Query-Answering Algorithms for Information Agents

Alon Y. Levy
AI Principles Research Department
AT&T Research
levy@research.att.com

Anand Rajaraman*
Department of Computer Science
Stanford University
anand@cs.stanford.edu

Joann J. Ordille
Computing Sciences Research Center
Bell Laboratories
joann@bell-labs.com

Abstract

We describe the architecture and query-answering algorithms used in the Information Manifold, an implemented information gathering system that provides uniform access to structured information sources on the World-Wide Web. Our architecture provides an expressive language for describing information sources, which makes it easy to add new sources and to model the fine-grained distinctions between their contents. The query-answering algorithm guarantees that the descriptions of the sources are exploited to access *only* sources that are relevant to a given query. Accessing only relevant sources is crucial to scale up such a system to large numbers of sources. In addition, our algorithm can exploit run-time information to further prune information sources and to reduce the cost of query planning.

Introduction

The number of structured information sources that is available online is growing rapidly. For example, there are many sources on the World-Wide Web (WWW) that provide a query interface (e.g., HTML forms). Even more sources behave as structured sources if we use an appropriate interface program (e.g., for parsing structured text files). As another example, large institutions have a vast number of internal databases available online. Even though each of these sources is structured and supports high-level queries, the interaction with a multitude of sources is much like *browsing.* The user must consider the list of available sources, decide which ones to access, and then interact with each one individually. Furthermore, the user must manually *combine* information from multiple sources.

Information agents are systems that provide a uniform query interface to multiple information sources (e.g., Internet Softbot [Etzioni and Weld, 1994], SIMS [Arens *et al.*, 1996], TSIMMIS [Chawathe *et al.*, 1994], Infomaster [Geddis *et al.*, 1995], HERMES [Subrahmanian *et al.*, 1995], Nomenclator [Ordille and Miller, 1993]). In such a system the user expresses *what* he or she wants, and the system determines which information sources are relevant to the query using *descriptions* of the sources available to the system. The system therefore frees the user of having to find the relevant information sources and interact with each one separately, and it combines information from multiple sources to answer user queries. Such systems face many challenges because the sources are autonomous, heterogeneous and use different data models and vocabularies. One of the most important challenges arises from the fact that many information sources contain *closely related* data. For example, there may be many sources describing items for sale and their properties, and the sources differ depending on properties such as the type of items in the database, price ranges, etc. The descriptions of the information sources need to be able to model the fine-grained differences between contents of sources. Primarily, such modeling ability is needed in order to be able to prune the number of sources accessed for a given query, since the number of information sources accessed is a dominant factor in the cost of query processing.

This paper describes the architecture and query-answering algorithm used in the Information Manifold, an implemented system that provides uniform access to structured information sources on the WWW. The Information Manifold has several distinguishing features. First, contents of information sources are described by *query expressions* that can use classes and roles defined in the CLASSIC description language as well as ordinary predicates of any arity and order predicate (e.g., $\leq$, $<$). Consequently, it is easy to *add* information sources without changing the view seen by the user, and to model the fine-grained differences between the contents of sources. Second, the query-planning algorithm uses the source descriptions to determine *precisely* which information sources (and combinations of information sources) can yield answers to the query. Pruning irrelevant sources is crucial for scaling up such

*Part of this work was done while this author was visiting AT&T Bell Laboratories.

a system to a large number of sources. Third, the algorithm inserts (efficiently verifiable) conditions in the query plans that enable using run-time bindings to prune additional source accesses.

Example

We illustrate the main components of the Information Manifold architecture and algorithms with the following example that we use throughout the paper. A user of the Information Manifold is presented with a *world-view* which is a set of relations (including a class hierarchy) modeling the kinds of information that can be obtained from the available sources. For example, suppose we are providing access to several databases describing second-hand cars for sale and their properties, and to several servers providing reviews of cars and other products. The world-view includes the relation $CarFS(model, year, price, owner)$ describing the car's model, year of manufacture, price and owner contact number, and the relation $Review(product, year, review)$ providing a review (e.g., string of text) for products. The world-view also contains a hierarchy of car classes shown in Figure 1.

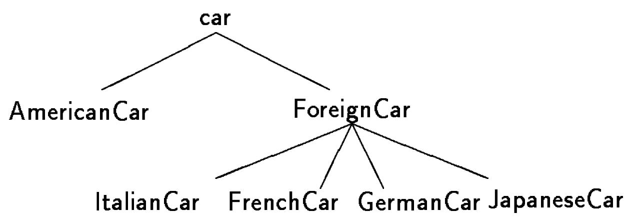

Figure 1: Class hierarchy of cars. The classes AmericanCar and ForeignCar are disjoint, as are ItalianCar, FrenchCar, GermanCar and JapaneseCar.

The relations in the world-view are only *virtual*. All the actual data are stored in external information sources. The Information Manifold contains descriptions of contents of these information sources, as seen in Figure 2. Information sources are modeled by describing which relations can be found in them. The relation in an information source (denoted throughout the paper by $v_1, \dots, v_n$) is described by a conjunctive formula. Informally, the formula describes the constraints satisfied by facts in the relation found in the source. In the example, information sources 1 and 2 contain cars for sale. Source 1 contains the relation v_1 that provides only Japanese cars manufactured beginning in 1990, while source 2 has only luxury cars priced over $20,000. Note that neither of these sources is said to contain *all* the cars satisfying these constraints. Sources 3–5 provide reviews of products. Source 3 provides reviews of foreign cars, source 4 provides reviews of cars manufactured up to 1989, and source 5 provides reviews of cars manufactured in the past 6 years.

Source 1: Japanese cars beginning 1990.
$v_1(model, year, price, owner)$:
$\quad CarFS(model, year, price, owner) \wedge$
$\quad JapaneseCar(model) \wedge year \geq 1990.$
Source 2: Luxury cars
$v_2(model, year, price, owner)$:
$\quad CarFS(model, year, price, owner) \wedge price \geq \$20,000.$
Source 3: Foreign car reviews
$v_3(product, year, review)$:
$\quad Review(product, year, review) \wedge ForeignCar(product)$
Source 4: Old car reviews
$v_4(product, year, review)$:
$\quad Review(product, year, review) \wedge Car(product) \wedge$
$\quad year \leq 1989$
Source 5: Last 6 years car reviews
$v_5(product, year, review)$:
$\quad Review(product, year, review) \wedge Car(product) \wedge$
$\quad 1991 \leq year \leq 1996$

Figure 2: Example information sources

Queries to the system are formulated in terms of the world-view relations, thereby freeing the user from having to know the specific vocabulary used in every information source. For example, suppose the user is interested in finding Japanese cars and their reviews, for cars manufactured beginning 1988 and priced at less than $5000. Ordinarily, the user would have to access each of the car sources individually, retrieve all the cars that may be relevant, and then search for a review for every relevant car. In the Information Manifold the user can pose the following query:

$$\begin{aligned}&q(Mo, Ye, Ow, Pr, Re) :\\ &\quad CarFS(Mo, Ye, Pr, Ow) \wedge Review(Mo, Ye, Re) \wedge\\ &\quad JapaneseCar(Mo) \wedge Pr \leq \$5000 \wedge Ye \geq 1988.\end{aligned}$$

Given a query, the system uses the descriptions of the sources to generate a query-answering plan. As stated, the key challenge we address in our system is generating a plan that prunes as many information sources as possible.

To answer this query, our algorithm first decides which information sources are relevant. Source 1 is relevant because it contains Japanese cars, and the manufacturing years given in the source (≥ 1990) overlap with the years requested in the query (≥ 1988). Source 2 is irrelevant because the prices of the cars in the source are too expensive. The source reviews (3–5) are relevant to the query because they each contain reviews of cars that may be relevant. For example, since the class $ForeignCar$ subsumes the class $JapaneseCar$, Source 3 is relevant.

Next, the algorithm will compute *conjunctive plans* for answering queries by considering the possible combinations of relevant sources. Combining information from Sources 1 and 3 is possible, because the reviews of cars that are found in Source 1 can be found in

Source 3. Therefore, the conjunctive plan:

$$v_1(Mo, Ye, Pr, Ow) \wedge v_3(Mo, Ye, Re) \wedge Pr \leq \$5000$$

is a valid one, and furthermore all the answers it returns are guaranteed to be answers to the query, since the Sources 1 and 3 together enforce the constraints on the country and year of manufacture that are required in the query. Similarly, combining Sources 1 and 5 is also a valid conjunctive plan.

Even though Source 4 is relevant to the query, it cannot be used in conjunction with Source 1, since the years of coverage of the two sources do not overlap. Therefore, there is no way to use Source 4 for answering this query, given the current sources.

Finally, the reasoning described above takes into consideration the constraints in the query and the descriptions of the sources. At run-time we may obtain additional information that enables pruning source accesses. For example, consider the plan combining Sources 1 and 5. If the year of the car obtained from Source 1 is 1990, then there is no point going to Source 5. Therefore, our algorithm will insert a run-time condition $Ye \geq 1991$ that will be checked after accessing Source 1. If the condition is not satisfied, Source 5 will not be searched for a review for the specific car. □

There are many issues that need to be addressed in a system that provides integrated access to multiple information sources. This paper focuses on the problem of selecting the relevant sources to a query and creating a query plan. Important aspects of our system that we do not describe include the *ordering* of accesses to information sources (which has significant impact on performance) and the question of determining whether two constants in different sources refer to the *same* object in the world.

Representation of information sources

Preliminaries

The representation language used in the world-view and information source descriptions in the Information Manifold, CARIN-CLASSIC [Levy and Rousset, 1996], combines the description logic used in CLASSIC and relations of arbitrary arity. Description logics are very natural in modeling multiple information sources since many sources can be viewed as containing instances of some class of objects, and we need to reason about the relationship among these classes. However, in order to model sources that may be arbitrary relational databases, we need the ability to represent relations with arbitrary arity. In addition, our language contains the interpreted order relations $\{\leq, <, =, \neq\}$.

Formally, unary relations in the world-view are called *classes,* and binary relations are called *roles.* Classes in the world-view are associated with *descriptions.* A description in CLASSIC can be built using the following grammar: (A denotes a class name, C and D denote descriptions, and R denotes a role):[1]

$$C, D : -A \mid C \sqcap D \mid (\mathsf{all}\ R\ C) \mid (\geq\ n\ R) \mid (\leq\ n\ R)$$

Intuitively, associating a concept C in the world-view with a complex description D specifies the conditions for an individual to belong to C. For example, the following is a description of the class of Japanese cars, for which all previous owners are American and that had at most two previous owners:

$JapaneseCar \sqcap (\text{all } owner\ AmericanPerson) \sqcap (\leq 2\ owner)$.

A query Q to the Information Manifold is a conjunction of the form:

$$Q(\bar{Y}) : \exists \bar{X}\, p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n),$$

where $\bar{X}, \bar{X}_1, \ldots, \bar{X}_n$ are tuples of variables or constants. and the p_i's are any of the relations in the world-view. The variables in $\bar{Y}$, called the *distinguished variables* of the conjunctive formula, are the variables that appear in $\bar{X}_1 \cup \ldots \cup \bar{X}_n$, but not in $\bar{X}$, and the answer to the query is the set of bindings that we can obtain for the variables in $\bar{Y}$. We require that any variable that appears in an atom of an order relation in a query also appear in a atom of non-order relation.

If ψ is a conjunctive formula with distinguished variables $Y_1, \ldots, Y_l$, and $\bar{Z} = Z_1, \ldots, Z_l$ is a tuple of variables, we denote by $\psi(\bar{Z})$ the formula in which Y_i is replaced by Z_i, for i, $1 \leq i \leq l$.

The contents of an information source

There are several desiderata for the representation of information sources. First, since many sources contain closely related data, it is important to be able to model the fine-grained differences between their contents, in order to be able to determine precisely which sources are relevant to a given query. In our example, there are several sources for cars and for reviews, but determining which are relevant was based on reasoning about the constraints on their contents. Second, since the number of information sources is likely to be large and frequently changing, we should be able to add new information sources without *changing* the world-view for each one of them, even if the contents of the information source do not correspond *directly* to one of the relations in the world-view. Finally, the language for describing sources should enable us to reason effectively about relevance of sources to queries.

We model an information source as containing tuples of a relation (or several relations), and the description of the source specifies the constraints on the tuples that can be found in the relation. Formally, the contents of an information source are described by a pair (or set of pairs) of the form (v, r_v), where v is a relation name with arity m_v, and r_v is a conjunctive formula of the

[1] These are only a subset of the CLASSIC constructors.

form $\exists \bar{U}\, p_1(\bar{U}_1) \wedge \ldots \wedge p_n(\bar{U}_n)$, i.e., a conjunctive query. The formula r_v has m_v distinguished variables. The relation name v is a new name that does not appear in the world-view, and is only used in a single pair in the description of one information source.

Such a description means that the source can be asked a query of the form $v(\bar{Z})$ (or any partial instantiation of it), and returns tuples of arity m_v that satisfy the following implication: $(\forall \bar{Z})[v(\bar{Z}) \Rightarrow r_v(\bar{Z})]$. That is, every tuple obtained from the information source satisfies the conjunction r_v. Note that the description does not imply that the source contains *all* the tuples that satisfy r_v (see [Etzioni *et al.*, 1994] for a formalism dealing with *complete* information).

The representation of information sources satisfies our desiderata. The expressive power of conjunctive queries, CLASSIC descriptions and order constraints provides a very rich language in which fine-grained distinctions between sources can be expressed. The fact that sources are described as *queries* means that we do not have to add a relation to the world-view whenever sources are added, since the relation found in a source does not have to correspond *directly* to a relation in the world-view. Finally, as we show in subsequent sections, we can determine *exactly* which sources are relevant to a given query.

Example 1: Source 1, containing Japanese cars whose manufacturing year is 1990 or later, is described by the pair

$(v_1(model, year, price, owner),$
$\quad CarFS(model, year, owner, price) \wedge$
$\quad JapaneseCar(model) \wedge year \geq 1990)).$

As an example of the advantage of describing contents as conjunctive queries, suppose we encounter a source in which Japanese cars have already been matched with their reviews. We do not have to create a new relation that has all the attributes of a car. Instead, we can describe the source as follows:

$(v_6(model, year, price, owner, review),$
$\quad CarFS(model, year, owner, price) \wedge$
$\quad Review(model, year, review) \wedge JapaneseCar(model)).$

Query plans

Given a query of the form

$$Q(\bar{Y}) : \exists \bar{X}\, p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n),$$

the query processor generates a set of *conjunctive plans* for answering $Q(\bar{Y})$. A conjunctive plan is a sequence of operations that accesses some of the information sources, and performs local relational operations (e.g., join, union, selections). In this paper we focus on the logical aspects of query-answering plans, and therefore we represent a conjunctive plan P as a conjunctive query *over the information source relations*, i.e., as a formula of the form

$$Q(\bar{Y}) : \exists \bar{U}\, v_1(\bar{U}_1) \wedge \ldots \wedge v_l(\bar{U}_l) \wedge C_P$$

where each of the v_i's is a relation name associated with an information source, and C_P is a conjunction of atoms of order relations. Note that the distinguished variables in the plan are the same as the ones in the query. Given a conjunctive plan P, the descriptions of the information sources imply that the following constraints hold on the answers it produces: (recall that r_{v_i} is the formula describing the constraints on the tuples found in v_i)

$$Con_P : r_{v_1}(\bar{U}_1) \wedge \ldots \wedge r_{v_l}(\bar{U}_l) \wedge C_P.$$

Example 2: The plan $P_{1,3}$

$$v_1(Mo, Ye, Pr, Ow) \wedge v_3(Mo, Ye, Re) \wedge Pr \leq \$5000$$

guarantees that the answers returned satisfy $Con_{P_{1,3}}$

$CarFS(Mo, Ye, Pr, Ow) \wedge JapaneseCar(Mo) \wedge$
$Ye \geq 1990 \wedge Review(Mo, Ye, Re) \wedge$
$ForeignCar(Mo) \wedge Pr \leq \$5000.$ □

A query-answering plan is a set of *sound* conjunctive plans, as defined below.

Definition 1: *A conjunctive plan P is* sound *if all the answers it produces are guaranteed to be answers to the query, i.e., if the following entailment holds:*

$$(\forall \bar{Y}) Con_P \Rightarrow [\exists \bar{X}\, p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n)].$$

Note that we may need to use several conjunctive plans to answer a query because the information sources are not *complete,* and therefore one combination of information sources may not yield all the answers. The answer to a query $Q(\bar{Y})$ is defined to be the set of tuples that can be obtained by some sound conjunctive plan.

Computing query plans

When accessing external sources, the cost of query processing is dominated by the time taken to access external information sources, and the amount of data that needs to be transferred. Therefore, the main optimization we consider is to minimize the number of sources accessed. The set of conjunctive plans considered by a query processor provides an important measure by which to test its pruning ability. The Information Manifold is distinguished in that it only considers sound and *relevant* conjunctive plans. A conjunctive plan is relevant if, according to the descriptions of the sources and the constraints in the query, it *may* produce answers to the query.

Definition 2: *A conjunctive plan P is relevant to a query $Q(\bar{Y}) : \exists \bar{X}\, p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n)$ if the sentence $(\exists \bar{Y}, \bar{X}) Con_P \wedge p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n)$ is satisfiable.* □

Intuitively, this means that given the constraints on the contents of the sources, there *may* be tuples in each of the sources accessed by the plan P, such that together they can generate an answer to the query, using P. We now describe our query-planning algorithm.

The challenge in computing conjunctive plans is to ensure that the constraints required in the query are enforced by the information sources in the plan, or can be enforced by additional constraints in the plan. The problem is that an information source may not make all the constrained arguments available. For example, if Source 1 projected out the price of the car, we would not be able to enforce any additional constraint on the price that isn't already enforced by the source.

Example 3: In our example, the conjunctive plan $v_1(Mo, Ye, Pr, Ow) \wedge v_3(Mo, Ye, Re) \wedge Pr \leq \$5000$ is relevant, because the cars found in Source 1 overlap with the cars reviewed in Source 3, and therefore the plan can generate answers to the query. Note that this plan only generates a subset of the answers, because it can only return a restricted class of Japanese cars. The conjunctive plan $v_1(Mo, Ye, Pr, Ow) \wedge v_4(Mo, Ye, Re)$ is not a relevant plan because cars found in Source 1 are not reviewed in Source 4. □

Computing query plans proceeds in two steps. In the first step, we compute, separately for each subgoal in the query, which information sources are relevant to it. Informally, an information source is relevant to a subgoal g if, the description of the source contains a subgoal g_1 that can be unified with g, such that after the unification, the constraints in the query and the constraints in the source description are mutually satisfiable. The details of this step are shown in Figure 3.

In the second step of the algorithm we consider conjunctive plans constructed by chosing one relevant source for every subgoal in the query, and check each one for soundness and relevance. Specifically, we consider every conjunctive plan Q' of the form

$$q'(\bar{Y}) \quad : \quad (\exists \bar{U}) v_1(\bar{U}_1) \wedge \ldots \wedge v_n(\bar{U}_n)$$

where $v_i(\bar{U}_i)$ has been deemed relevant to subgoal p_i in the query. For each such conjunctive plan we check that it is (1) **relevant,** (2) **sound** (if it is not a sound plan, we check whether it can be made sound by adding conjuncts of order predicates,[2] and (3) **minimal** (i.e., we cannot remove a subgoal from the plan and still obtain a sound plan). These properties are checked using algorithms for containment of conjunctive queries in CARIN [Levy and Rousset, 1996].

Example 4: Continuing with our example, our algorithm will consider the sources relevant to the first subgoal, $CarFS(Mo, Ye, Pr, Ow)$. The description of Source 1 has a subgoal of the relation $CarFS$, and therefore the algorithm will check whether the constraints in the query are consistent with the constraints on the data in the source. Specifically, it will check that the following is satisfiable:

$JapaneseCar(Mo) \wedge Ye \geq 1990 \wedge$
$Ye \geq 1988 \wedge Pr \leq \$5000.$

[2] If a sound plan can be obtained, it is enough to consider additional order atoms that include constants that occur in the *original* query, Q [Levy *et al.*, 1996].

```
procedure generate-relevant-sources($\mathcal{I}$,$Q$)
/* $\mathcal{I}$ is a set of information sources, and $Q$ is a conjunctive
query of the form $q(\bar{X}) : (\exists \bar{X}) g_1(\bar{X}_1) \wedge \ldots g_m(\bar{X}_m) \wedge C_q$,
where $C_q$ is the conjunction of order atoms in $Q$. */

for every subgoal $g_i(\bar{X}_i)$, $1 \leq i \leq m$ do:
  $relevantSources_i = \emptyset$
  for every non-order conjunct $u(\bar{Y})$ in a formula
    $r_v$ in the pair $(v, r_v)$ in a description of a source in
    $\mathcal{I}$ do:
    if $g_i = u$ or $g_i$ and $u$ are non-disjoint classes then:
      Let $\psi$ be the mapping defined on the variables of
      $r_v$ as follows:
        if $Y$ is the $j$'th variable in $\bar{Y}$ and is not
          existentially quantified in $r_v$
          then $\psi(Y) = X_j$, where $X_j$ is the $j$'th
          variable in $\bar{X}_i$.
          else $\psi(Y)$ is a new variable that does not
          appear in $Q$ or $r_v$.
      Let $C(Q)$ and $C(v)$ be the conjunction of
        constraint subgoals in $Q$ and $\psi(r_v)$, respectively.
      if $C(Q) \wedge C(v)$ is satisfiable, then add $\psi(r_v)$
        to $relevantSources_i$.
return $\{relevantSources_1, \ldots, relevantSources_m\}$.
end generate-relevant-sources.
```

Figure 3: Algorithm for determining which sources are relevant to each subgoal in the query. The *constraint subgoals* in a conjunctive formula include all the atoms with either order predicates, or CLASSIC classes and roles. Disjointness of constraints is determined by combining algorithms for disjointness of order constraints [Ullman, 1989] and of class constraints [Levy and Rousset, 1996].

Since it is, the algorithm will return $v_1(Mo, Ye, Pr, Ow)$ in $relevantSources_1$. Since the price of cars in Source 2 is higher than those required in the query, the algorithm will determine that Source 2 is irrelevant.

For the second subgoal, $Review(Mo, Ye, Re)$, the algorithm will return $(v_3(Mo, Ye, Re), v_4(Mo, Ye, Re), v_5(Mo, Ye, Re))$ for $relevantSources_2$.

In the second step, the algorithm will first consider a plan combining Sources 1 and 3. Since the following implication holds, the plan is sound.

$$\begin{aligned}
&[CarFS(Mo, Ye, Pr, Ow) \wedge JapaneseCar(Mo) \wedge \\
&\quad Review(Mo, Ye, Re) \wedge ForeignCar(Mo) \wedge \\
&\quad Ye \geq 1990 \wedge Pr \leq \$5000] \\
&\quad \Rightarrow \\
&\quad [CarFS(Mo, Ye, Pr, Ow) \wedge JapaneseCar(Mo) \wedge \\
&\quad Review(Mo, Ye, Re) \wedge Ye \geq 1988 \wedge Pr \leq \$5000]
\end{aligned}$$

Furthermore, it is also a relevant plan because the following formula is satisfiable:

$ForeignCar(Mo) \wedge Ye \geq 1990 \wedge JapaneseCar(Mo) \wedge$
$CarFS(Mo, Ye, Pr, Ow) \wedge Review(Mo, Ye, Re) \wedge$
$Pr \leq \$5000 \wedge Ye \geq 1988.$

The plan combining Sources 1 and 5 is also a sound and relevant one. These plans are also minimal, in that we cannot remove any subgoals from them. □

Our algorithm is guaranteed to produce *only* sound and relevant plans. Furthermore, it is possible to use the algorithm described in [Levy and Rousset, 1996] to compare conjunctive plans and to guarantee that no conjunctive plan is *redundant*. That is, every conjunctive plan may yield some answer that is not given by any of the other conjunctive plans. As a consequence, it can be shown that the plan accesses the minimal number of information sources relevant to the query. The second question to ask is whether our algorithm produces *all* the necessary conjunctive plans. The answer is based on the close relationship between the problem of finding conjunctive plans and the problem of *answering queries using materialized views*. It follows from [Levy *et al.*, 1995a] that if there are no order predicates or CLASSIC definitions, we need only consider conjunctive plans whose length is at most the length of the query, which is precisely what our algorithm does. As shown in [Levy *et al.*, 1995a] we may need to modify our algorithm to consider longer plans in the presence of order constraints and CLASSIC definitions. It also follows from that paper that the problem of rewriting queries using views is NP-complete in the number of sources even when order predicates and CLASSIC definitions are not present. The algorithm described here is designed to reduce the number of plans considered. The first step of our algorithm, in which we consider the sources relevant to each subgoal in isolation, provides a very effective filter on candidate plans we consider. It guarantees that in every plan we check, each of the information sources has *already* been found to be relevant to the corresponding subgoal in the query. This reduction is especially effective when the relevance of a source depends mostly on the subgoal it is being matched with and not on which information source is used to satisfy the other subgoals. Finally, it should be noted that although the cost of checking minimality and soundness of a conjunctive plan is exponential, it is exponential only in the size of the query, which tends to be small, and not in the number of information sources or their contents.

Exploiting run-time information

The conjunctive plans computed in the previous section consider only the constraints appearing in the query and in the descriptions of the information sources. This section describes two ways run-time information can be used to further optimize query evaluation. First we describe an algorithm for inserting run-time conditions that use bindings to check the relevance of a source, and then we describe how parts of the *planning* can be postponed until some parts of the plan have been executed and run-time bindings obtained. In this section we assume that the query planner decided on *some* ordering the subgoals of the plan. In our examples, the ordering is left to right.

Computing run-time conditions

Informally, the reason that a conjunctive plan may become irrelevant at run-time is that a binding obtained for some variable X from one of the information sources contradicts a constraint on X that is known to hold in an information source that is accessed subsequently in the conjunctive plan. In our example, even though the combination of Sources 1 and 5 is sound and valid, if the car obtained from Source 1 is from 1990, then its review will not be found in Source 5. Figure 4 describes an algorithm for computing additional run-time conditions that guarantee that we do not pursue useless conjunctive plans.

```
procedure compute-runtime-conditions($P$)
/* $P$ is a conjunctive plan of the form
        $v_1(\bar{U}_1) \wedge \ldots \wedge v_n(\bar{U}_n) \wedge C$,
where $C$ is a conjunction of order atoms. */

$Cond_0 = True$.
for $i = 1, \ldots, n-1$,
    $\bar{T}_i$ is the set of variables appearing in $v_1, \ldots, v_i$.
    $FinalCond_i$ is the projection of $Con_P$ on $\bar{T}_i$.
    $C_i$ is the projection of $C$ on the variables in $\bar{T}_i$.
    $TempCond_i = r_{v_1}(\bar{U}_1) \wedge \ldots \wedge r_{v_i}(\bar{U}_i) \wedge C_i$.
    $Cond_i$ is a non redundant constraint $D$ such that
            $TempCond_i \wedge D$ is satisfiable if and only if
            $FinalCond_i$ is satisfiable.
    Insert $Cond_i$ after $v_i$ in the plan.
end generate-runtime-conditions.
```

Figure 4: Computing run-time conditions.

Example 5: Consider the plan combining Sources 1 and 5. The final conditions that apply to the variables appearing in the first subgoal (i.e., $FinalCond_1$) are:

$$1991 \leq Ye \leq 1996 \wedge JapaneseCar(Mo) \wedge Pr \leq \$5000.$$

However, after obtaining bindings from Source 1 (and applying the condition on the price), these bindings are known to satisfy (i.e., $TempCond_1$ is):

$$1990 \leq Ye \wedge JapaneseCar(Mo) \wedge Pr \leq \$5000.$$

Therefore, the condition $D = 1991 \leq Ye \leq 1996$ is inserted between the accesses to Sources 1 and 5. □

The property of algorithm **compute-runtime-conditions** is summarized by the following theorem. Informally, it guarantees that at every point during the execution of a conjunctive plan, the information sources accessed *may* still lead to answers of the query. The proof is omitted because of space limitations.

Theorem 1: *Let $v_1(\bar{U}_1) \wedge \ldots \wedge v_n(\bar{U}_n) \wedge C$ be a sound and relevant conjunctive plan for the query Q, and let $c_1 \wedge \ldots \wedge c_{n-1}$ the conditions inserted in the plan by algorithm* **compute-runtime-conditions**. *Suppose the* *subgoals $v_1, \ldots, v_l$ have been executed yielding bindings $a_1, \ldots, a_m$ for the variables $Z_1, \ldots, Z_m$ appearing in $v_1, \ldots, v_l$, and that the conditions $c_1, \ldots, c_l$ are satisfied. Then, the following formula is satisfiable:*

$$(\exists \bar{Y}')\ Con_P \wedge (Z_1 = a_1) \wedge \ldots \wedge (Z_m = a_m),$$

where $\bar{Y}'$ are the variables that appear in Con_P but not in $Z_1, \ldots, Z_m$.

Interleaving planning and execution

In [Golden *et al.*, 1994; Knoblock, 1995] it is shown that there are significant advantages to interleaving query planning and execution in the presence of many information sources. Because of the large number of sources, the space of possible plans may be large. After executing *partial* plans (i.e., plans that answer a subset of the subgoals in the query), we obtain additional information that enables us to prune significantly the number of relevant information sources and therefore reduce the cost of planning for the rest of the query.

The key issue in effectively interleaving planning and execution is to decide when to stop planning and start executing. An important advantage of our query-planning algorithm is that it provides a *natural* criterion for making this decision. Recall that in the first step of our query-planning algorithm we determine which information sources are relevant to each subgoal in isolation. The result of this step provides a clear indication of the causes of a large space of conjunctive plans. Specifically, subgoals in the query for which there is a large number of relevant sources will cause the number of combinations checked in the second step of the algorithm to grow significantly. Hence, it is better to postpone the planning for those subgoals until binding from others are obtained.

Conclusions and related work

The problem of providing a uniform interface to multiple information sources has received considerable attention in both the AI and Database literature. The Information Manifold is distinguished in that it combines novel methods from both fields to solve this problem. It provides an expressive language to describe the contents of information sources declaratively, and express the fine-grained distinctions between their contents. The query-planning algorithm is based on a novel method for answering queries using materialized database views, and guarantees that only *relevant* information sources are accessed. The algorithm also enables interleaving planning and execution and exploiting run-time bindings to further prune the relevant sources. The Information Manifold is a fully implemented system and currently provides access to 100 information sources on the WWW.

An important aspect of our system that has not been described here is the modeling of source *capabilities*, i.e., the kinds of queries that the source can answer about its contents. For example, a bibliography source may require that either the name of the author or the title be given as input. After generating query plans, the query processor searches for an ordering of the subgoals such that the source capability restrictions are satisfied. The problem of answering queries using views, when the views can be queried with restricted binding patterns, was first considered in [Rajaraman *et al.*, 1995]. In [Kwok and Weld, 1996] it is shown that if source capabilities are considered and we restrict ourselves to conjunctive plans, then there is no bound on the size of the plans that need to be considered, if we want to guarantee that all answers to the query are obtained. They also propose methods for pruning the space of plans that need to be considered in that case.

In the database community *multidatabase* systems also have the goal of providing uniform access to multiple information sources [Litwin *et al.*, 1990]. In these systems the correspondence between the contents of a source and the world-view is more direct. As one example, in the Pegasus system [Ahmed *et al.*, 1991] every source is modeled as a class in a class hierarchy, which is disjoint from other classes. It is therefore not possible to express relationships between sources by means of constraints, and the relationships need to be wired in to the schema. It should be noted that some of these restrictions are imposed because multidatabase systems also intended also to enable users to *update* the sources, and therefore they require source descriptions that enable to maintain consistency, and are concerned with concurrency control and transaction processing issues.

More recently, several systems for integrating multiple information sources are being built on the notion of a *mediator* (e.g., TSIMMIS [Chawathe *et al.*, 1994], HERMES [Subrahmanian *et al.*, 1995], CARNOT [Collet *et al.*, 1991], Nomenclator [Ordille and Miller, 1993], DISCO [Florescu *et al.*, 1995]). Broadly, these systems choose a set of queries, and provide for each one a procedure to answer the query using the available sources. Given a new query, their algorithms answer it by trying to relate it to existing queries. In Information Manifold, since the sources are described independently of the queries they will be used for, we are not restricted by which queries can be answered by the system, and it is easier to add or delete sources because we do not have to modify the query-specific procedures.

In the AI community several systems have been developed based on explicit representations of the contents of information sources (e.g., Internet Softbot [Etzioni and Weld, 1994], SIMS [Arens *et al.*, 1996], InfoMaster [Geddis *et al.*, 1995]). SIMS requires every information source to map to a class in the description logic system LOOM [MacGregor, 1988]. However, instead of taking advantage of the reasoning services of LOOM, SIMS selects the relevant sources using a set of transformation rules. These rules correspond to sound inferences that would be made in the CARIN language underlying the Information Manifold. The transformation rules in SIMS do not guarantee that a query-answering plan will be found if one exists. The Information Manifold provides a more expressive language than SIMS, allowing both a description logic and relations of arbitrary arity that are needed to model relational-database sources. The ability to describe sources by *arbitrary* conjunctive queries provides more flexibility than in SIMS. For example, SIMS cannot express that an information source contains a *join* of two world-view relations. Most importantly, the Information Manifold relies on the reasoning services of the underlying representation language. Therefore we do not have to mimic inferences in the query planner (perhaps less efficiently), and can immediately benefit from extensions provided by the knowledge representation system.

A formalism based on description logics for representing information sources is described in [Catarci and Lenzerini, 1993]. They use subsumption-style reasoning to determine which sources are relevant to a given query. They consider queries that ask for members of a class (and not arbitrary conjunctive queries). Therefore, they can determine relevance of sources solely based on subsumption. A preliminary version of our formalism for describing contents of sources was described in [Levy *et al.*, 1995b], but algorithms for answering queries for this formalism were not described. Context logics have also been proposed for modeling contents of information sources [Farquhar *et al.*, 1995], but designing algorithms for determining relevance of sources has not been addressed. Finally, [Etzioni *et al.*, 1994] describes an elegant formalism and algorithms for representing that a source has *complete* information of a certain kind, and shows that such information can be used to prune accesses to information sources. One direction in which we are currently extending the Information Manifold is to exploit such knowledge using techniques of rewriting queries using views.

Acknowledgements

The authors thank Chung Kwok, Marie-Christine Rousset and Dan Weld for comments on earlier versions of this paper.

References

Ahmed, R.; De Smedt, P.; Du, W.; Kent, W.; Ketabchi, M. A.; Litwin, W. A.; Rafii, A.; and Shan, M. 1991. The pegasus heterogeneous multidatabase system. *IEEE Computer* 19–26.

Arens, Y.; Knoblock, C.; and Shen, W. 1996. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems.*

Catarci, T. and Lenzerini, M. 1993. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems.*

Chawathe, S.; Garcia-Molina, H.; Hammer, J.; Ireland, K.; Papakonstantinou, Y.; Ullman, J.; and Widom, J. 1994. The TSIMMIS project: Integration of heterogeneous information sources. In proceedings of IPSJ, Japan.

Collet, C.; Huhns, M. N.; and Shen, W. 1991. Resource integration using a large knowledge base in carnot. *IEEE Computer* 55–62.

Etzioni, Oren and Weld, Dan 1994. A softbot-based interface to the internet. *CACM* 37(7):72–76.

Etzioni, Oren; Golden, Keith; and Weld, Daniel 1994. Tractable closed world reasoning with updates. In *Proceedings of KR-94.*

Farquhar, A.; Dappert, A.; Fikes, R. E.; and Pratt, W. 1995. Integrating information sources using context logic. In *AAAI Spring Symposium on Information Gathering from Heterogeneous Distributed Environments.*

Florescu, Daniela; Rashid, Louiqa; and Valduriez, Patrick 1995. Using heterogeneous equivalences for query rewriting in multidatabase systems. In *COOPIS '95.*

Geddis, D.; Genesereth, M.; Keller, A.; and Singh, N. 1995. Infomaster: a virtual information system. In *CIKM workshop in intelligent information agents.*

Golden, Keith; Etzioni, Oren; and Weld, Daniel 1994. Omnipotence without omniscience:sensor management in planning. In *Proceedings of AAAI-94.* 1048–1054.

Knoblock, Craig A. 1995. Planning executing, sensing and replanning for information gathering. In *Proceedings of IJCAI-95.*

Kwok, Chung T. and Weld, Daniel S. 1996. Planning to gather information. In *Proceedings of AAAI-96.*

Levy, Alon Y. and Rousset, Marie-Christine 1996. CARIN: a representation language integrating rules and description logics. In *Proceedings of ECAI-96.*

Levy, A. Y.; Mendelzon, A. O.; Sagiv, Y.; and Srivastava, D. 1995a. Answering queries using views. In *Proceedings of ACM PODS 1995.*

Levy, A. Y.; Srivastava, D.; and Kirk, T. 1995b. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems* 5 (2).

Levy, Alon Y.; Rajaraman, Anand; and Ullman, Jeffrey D. 1996. Answering queries using limited external processors. In *Proceedings of ACM PODS 1996.*

Litwin, Witold; Mark, Leo; and Roussopoulos, Nick 1990. Interoperability of multiple autonomous databases. *ACM Computing Surveys* 22 (3):267–293.

MacGregor, R. M. 1988. A deductive pattern matcher. In *Proceedings of AAAI-88.*

Ordille, J. J. and Miller, B. P. 1993. Distributed active catalogs and meta-data caching in descriptive name services. In *Proceedings of the 13th International IEEE Conference on Distributed Computing Systems.*

Rajaraman, Anand; Sagiv, Yehoshua; and Ullman, Jeffrey D. 1995. Answering queries using templates with binding patterns. In *Proceedings of ACM PODS 1995.*

Subrahmanian, V.S.; Adali, S.; Brink, A.; Emery, R.; Lu, J.; Rajput, A.; Rogers, T.; Ross, R.; and Ward, C. 1995. HERMES: A heterogeneous reasoning and mediator system. Technical report, University of Maryland.

Ullman, Jeffrey D. 1989. *Principles of Database and Knowledge-base Systems, Volumes I, II.* Computer Science Press, Rockville MD.