in this case, it is not hard to show that: algorithm *Alg_Rm* is correct.

Table 2 shows the application of the *Alg_Rm* algorithm to a query containing just the term c. The first column shows the source at which the algorithm is applied; the second column the resulting triple of the term c. The first line results from the first stage executed at the source S_o where the query is posed.

Source	(t, C, A)
S_o	$(c, \{c\}, \{\})$
S_3	$(c, \{b3, a3\}, \{c\})$
S_2	$(c, \{a3\}, \{c, b3\})$
S_1	$(c, \{b1\}, \{c, b3, a3\})$
S_2	$(c, \{\}, \{c, b3, a3, b1\})$

Table 2. An application of *Alg_Rm*

An important remark that we have to mention here is that the single controller approach is based on the assumption that each source (specifically S_o) knows $g(t)$ for each term $t \in T$. Notice that this is not aligned with the pure P2P paradigm, as this presupposes the existence of a registration server, known to all sources of the network, where all terms are registered to. Notice that the multiple controller approach is not based on this assumption, as each source has to know $g(t)$ only for each term t that appears in its articulation, something quite reasonable even for a pure P2P system.

From a complexity point of view, note that *Alg_R1* operates on a per-term basis, while *Alg_Rm* operates on a per-source basis. This means that in the worst case, the number of messages that have to be exchanged for computing the entries $E(t)$ of a term $t \in T$ is $2|E(t)| \leq 2|T|$ in $R1$, and k, where k is the number of sources, in Rm.

3.3 The Direct (*D1* and *Dm*) Approaches

The algorithms for the direct query evaluation approaches are direct extensions of the techniques presented for query rewriting. The only difference is that now the stored interpretation of each involved term is also collected. Hence, the resulting algorithms return the set $\bar{I}(t)$.

Specifically, an algorithm for the *D1* approach can be obtained by modifying *Alg_R1* as follows:

- add the statement $R := R \cup \bar{I}_{g(t')}(t')$ after the line (8), and
- replace the statement of line (12) with the statement **return**(A).

Analogously, we can obtain an algorithm for the *Dm* approach by modifying *Alg_Rm* as follows:

- add the statement $R := R \cup \bar{I}_{g(t')}(t')$ after the line (3), and
- add the statement $Send(So, R)$ after the line 4.

Another algorithm that implements a direct and multiple controller approach is the one presented in [20], which is presented in Figure 7 using the notations employed in the current paper. Notice that cycles in the network subsumption relation may cause endless query loops. According to [20], this phenomenon can be avoided if each source maintains a log file of the received queries . An alternative way, is to add to the parameters of the algorithm a data structure that keeps track of the answered terms (i.e. a data structure like D).

Algorithm: $Alg\_Dm^{\beta}\_i$ (for a source S_i)
Input: a term t
Output: the set $\bar{I}(t)$
begin
(1) $R_i := \bar{I}_i(t_i)$;
(2) **for each** $t' \in H_i(t_i)$ **do**
(3) $R_i := R_i \cup Alg\_Dm^{\beta}_{g(t')}(t')$;
(4) **return**(R_i);
end

Fig. 7. An alternative algorithm for the *Dm* approach

3.4 Comparative Evaluation

The essential difference between the query rewriting and the direct evaluation approach consists in the fact that the former approach is based on the idea of planning the access to local sources, while the latter relies on a simple-minded,

straightforward style. None of these two approaches stands out as the best in all cases. Typically, on queries that can be answered with a few accesses to local sources, or producing a small result, planning does not provide any specific benefit, and may indeed result in a slower evaluation procedure. On the other hand, the evaluation of queries requiring a significant number of accesses to local sources, or producing a large result, may indeed greatly benefit from rewriting, for the reason illustrated next. As the application of the *Alg_Rm* algorithm illustrated in Table 2 shows, it may happen that a source must be accessed more than once during the processing of a query. This is due to the structure of the global subsumption relation, and would occur even if the algorithm *Alg_R1* were used. In this case, rewriting can result in a more efficient query evaluation strategy, because sources need to be accessed more than once only in the first stage, when indexes of terms are retrieved. Once the coordinating node knows the local entries of the target terms, it can plan the second stage so that each source is accessed only once for retrieving the data. Instead, in the direct evaluation approach, this optimization is not possible, and the same source may be accessed more than once for retrieving different data. Since indexes are typically much smaller and much faster to obtain than data, multiple accesses to the same source has a less negative impact in the rewriting approach. Another remark is that the multiple controller approach is less robust than the single controller one, because if the node that holds the data structure D disconnects (and note that in P2P systems peers come and go), then the network query will not be answered. Instead, in the single controller approach a query is not answered only if the controller, i.e. the original source, disconnects but in that case there is no need to compute any answer.

Table 3 summarizes the above discussion. In brief, in the rewriting approach more messages have to be exchanged (as there are 2 stages). However, if the size of the local answers is big, i.e. if local answers consist of big number of objects, or if the size of objects is big (e.g. audio/video files), then the rewriting approach is more preferred as planning can be employed in order to reduce the network throughput.

	Query Evaluation Approaches			
	R1	*Rm*	*D1*	*Dm*
number of messages	more more	more	less	less less
size of messages	small	small	big	big
planning	possible	possible	impossible	impossible
robustness	more	less	more	less
applicability	hybrid P2P	pure P2P	hybrid P2P	pure P2P
best if local answers are	big	big	small	small

Table 3. Comparison of query evaluation approaches

3.5 Optimization Issues

Below we discuss in brief a number of techniques for making the evaluation of queries more efficient.

– Instead of collecting the entries (and local answers) of one term at a time, we can collect the entries (and local answers) of *all* terms that appear in q. For doing so, we have to modify *Alg_R1* (and *Alg_D1*) so that to take as input a set of terms, and *Alg_Rm* (and *Alg_Dm*) so that the data structure D to be a set of triples. In this way we can reduce the number of messages (either local queries or local answers) that have to be exchanged.
– We can exploit parallelism, and thus reduce the latency time of the system, by sending in parallel all *Send* statements (e.g. the calls of line (5) of *Alg_R1*).
– We can increase the robustness of the multiple controller approaches by adopting more than one control structures D, or by adopting more sophisticated mechanisms like the those employed by distributed databases.

4 Related Work

Semantic-based retrieval in P2P systems is a great challenge. In general, the language that can be used for indexing the objects of the domain and for formulating semantic-based queries, can be *free* (e.g natural language) or *controlled*, i.e. object descriptions and queries may have to conform to a specific vocabulary and syntax. The first case, resembles distributed Information Retrieval (IR) systems and this approach is applicable in the case where the objects of the domain have a textual content (e.g. [14]). In this paper we have focused on the second case where the objects of a peer are indexed according to a specific conceptual model represented in a data model (e.g. relational, object-oriented, logic-based, etc), and content searches are formulated using a specific query language. This approach, which can be called "database approach", starts to receive noteworthy attention by the researchers, as is believed that the database and knowledge base research has much to contribute to the P2P grand challenge through its wealth of techniques for sophisticated semantics-based data models and query processing techniques (e.g. see [9, 6, 11]). Of course, a P2P system might impose a single conceptual model on all participants to enforce uniform, global access, but this will be too restrictive. Alternatively, a limited number of conceptual models may be allowed, so that traditional information mediation and integration techniques will likely apply (with the restriction that there is no central authority), e.g. see [16, 15]. The case of fully heterogeneous conceptual models makes uniform global access extremely challenging and this is the case that we are interested in.

From a data modeling point of view several approaches for P2P systems have been proposed recently, including relational-based approaches [6], XML-based approaches [10] and RDF-based [15]. In this paper we consider a taxonomy-based conceptual modeling approach. This approach has three main advantages (for more see [20]): (a) it is very easy to create the conceptual model of a source, (b) the integration of information from multiple sources can be done easily, and (c) automatic articulation using data-driven methods (like the one presented in [19]) are possible.

From an architectural point of view, and according to the SIL (Search Index Link) model presented in [8], our networks falls into the case of P2P sys-

tems which have only forwarding search links. Specifically, our work specializes content-based queries to taxonomy-based queries. Another distinguishing characteristic, is that in our model a peer does not just forward the received queries to its neighbors, it first translates them. Also note that the relationships stored in the articulations not only determine query translation but also query propagation. Of course, work done on P2P architectures, e.g. [23, 8], could be also exploited in our setting in order to enhance the efficiency of a taxonomy-based P2P system. Our approach has some similiraties with Edutella [16, 15], an RDF-based metadata infrastructure for P2P systems. However, the mediators of Edutella distribute a query to a peer only if the query can be answered completely by the peer. In contrast, in our model the answers of queries are formed collaboratively. Moreover, in Edutella special servers are devoted for registering the schema that each peer supports. In our model we do not make any such assumption.

An approach for supporting object queries appropriate for domains where no accepted naming standards exist (and thus it generalizes the functionality provided by systems like Napster and Gnutella) is described in [11]. The mapping tables employed there can express only exact mappings, however the open/closed-world semantics that are given are quite interesting and their application to our setting is one topic of our research agenda.

5 Concluding Remarks

In this paper, we focused on query evaluation in P2P systems that support semantic-based retrieval services. We gave four algorithms to carry out this task according to two different approaches (rewriting *vs.* direct evaluation) and computation styles (centralized *vs.* decentralized). Although we adopted a conceptual modeling approach that is based on taxonomies, much of the results presented of this paper (i.e. the query evaluation approaches presented) can be adapted to cases where other conceptual modeling approaches are employed.

For reason of space, we have not considered in depth optimization issues, limiting ourselves to lay down the basics of the four methods. Issues for further research include also query evaluation in cases we have articulations which relate terms with queries.

References

1. "About LEGION - The Grid OS"(www.appliedmeta.com/legion/about.html), 2000.
2. "How Entropia Works"(www.entropia.com/how.asp), 2000.
3. "Groove" (www.groove.net), 2001.
4. "Napster"(www.naptster.com), 2001.
5. T.E. Anderson, M. Dahlin, J. M. Neefe, D. A. Patterson, D. S. Roselli, and R. Wang. "Serveless Network File Systems". *SOSP*, 29(5), 1995.
6. P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. "Data Management for Peer-to-Peer Computing: A Vision". In *Proceedings of WebDB02*, Madison, Wisconsin, June 2002.

7. W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. "Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs". In *Proceedings of Measurement and Modeling of Computer Systems*, June 2000.
8. B. Cooper and H. Garcia-Molina. "Modeling and Measuring Scalable Peer-to-peer Search Networks". Technical report, University of Stanford, September 2002.
9. S. Gribble, A. Halevy, Z. Ives, M. Rodrig, and D. Suiu. "What can Databases do for Peer-to-Peer?". In *Proceedings of WebDB01*, Santa Barbara, CA, 2001.
10. Alon Halevy, Zachary Ives, Peter Mork, and Igor Tatarinov. "Piazza: Data Management Infrastructure for Semantic Web Applications". In *Proceedings of WWW'2003*, May 2003.
11. A. Kementsietsidis, M. Arenas, and R. J. Miller. "Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues". In *Int. Conf. on Management of Data, SIGMOD'2003*, San Diego, California, June 2003.
12. Edgar Knapp. "Deadlock Detection in Distributed Databases". *ACM Computing Surveys*, 19(4), 1987.
13. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. "Oceanstore: An Architecture for Global-Scale Persistent Storage". In *ASPLOS*, November 2000.
14. Bo Ling, Zhiguo Lu, Wee Siong Ng, BengChin Ooi, Kian-Lee Tan, and Aoying Zhou. "A Content-Based Resource Location Mechanism in PeerIS". In *Procs of Int. Conf. on Web Information Systems Engineering, WISE 2002*, Singapore, Dec 2002.
15. W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch. "EDUTELLA: A P2P networking infrastructure based on RDF". In *WWW'2002*, 2002.
16. W. Nejdl, B. Wolf, S. Staab, and J. Tane. "EDUTELLA: Searching and Annotating Resources within an RDF-based P2P Network". In *Semantic Web Workshop 2002*, Honolulu, Havaii, May 2002.
17. Chia-Shiang Shih and John A. Stankovic. "Survey of Deadlock Detection in Distributed Concurrent Programming Environments and its Application to Real-Time Systems and Ada". Technical Report UM-CS-1991-043, University of Massachusetts, 1991.
18. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications". In *Procs of the 2001 ACM SIGCOMM Conference*, 2001.
19. Y. Tzitzikas and C. Meghini. "Ostensive Automatic Schema Mapping for Taxonomy-based Peer-to-Peer Systems". In *Int. Workshop on Cooperative Information Agents, CIA-2003*, Helsinki, Finland, August 2003.
20. Y. Tzitzikas, C. Meghini, and N. Spyratos. "Taxonomy-based Conceptual Modeling for Peer-to-Peer Networks". In *Procs of 22th Int. Conf. on Conceptual Modeling, ER'2003*, Chicago, Illinois, October 2003.
21. Y. Tzitzikas, N. Spyratos, and P. Constantopoulos. "Mediators over Ontology-based Information Sources". In *Second International Conference on Web Information Systems Engineering, WISE 2001*, Kyoto, Japan, December 2001.
22. G. Wiederhold. "Mediators in the Architecture of Future Information Systems". *IEEE Computer*, 25:38–49, 1992.
23. Beverly Yang and Hector Garcia-Molina. "Comparing Hybrid Peer-to-Peer Systems". In *The VLDB Journal*, pages 561–570, sep 2001.