

A Distributed Case-Based Query Rewriting

Maurizio Panti Luca Spalazzi Loris Penserini

Istituto di Informatica, University of Ancona, via Brecce Bianche, 60131 Ancona, Italy
{panti, spalazzi, pense}@inform.unian.it

Abstract

In literature, the mediator architecture has been proposed for taking information from distributed, heterogeneous, and often dynamic sources and making them work together as a whole. In this paper we propose a distributed case-based approach for the main problem of a mediator, i.e. rewriting queries according to mediator's schema. According to this approach we use a case memory as mediator's schema. Therefore, such a schema is not static (as in other systems) but is dynamically updated through the cooperation with information sources and other mediators, strongly influenced by the queries submitted by a consumer. From the analysis of different cooperation strategies arises that it is more efficient and effective for a mediator to directly cooperate with information sources, when the sources are few. Otherwise, it is more efficient to cooperate with other mediators.

1 Introduction

Nowadays, information systems can be thought as collections of *information sources* and *information consumers* that are often *distributed* in world-wide networks. This means that sources and consumers can be autonomous and thus are often *heterogeneous* and *dynamic*. Information sources are heterogeneous due to discrepancies at the physical level (different DBMSs, software and hardware platforms), logical level (different data models), and conceptual level (different schemas, concept and relation names).

Example 1.1 Figure 1 shows an example of three information sources about articles on computer science and their authors. Let us suppose that the three sources receive the query $Q_0 \doteq \forall pub.ai. w_3$ can answer to Q_0 (it finds a view that matches with Q_0), whereas w_1 and w_2 can not. Indeed, the schema of w_1 is different (*heterogeneity of schemas*). Notice that, if we rewrite the query as $Q'_0 \doteq \forall pub.\forall keyword.\{“AI”\}$, w_1 is able to answer since $\forall pub.\forall keyword.\{“AI”\}$ is part of its schemas. For what concerns w_3 , it is not a matter of heterogeneity, w_3 simply does not have the required information.

Information consumers are heterogeneous as well, since each consumer may have different customization needs due to distinct business objectives. Moreover autonomous information sources are dynamic, since they may be added to the system, may become (temporarily or definitively) unavailable, or, sometimes, may also vary their conceptual schemas. Information consumers are also dynamic. Indeed new consumers can be inserted or removed. Furthermore, their information needs can change very often. Our work deals with information integration in distributed, heterogeneous, and dynamic information systems, i.e., constructing answers to query from consumers. Our primary goal consists on rewriting a consumer's query into queries to specific information sources. In [M. Panti, L. Spalazzi, A. Giretti, 2000] we proposed a distributed case-based approach to the problem of rewriting queries. According to this approach we have a dynamic mediated schema (the case memory) instead of a static one (as in other systems). Indeed, our mediator's schema is dynamically updated through the cooperation with information sources and other mediators, strongly influenced by the queries submitted by a consumer. In this paper we focus on the cooperation strategies of the distributed case-based reasoner. Indeed, we compare different strategies from a theoretical point of view and discuss the results. The paper is organized as follows. Section 2 reports the related work. An overview of our work is provided in Section 3. Section 4 summarizes results about local query rewriting (i.e. when a query can be reformulated by a single mediator). Section 5 describes the distributed query rewriting (i.e., when a query can be reformulated only through cooperation among mediators) and discuss some theoretical results. Finally, some conclusions are given in Section 6.

2 Related Work

Intelligent Information Integration (I^3). The most significant approach to information integration relies on the so called *mediator architecture* [Wiederhold, 1992]: an architecture with three-layers. A layer is devoted to information consumers, another layer is devoted to information sources, the middle layer (i.e. the mediator) deals with the reformulation of a given query in a set of queries, each targeted at a selected source. Intuitively, such a reformulation would be equivalent to the original query (i.e., denote the same set of instances). Nevertheless, often this is not possible. Therefore, a query

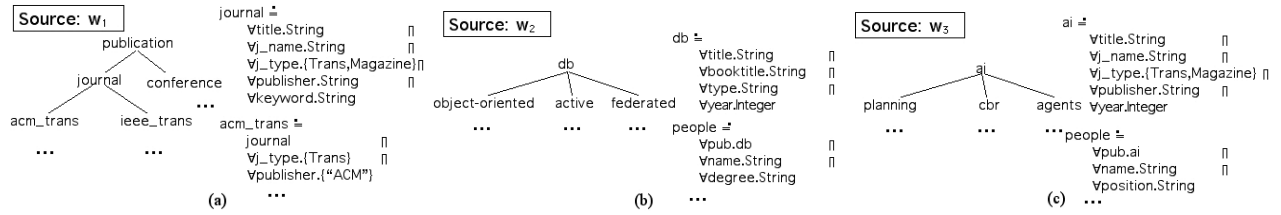


Figure 1: An example of information sources.

reformulation *contained in* (instead of *equivalent to*) the original query is considered adequate (see for example [Ullman, 1997; Beeri *et al.*, 1997]). Furthermore, when a *description logic* (e.g. [Borgida and Patel-Schneider, 1994]) is used as data modeling and query language (e.g. [Beeri *et al.*, 1997; Kirk *et al.*, 1995; Ullman, 1997]), the problem of query containment corresponds to the subsumption problem.

There are two basic approaches to intelligent information integration: the procedural approach and the declarative approach. In the *procedural* approach (e.g., TSIMMIS [Garcia-Molina *et al.*, 1997]), mediators integrate information from sources through ad-hoc procedures defined with respect to a set of predefined information needs. When such needs or sources change (i.e., we have a dynamic information system), a new mediator must be generated. In the *declarative* approach (e.g., SIMS [Arens *et al.*, 1993], Information Manifold [Kirk *et al.*, 1995], and Infomaster [Duschka and Gensereth, 1997]), mediators use suitable mechanisms to rewrite queries according to source descriptions. Usually, the proposed systems do not allow automatic adaptation of such descriptions when something changes.

Distributed Case-Based Reasoning (DCBR). A *distributed case-based reasoner* [Prasad *et al.*, 1996; Plaza *et al.*, 1997] learns by experience and cooperation how problems can be solved. Therefore, it is appropriate for distributed and dynamic application domains, when it is impossible to have predefined solutions. Nevertheless, as far as we know, its application to information integration is a novelty. In DCBR, there are several agents; each agent has its own case memory since it could have acquired its own independent problem-solving experiences. A new problem is solved through agent cooperation. Indeed each agent reuses the local past case that best contributes to the overall case. Description logic is applied to case-based reasoning (and thus DCBR) as well (e.g. [Koehler, 1994]). Indeed, in case-based reasoning, subsumption becomes a powerful tool to automatically derive case hierarchies that can be used in case retrieval and case retention.

3 Work Overview

Our goal is to build an information system capable of interconnecting information consumers and sources. Previous approaches solve several problems concerned with distributed, heterogeneous information systems. Their main limitation is related to the capability of evolving according to dynamic information systems. We propose (see also [M. Panti, L. Spalazzi, A. Giretti, 2000]) a multi-agent system which is based on a mediator architecture, i.e. mediators, sources, and

consumers are agents. All the agents adopt a description logic called C-CLASSIC [Borgida and Patel-Schneider, 1994] as data modelling and query language. Indeed, subsumption in C-CLASSIC can be computed in polynomial time.

In our work, the capability of a **mediator** agent to face heterogeneous and dynamic systems relies on a thesaurus and a distributed case-based reasoner. Each mediator has its own *thesaurus* in order to solve name heterogeneity. A thesaurus is composed by a set of classes of synonyms. Each element of a class has the reference to information sources that use it as term. Every time a new query arrives, its terms are translated in standard terms by means of the thesaurus. The reverse process occurs when the rewritten query must be sent to information sources. The thesaurus must be dynamically updated when a change in the system occurs. The classes of the thesaurus are modified by means of clustering. The explanation of this technique is out of the scope of the paper, for more details we remind to [Diamantini and Panti, 1999]. Therefore, in the rest of the paper we assume that *no name heterogeneity occurs* (i.e., the thesaurus has a maximum precision). Each mediator has also its own *case-based reasoner* in order to solve heterogeneity of schemas and consumer's needs. Each case contains a query and the corresponding reformulation, and it is stored in the mediator's case memory. When a new query from the consumer arrives, the mediator looks for past queries subsumed by the new one and adapts the corresponding solutions in order to obtain a reformulation of the new query. This means that the mediated schema is strongly influenced by the queries submitted by consumers (mediator's experience). When this experience does not help, the mediator cooperates with other mediators and/or information sources. This approach allows the mediator to update its schemas and therefore to take into account changes in information sources and consumer's needs. This can be considered a kind of distributed case-based reasoning.

In our system, **information source** agents¹ are able to cooperate with mediators through a query reformulation based on local schemas. Indeed each source is able to find a view over its schema that is subsumed by an input query.

Example 3.1 Let us consider the distributed information system of Figure 1 (Example 1.1). According to the above assumption, we consider that all the (concept and role) names have been already translated by a thesaurus and thus no name heterogeneity occurs. Nevertheless, conceptual schemas are different (as showed in Example 1.1).

¹Notice that when a source is a non-agent software, we need a wrapper as interface between the source and the rest of the system.

Query	Rewriting of the query	Information Sources
H	$(\forall pub.journal \sqcap \forall pub.db), (\forall pub.ai \sqcap \forall pub.db)$	$(\forall pub.journal, w_1), (\forall pub.db, w_2), (\forall pub.ai, w_3)$
I	$\forall pub.\forall publisher.\{ "ACM" \}$	$(\forall pub.\forall publisher.\{ "ACM" \}, w_1), (\forall pub.\forall publisher.\{ "ACM" \}, w_3)$
J	$(\forall pub.\forall keyword\{ "AI" \}), (\forall pub.ai)$	$(\forall pub.\forall keyword\{ "AI" \}, w_1), (\forall pub.ai, w_3)$
K	$(\forall pub.\forall keyword.\{ "Agents" \} \sqcap \forall pub.db),$ $(\forall pub.agents \sqcap \forall pub.db)$	$(\forall pub.\forall keyword.\{ "Agents" \}, w_1), (\forall pub.db, w_2), (\forall pub.agents, w_3)$

Figure 2: An example of case memory.

Let us suppose that source w_1 receives a query $Q_1 \doteq \forall j.type.\{Trans\}$. w_1 finds that Q_1 subsumes *acm.trans* and *ieee.trans* and thus returns the corresponding instances (denoted by $I_{w_1}(acm.trans) \cup I_{w_1}(ieee.trans)$).

4 Local Query Rewriting

Generally speaking, a case is an arbitrary set of features (attribute-value pairs). Some features are devoted to represent a problem (in our application the query to be reformulated) in order to make easier the retrieval of a past problem similar to the current situation. The rest of features are devoted to represent the problem solution (the reformulated query and information sources where the reformulated query has been sent) to reuse it in the current problem. Furthermore, the problem of rewriting queries has a fundamental condition that must be satisfied: *the rewritten query must be contained in the original query* [Beeri et al., 1997; Ullman, 1997]. We use C-CLASSIC as a language for representing queries and thus cases. The subsumption relation is used as query containment and thus for case retrieval. According to above considerations, a **case** is a structure as the following: $\langle Q, Sol(Q), \langle (Q_1, w_1), \dots, (Q_n, w_n) \rangle \rangle$ where $w_1, \dots, w_n$ are information sources and $Q_1, \dots, Q_n$ are queries to $w_1, \dots, w_n$ respectively. Q is a given query and $Sol(Q)$ is its reformulation, i.e., a set of arbitrary combinations of $Q_1, \dots, Q_n$ such that each element of $Sol(Q)$ is subsumed by Q . A collection of cases is called **case memory**. A **terminology** (i.e., a set of concept definitions and their relations) is always associated to a case memory. $\top, \perp$ represent the *top* and *bottom* concepts, respectively. The initial case memory can be built as static mediated schemas of traditional mediators (e.g. SIMS [Arens et al., 1993]). In our system, the added value is that the DCBR works to maintain updated such a schema.

Example 4.1 Figure 2 and Figure 3 show the case memory and the terminology of a mediator of Example 3.1.

Query Retrieval. Every time a new query arrives, it is inserted in the terminology and classified by means of subsumption. If there exists a past problem equal to the new query, then its solution can be used as solution for the new query. Otherwise, the solutions of past problems that are subsumed by the new query can be used as solution for the new problem. The closest to the new query the retrieved cases are, the best the solutions are. This drives us to the notion of problems maximally contained in the new query. This notion allows the definition of the retrieval function $M_{inf}(Q, \mathcal{P})$ as a set of conjunctions of concepts $(\sqcap_{i \geq 1} X_i)$ of the given terminology $(\mathcal{P})$. For each conjunction of concepts there does not exist another conjunction of concepts of the same terminology $(\sqcap_{j \geq 1} Y_j)$ such that $(\sqcap_{i \geq 1} X_i) \sqsubseteq (\sqcap_{j \geq 1} Y_j) \sqsubseteq Q$.

Example 4.2 Let us suppose to have $Q_2 \doteq \forall pub.(ai \sqcap db)$ as input query in Example 4.1. Applying the retrieval function we obtain: $M_{inf}(Q_2, \mathcal{P}) = \{K, H \sqcap J\}$

Query Reuse. The basic reuse algorithm $(\mathcal{S}_{inf}(Q, \mathcal{P}))$ simply copies past solutions of past problems returned by the retrieval function.

Example 4.3 Applying the basic reuse algorithm to Example 4.2, we obtain: $\mathcal{S}_{inf}(Q_2, \mathcal{P}) = Sol(K) \cup (Sol(H) \sqcap Sol(J))^2$

Now we have all the building blocks to define the local reuse algorithm $(\mathcal{D}(Q, \mathcal{P}))$. Such an algorithm decomposes each consumer's query in basic components, applies the basic reuse algorithm to each component, and finally assembles the corresponding results (a more detailed description can be found in [M. Panti, L. Spalazzi, A. Giretti, 2000]).

Example 4.4 Let us suppose to have $Q_3 \doteq \forall pub.ai \sqcap \forall pub.acm$ as input query in Example 4.1. Applying the reuse algorithm we obtain:

$$\begin{aligned} \mathcal{D}(Q_3, \mathcal{P}) &= \mathcal{S}_{inf}(\forall pub.ai, \mathcal{P}) \sqcap \mathcal{S}_{inf}(\forall pub.acm, \mathcal{P}) \\ &= (Sol(K) \sqcap Sol(I)) \cup (Sol(H) \sqcap Sol(J) \sqcap Sol(I)) \end{aligned}$$

Notice that this is just one possible answer, another mediator with a different case memory may rewrite the query in a different way and thus return a different answer.

It is easy to prove that each element of $\mathcal{S}_{inf}(Q, \mathcal{P})$ and $\mathcal{D}(Q, \mathcal{P})$ is subsumed by Q . Therefore, a reformulation is contained in the original query as required in the definition of case. Finally, notice that in general $\mathcal{S}_{inf}(Q, \mathcal{P}) \neq \mathcal{D}(Q, \mathcal{P})$.

Query Evaluation. When the mediator has a reformulation of the received query, it must send such a reformulation to related information sources and wait for their answers.

Example 4.5 The evaluation of $Sol(H)$ consists on querying sources w_1, w_2 , and w_3 with $\forall pub.journal, \forall pub.db$, and $\forall pub.ai$ respectively, and integrating the related answers. The semantics of concepts helps us on integrating the answers. Let $I_{w_1}(\forall pub.journal), I_{w_2}(\forall pub.db)$, and $I_{w_3}(\forall pub.ai)$ be answers from w_1, w_2 , and w_3 respectively, then the answer to $\forall pub.(journal \sqcap db)$ is: $(I_{w_1}(\forall pub.journal) \cap I_{w_2}(\forall pub.db)) \cup (I_{w_3}(\forall pub.ai) \cap I_{w_2}(\forall pub.db))$.

Failures. We have a *local failure in query reuse* when a mediator is not able to rewrite a given query Q , i.e., $Sol(Q) = \perp^3$. This means that the mediator's case memory contains no past cases that can be used to reformulate Q . Usually this is due to the fact that it is the first time that the consumer formulates such a query, i.e., the consumer has a new information need.

² $Sol(H) \sqcap Sol(J)$ means that each element of $Sol(H)$ is conjuncted with each element of $Sol(J)$.

³ Notice that $I(\perp) = \emptyset$.

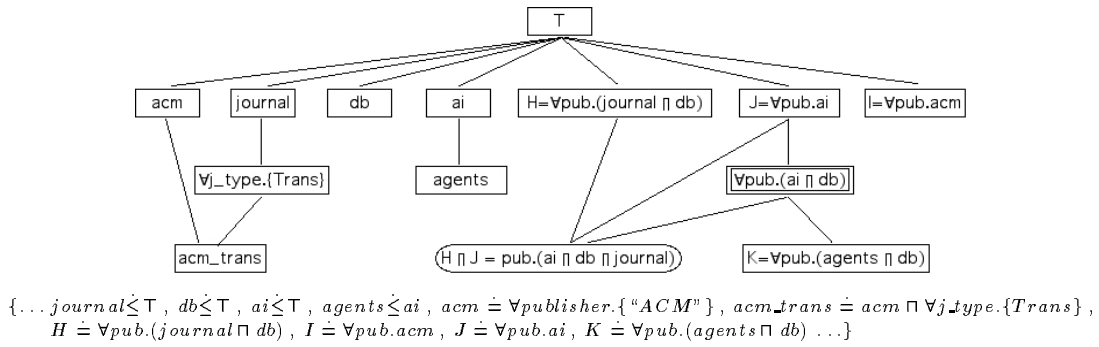


Figure 3: An example of terminology.

Example 4.6 Let us consider the case memory of Example 4.1 and the query $Q_4 \doteq Vpub.ai \cap Vaffiliation.\{Stanford\}$. The mediator is not able to rewrite the query. Indeed, we have

$$\mathcal{D}(Q_4, \mathcal{P}) = (Sol(K) \cap \perp) \cup (Sol(H) \cap Sol(J) \cap \perp) = \{\perp\}$$

Furthermore, we have a *local failure in query evaluation* when a mediator sends a rewritten query to related sources and receives at least an empty answer. This means that the case memory of the mediator is not updated. Typically, an information source has been removed from the system or changed its schema.

Example 4.7 Let us suppose that source w_1 has been (perhaps temporarily) removed, then the evaluation of the reformulated query in Example 4.4 fails. Indeed $I_{w_1}(Vpub.Vkeyword.\{Agents\}) = I_{w_1}(Vpub.journal) = I_{w_1}(Vpub.Vpublisher.\{ACM'\}) = I_{w_1}(Vpub.Vkeyword.\{AI\}) = \emptyset$

5 Distributed Query Rewriting

Even if a local failure (in query reuse or evaluation) occurs, the distributed case-based reasoning has still a possibility of solving the problem by means of cooperation with other mediators and sources. Notice that this provides the system the most important way of learning. Indeed, if the query is reformulated, the new rewritten query and/or the new sources can be stored as a new case. In such a way the mediator can support dynamic information systems. Furthermore, it does not need to maintain consistent its case memory every time something changes, but only when a consumer sends a query that fails. Therefore, the distributed information system is not overloaded with consistency maintenance operations, that usually are time consuming. On the contrary, traditional mediation systems can only update mediated schema with expensive hand-made procedures.

Cooperation strategies can be classified according to three parameters: partners, queries, and answers. In the following we describe such strategies, we also analyse them (from a theoretical point of view), since the choice of the right cooperation strategy is crucial for effectiveness and efficiency, and discuss the results.

Partners. When a mediator (say M) fails, it can cooperate with other *mediators* asking them to rewrite a query according to their own case memories. If they succeed, M can store

the result as a new case in its case memory. The mediators involved in this strategy can be all the mediators of the system, just mediators that have never been in touch with M , or mediators that successfully cooperated with M in the past. The mediator M can directly cooperate with *information sources* as well. This strategy can involve all the sources (it is very expensive), sources responsible of the local failure (this strategy takes into account changes of schemas), recently added sources (this strategy takes into account new sources), or the sources that successfully cooperated with M in the past (when a user has a new information need, the application of this strategy supposes that the new need involves the same sources of the past). Each source classifies the query and returns the subsumed concepts. Let us analyse such strategies focusing on recall and precision ratios in order to evaluate the effectiveness of the strategy. We also evaluate the information redundancy that such strategies produce.

First, let us consider a mediator M that cooperates with other mediators, sends them the original query, and asks for receiving a rewritten query. This strategy has a possible disadvantage that another mediator can send a wrong rewriting. This fact has harmful consequences on the precision, as stated by the following theorem.

Theorem 5.1 Let $S_1, \dots, S_n$ be n information sources. Let $\mathcal{V}$ be a view of $S_1, \dots, S_n$. $\mathcal{V}$ is represented as a case memory that does not change. Let M, N be two mediators such that M interacts with N when M fails. Let $\mathcal{C}_n(M)$ be the case memory of M after n interactions with N . Then

$$\begin{aligned} \text{Recall:} \quad & \lim_{n \rightarrow \infty} \frac{card(\mathcal{C}_n(M) \cap \mathcal{V})}{card(\mathcal{V})} \leq 1 \\ \text{Precision:} \quad & \lim_{n \rightarrow \infty} \frac{card(\mathcal{C}_n(M) \cap \mathcal{V})}{card(\mathcal{C}_n(M))} \text{ is indeterminate} \end{aligned}$$

In the above theorem, $\mathcal{C}_n(M)$ denotes the case memory of the mediator M (e.g. see Figure 2). $\mathcal{V}$ denotes the information need of a given consumer, i.e., it is the schema to which $\mathcal{C}_n(M)$ must converge. Such a theorem states that when the mediator N sends a rewriting that is part of $\mathcal{V}$ the recall and the precision grow, otherwise the recall does not change and the precision decreases. This means that is crucial the choice of what and how many mediators to cooperate with. Another possible disadvantage of such a strategy is that asymptotically all the mediators may have the same case memory, i.e. this strategy produces a certain redundancy. This result is a consequence of the following theorem.

Theorem 5.2 Let M, N be two mediators such that M interacts with N when M fails. Let $\mathcal{C}_n(M)$ be the case memory of M after n interactions with N . Let $\mathcal{C}(N)$ be the case memory of N such that it does not change while N interacts with M . Then

$$\lim_{n \rightarrow \infty} \frac{\text{card}(\mathcal{C}_n(M) \cap \mathcal{C}(N))}{\text{card}(\mathcal{C}(N))} = 1$$

The above theorem states that the case memory of M converges to the case memory of N when M cooperates with N . This seems to suggest that when the mediator cooperates with other mediators it would directly ask data instead of the rewritten query. This would allow any mediator to maintain its expertise and avoid redundancy.

As a second strategy, let us consider a mediator M that directly cooperates with information sources, sends them the original query, and receives the rewritten query (and eventually the data). This strategy guarantees a given consumer that the mediator M converges to the consumer's information need. Indeed it is possible to prove the following theorem.

Theorem 5.3 Let $S_1, \dots, S_n$ be n information sources. Let $\mathcal{V}$ be a view of $S_1, \dots, S_n$. $\mathcal{V}$ is represented as a case memory that does not change. Let M be a mediator such that M interacts with $S_1, \dots, S_n$ when it fails. Let $\mathcal{C}_n(M)$ be the case memory of M after n interactions with $S_1, \dots, S_n$. Then

$$\begin{aligned} \text{Recall: } \lim_{n \rightarrow \infty} \frac{\text{card}(\mathcal{C}_n(M) \cap \mathcal{V})}{\text{card}(\mathcal{V})} &= 1 \\ \text{Precision: } \lim_{n \rightarrow \infty} \frac{\text{card}(\mathcal{C}_n(M) \cap \mathcal{V})}{\text{card}(\mathcal{C}_n(M))} &= 1 \end{aligned}$$

The above theorem states that the mediator will asymptotically satisfy the information need of a given consumer (denoted by $\mathcal{V}$). Indeed, under the hypothesis that no name heterogeneity occurs (see Section 3), it is impossible that the source has wrong schemas. Therefore, the information source has either the right answer or no answer at all. This means that the choice of sources to cooperate with is crucial only for efficiency of the system, i.e. how fast the mediator's case memory converges to user's needs.

Example 5.1 In the situation of Example 4.7, let us suppose that the mediator chooses to look for new sources and cooperate with them. Let us suppose that w_4 is a new information source which contains ACM publications and has the following concepts in its schema:

$$\begin{aligned} \text{acm_tods} &\doteq \forall j_name. \{ \text{"TODS"} \} \sqcap \forall publisher. \{ \text{"ACM"} \} \\ \text{acm_tocl} &\doteq \forall j_name. \{ \text{"TOCL"} \} \sqcap \forall publisher. \{ \text{"ACM"} \} \\ \text{acm_tois} &\doteq \forall j_name. \{ \text{"TOIS"} \} \sqcap \forall publisher. \{ \text{"ACM"} \} \end{aligned}$$

The mediator decomposes the query Q_3 and sends each component to w_4 . For example, when w_4 receives $\forall pub. \forall publisher. \{ \text{"ACM"} \}$, it looks for maximally contained concepts and obtains $M_{inf}(\forall pub. \forall publisher. \{ \text{"ACM"} \}, \mathcal{P}) = \{ \forall pub. \text{acm_tods}, \forall pub. \text{acm_tocl}, \forall pub. \text{acm_tois} \}$. When the mediator receives the answer, retains as new case:

$$\langle \forall pub. \text{acm}, \langle \forall pub. \text{acm_tods}, \forall pub. \text{acm_tocl}, \forall pub. \text{acm_tois} \rangle, \langle (\forall pub. \text{acm_tods}, w_4), (\forall pub. \text{acm_tocl}, w_4), (\forall pub. \text{acm_tois}, w_4) \rangle \rangle$$

Queries. Cooperation strategies can also be classified according to queries. Indeed M can send the query as received by the consumer (this is a strategy that takes into account the new user's need). It can send the retrieved query (if any), i.e. the problem returned by the retrieval function (this is a strategy that takes into account the need of maintaining consistent the mediator's case memory). Finally, the mediator can send the reformulated query (if any). In the latter case the "approximation of the solution grows", as stated by the following theorem.

Theorem 5.4 Let Q be the original query. Let $Q' \in M_{inf}(Q, \mathcal{P})$ be one of the past queries retrieved by M . Let $Sol_M(Q')$ be one of the reformulations of Q' used by M as rewriting of Q . Let N be the mediator or source that cooperates with M . Let $Sol_N(x)$ be the query rewritten by N if N receives x as input query. Then

$$\begin{aligned} Sol_N(Q) &\subseteq Q, & Sol_N(Q') &\subseteq Q' \subseteq Q \\ Sol_N(Sol_M(Q')) &\subseteq Sol_M(Q') \subseteq Q' \subseteq Q \end{aligned}$$

As a consequence, the solution returned by N is more distant from Q than Q' when M does not send the original query.

Cooperation strategies can also be classified depending on the fact that the query can be sent as it is or be decomposed in basic components. This second strategy is applicable since it is straightforward to prove that the following theorem holds when we use C-CLASSIC:

Theorem 5.5 Let $Q_1, \dots, Q_n$ be queries. Let $Q = f(Q_1, \dots, Q_n)$ be the user's query. Let N be the mediator or source that cooperates with M . Let $Sol_N(x)$ be the query rewritten by N if N receives x as input query. Then

$$Sol_N(f(Q_1, \dots, Q_n)) = f(Sol_N(Q_1), \dots, Sol_N(Q_n))$$

The theorem above states that M does not need to decompose a query before its sending, since the query is directly decomposed by partners. This suggests that decomposition is convenient (efficient) only if M failed to reformulate / evaluate just some components of the query.

Answers. Finally, cooperation strategies can be classified according to answers. M can ask for rewriting the query. Its goal is to update its case memory with a rewritten query obtained from its collaborators. M can also ask for data that answer the query. In such a situation, it stores in the case memory the addresses of the mediators/sources that answered. We can analyse such strategies depending on partners. Table 1 reports the results of cooperation with only one agent. We assume that such an agent (say a source S or a mediator N) does not change while the strategy is in progress. We also suppose that N has $\langle Q, Sol_S(Q), S \rangle$ in its case memory. For each strategy, Table 1 reports the new case that M stores in its case memory, the set of instances related to the reformulation, the number of messages to obtain instances of Q (when M receives Q the first time and the next time), and the results about redundancy (see Theorem 5.2). When we consider cooperation with mediators, it seems more efficient to ask for a rewriting, since M interacts with N only the first time, the next time M directly interacts with S . On the other hand, the strategy of requiring data does not produce redundancy, even if it is less efficient. Notice that, for what concerns such evaluation parameters, the strategy of cooperating

Partner	Answer	New Case	Data	No. of messages.		Redundancy
				1st. time	Next time	
Source (S)	Rewriting	$\langle Q, Sol_S(Q), S \rangle$	$I_S(Sol_S(Q))$	4	2	No
"	Data	$\langle Q, Q, S \rangle$	$I_S(Sol_S(Q))$	2	2	No
Mediator (N)	Rewriting	$\langle Q, Sol_S(Q), S \rangle$	$I_S(Sol_S(Q))$	4	2	Yes
"	Data	$\langle Q, Q, N \rangle$	$I_S(Sol_S(Q))$	4	4	No

Table 1: A comparison of different cooperation strategies.

with sources requiring data seems to be the most appropriate when the number of sources is comparable to the number of mediators.

6 Conclusions

We considered the problem of rewriting queries by means of distributed case-based reasoning. As far as we know, this is a novel approach. This allows us to have dynamic mediated schemas instead of static ones. We showed that a real dynamic mediator is obtained by means of distributed query reformulation, i.e., cooperation with other mediators and sources. Thanks to this approach, the mediator is able to be updated when sources and consumers are added/removed or change their schemas/needs. This operation is performed only when a consumer sends a query that fails and not every time something changes. This allows us to avoid of overloading the distributed information system with expensive consistency maintenance operations. In distributed case-based reasoning, the choice of the right cooperation strategy is crucial. We hinted several possible strategies and discussed their advantages. From such an analysis arises that we can obtain good results for effectiveness and efficiency when a mediator directly cooperates with information sources when the number of information sources is small. Otherwise, the most efficient strategy is the cooperation with other mediators. Finally, notice that the choice of C-CLASSIC is appropriate since the algorithm complexity for query retrieval and reuse is polynomial.

Acknowledgements

We thank J. Mylopoulos for suggestions and useful discussions about the topics presented in this paper. We also thank P. Coupey and S. Gianfelici that have participated to the development of the C-CLASSIC reasoner.

References

- [Arens *et al.*, 1993] V. Arens, C. Y. Chee, C-N. Hsu, and C. A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.
- [Beeri *et al.*, 1997] C. Beeri, A. Y. Levy, and M.-C. Rousset. Rewriting Queries Using Views in Description Logics. In *Proc. of the 16th ACM Symposium on Principles of Database Systems (PODS)*, pages 99–108, New York, NY, May 12–14, Tucson, Arizona 1997. ACM Press.
- [Borgida and Patel-Schneider, 1994] A. Borgida and P. F. Patel-Schneider. A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.
- [Diamantini and Panti, 1999] C. Diamantini and M. Panti. A Conceptual Indexing Method for Content-Based Retrieval. In A. M. Tjoa, A. Cammelli, and R. R. Wagner, editors, *Tenth International Workshop on Database and Expert Systems Applications (DEXA'99)*, Florence, Italy, 1–3 September 1999. IEEE Computer Society.
- [Duschka and Genesereth, 1997] O. M. Duschka and M. R. Genesereth. Infomaster - An Information Integration Tool. In *Proc. of the International Workshop on Intelligent Information Integration*, Freiburg, Germany, September 1997.
- [Garcia-Molina *et al.*, 1997] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: data models and languages. *Journal of Intelligent Information Systems*, 8:117–132, 1997.
- [Kirk *et al.*, 1995] T. A. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In *AAAI Spring Symposium on Information Gathering*. AAAI, 1995.
- [Koehler, 1994] J. Koehler. An Application of Terminological Logics to Case-based Reasoning. In P. Torasso, J. Doyle, and E. Sandewall, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 351–362, Bonn, Germany, May 1994. Morgan Kaufmann.
- [M. Panti, L. Spalazzi, A. Giretti, 2000] M. Panti, L. Spalazzi, A. Giretti. A case-based approach to information integration. In *Proceedings of the 26th International Conference on Very Large Databases*, Cairo, Egypt, 10–14 September 2000.
- [Plaza *et al.*, 1997] E. Plaza, J. L. Arcos, and F. Martín. Co-operative Case-Based Reasoning. In G. Weiss, editor, *Distributed Artificial Intelligence meets Machine Learning*, Lecture Notes in Artificial Intelligence, Berlin, 1997. Springer Verlag.
- [Prasad *et al.*, 1996] M. V. N. Prasad, V. R. Lesser, and S. E. Lander. Retrieval and reasoning in distributed case bases. *Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries*, 7(1):74–87, March 1996.
- [Ullman, 1997] J. D. Ullman. Information Integration Using Logical Views. In *Proceedings of the International Conference on Database Theory*, pages 19–40, 1997.
- [Wiederhold, 1992] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer Magazine*, 25:38–49, March 1992.