Towards Efficient Information Gathering Agents

Alon Y. Levy
AT&T Bell Laboratories
AI Principles Research Department
Murray Hill, NJ, USA.
levy@research.att.com

Yehoshua Sagiv
Department of Computer Science
Hebrew University
Jerusalem, Israel.
sagiv@cs.huji.ac.il

Divesh Srivastava
AT&T Bell Laboratories
AI Principles Research Department
Murray Hill, NJ, USA.
divesh@research.att.com

Abstract

Information gathering agents are required in many software agent applications to answer queries, posed by other agents, using a variety of available information sources. We formally consider the problem of designing information gathering agents, and make two important contributions. First, we examine the key issue of integrating knowledge from external sites into our knowledge base, and present an expressive language for this purpose. A noteworthy feature of our language is its ability to capture the knowledge that some external sites have complete information of a certain kind, using rich semantic constraints. Given a query on the knowledge base, it is important for the agent to first determine the set of external sites that contain information relevant to answering the query, and then access those sites. Our second contribution is to show that, given a query and the descriptions of the external sites in our language, it is possible to determine minimal subsets of sites that are needed to answer the query.

1 Introduction

The task of an information gathering agent is to answer queries posed by other agents (machines or humans) using a variety of available information sources. The functionality of such agents is required in many agent applications (e.g., [11; 12; 20]) and is assumed to be a desired feature in agent architectures and formalisms. For example, an agent whose task is to devise a meeting schedule for a visitor may need to consult several information sources in order to find additional constraints on the visitor's schedule (e.g., check the conference room availability, check the organizational chart to determine priorities in assigning meeting times). The current availability of a large number of information sources make information gathering a prime challenge in itself. For example, we are witnessing an explosion in the amount of information that is available over the Internet, such as technical papers, public domain software, and various databases (e.g., airline schedules, stock market listings). Providing easy access to such information is a key challenge for information gathering agents.

This paper addresses one of the important issues in building information gathering agents, namely the problem of integrating knowledge from external sources into the representation of our domain. A key issue in integrating external knowledge is designing a *site description language* for describing the contents of external sites and their relationship to our conceptualization of the domain. We present an expressive site description language, $\mathcal{SL}_1$, that has several important features that are desirable of such a language. First, $\mathcal{SL}_1$ is able to capture complex relationships between our conceptualization of the domain and the contents of external sites. In particular, it is able to capture the knowledge that some site has complete information of a certain type. Second, given a query and the descriptions of the sites, it is possible to find a minimal set of sites (or rather, subsets of sites) needed to answer the query. Since there are likely to be many external information sites with significant access costs, finding such a minimal set of sites is essential for efficient information gathering.

The problem of information gathering agents has also been considered in the SIMS project [1]. Our work presents a formal framework and analysis of information gathering agents. As such, it sheds light on the SIMS work and shows formally how the representation and reasoning used in SIMS can be significantly extended. A more detailed comparison is presented in Section 5.

The problem of integrating multiple information sources has also been considered in work on multidatabase systems (e.g., [15]). However, the results obtained in that area have limited applicability for our purposes for two reasons. First, although multidatabases attempt to integrate multiple information sources, they have not addressed the issue of automatically finding the data relevant to a given query. It is usually assumed (e.g., in the query language MSQL [9]) that the location of the relevant data is implicit in the query itself. Second, the work in that area deals with representation languages that lack the expressive power needed in our application. For example, the representation language and the site description language that we use combine a KL-ONE based description language [4] and a rule-based language, both of which are not considered in the work on multidatabase systems. Our work can also be viewed

as extending semantic query optimization techniques to a more general setting.

2 Information Gathering Agents

Queries are posed to information gathering agents in terms of the relations and objects that are present in a knowledge base shared between the agents (or with which a human is interacting). Part of the knowledge required to answer a query may exist locally in the knowledge base. However, an information gathering agent is expected to utilize information available at sites that are external to the knowledge base. In order to do so, the agent needs a precise description that relates the relations and objects in the knowledge base with the information available at the external sites. This description is given by statements in a *site description language*. To illustrate, we use the following example throughout the paper.

Example 2.1: Consider an application in which we can obtain information about airline flights from various travel agents. We have access to fares given by specific travel agents and to telephone directory information to obtain their phone numbers. In practice, the information about price quotes and telephone listings may be distributed across different external database servers which contain different portions of the information. For example, some travel agent may deal only with domestic travel, another may deal with certain airlines. Some travel brokers deal only with last minute reservations, e.g., flights originating in the next one week. Similarly, directory information may be distributed by area code. In some area codes, all listings may be in one database, while others may partition residential and business customers. ∎

As a basis for our discussion, we describe the representation language used in our knowledge base. This language combines a description language from the KL-ONE family [4] (a.k.a. terminological logics) with general n-ary relations.

A description language consists of three types of entities: concepts (representing unary relations), roles (binary relations) and individuals (object constants). Concepts can be defined in terms of *descriptions* that specify the properties that individuals must satisfy to belong to the concept. Binary relationships between objects are referred to as roles and are used to construct complex descriptions for defining concepts. Description logics vary by the type of constructors available in the language used to construct descriptions. Description logics are very convenient for representing and reasoning in domains with rich hierarchical structure. A variety of such languages have been considered in the literature (e.g., CLASSIC [3], LOOM [16], ACLNR [5]). We do not limit ourselves to a specific description language, but require that the question of subsumption (i.e., does a description D_1 always contain a description D_2) be decidable. We denote the concepts in our representation language by $\mathcal{D} = D_1, \ldots, D_l$.

In our example, we can have a hierarchy of concepts describing various types of telephone customers. The concept *customer* is a primitive concept that includes all customers and specifically the disjoint subconcepts *Business* and *Residential*. Each instance of a business customer has a role *BusinessType*, specifying the types of business it performs. Given these primitive concepts, we can define a concept *TravelAgent* by the description:[1]

(AND Business (fills BusinessType "Travel")).

One limitation of description languages is that they do not naturally model general n-ary relations. Such relations arise very commonly in practice and dealing with such relations is essential to modeling external information sources that contain arbitrary relational databases. Hence our representation language augments description languages with a set of general n-ary relations $\mathcal{E} = E_1, \ldots, E_n$. It should be emphasized that the general n-ary relations are *not* part of the description language. Hereafter, we refer to the set of relations $\mathcal{E} \cup \mathcal{D}$ as the *KB relations*, to distinguish them from relations stored at external sites. Our application domain is naturally conceptualized by the following two relations:

- $Quote(ag, al, src, dest, c, d)$, denotes that a travel agent ag quoted a price of c to travel from src to $dest$ on airline al on date d.
- $Dir(cust, ac, telNo)$, gives the directory listing of customer $cust$ as area code ac and phone number $telNo$.

A key aspect of our representation language is the ability to capture rich semantic structure using *constraints*, with which we can reason efficiently. An *atomic constraint* is an atom either of the form $D(x)$, where D is some concept in $\mathcal{D}$, and x is a variable, or $(x_i\ \theta\ x_j)$ (or $(x_i\ \theta\ a)$) where x_i and x_j are variables, a is a constant and $\theta \in \{>, \geq, <, \leq, =, \neq\}$. Arbitrary constraints are formed from atomic constraints using logical operators $\wedge$ and $\vee$. We can determine efficiently whether one constraint entails another using subsumption reasoning in the description logic and implication reasoning of order constraints [23].[2] Constraints play a major role in information gathering and are used in several ways. First, semantic knowledge about the general n-ary relations $\mathcal{E}$ can be expressed by constraints over the arguments of the relations. In our example, we can specify that the first argument of the relation *Quote* must be an instance of the concept *TravelAgent*. Second, as we discuss in subsequent sections, constraints can be used to specify subsets of information that exist at external sites. For example, a travel agent may have only flights whose cost is less than $1000. Finally, as we see below, constraints are extremely useful in specifying complex queries.

Some of the extensions of the knowledge base relations may exist in the knowledge base, however most of them will not. Given a query (defined formally below), the agent must infer the missing portions of the extensions of relations needed to answer the query, using the information present at the external sites. For the purpose of

[1] Our syntax follows that used in CLASSIC [3].

[2] We can allow atomic constraints from any domain where implication/subsumption reasoning can be done efficiently.

our discussion, the knowledge base can also be viewed as an information source containing part of the extensions.

The query language that we use in our discussion combines the use of concepts from description logics and Horn rules, as described in [8]. A query is essentially a relation defined in terms of a set of Horn rules, using the relations $\mathcal{E}$, intermediate relations $\mathcal{I}$, and constraints. Each rule is of the form

$$L_1(\bar{x}_1) \wedge \ldots \wedge L_n(\bar{x}_n) \Rightarrow L(\bar{x}).$$

where each L_i is either a constraint or an atom of a relation in $\mathcal{E} \cup \mathcal{I}$, and L is a relation in $\mathcal{I}$.[3] For example, we might be interested in the following query that retrieves phone numbers of travel agents in New York City who sell tickets from NYC to Paris for under $500, on Air France:

$$Quote(name, AirFrance, NYC, Paris, c, d) \wedge Dir(name, ac, telNo) \wedge (c \leq \$500) \wedge (ac = 212) \Rightarrow Answer(telNo).$$

3 The Site Description Language $\mathcal{SL}_0$

Answering queries that use information stored at external sites requires a precise description of the information available at each site and its relationship to our conceptualization of the domain (i.e., the relations and objects in our knowledge base). This description, given in a site description language, should enable us to determine external sites that contain information relevant to a given query. Since the number of sites is likely to be very large and the cost of accessing them will be significant, it is important that the agent be able to find a minimal set of relevant sites (or portions of sites). In this section we describe a language for this purpose, $\mathcal{SL}_0$, which is both expressive and enables efficient determination of a minimal set of relevant sites.

In our discussion we assume that the external sites contain extensions of relations, denoted by $\mathcal{R} = R_1, \ldots, R_m$.[4] Note that a site need not explicitly store a certain relation, but only be able to compute it effectively when queried for it. Formally, $\mathcal{SL}_0$ is the language of Horn rules containing relation names from $\mathcal{R}$, $\mathcal{E}$ and $\mathcal{D}$, and constraints. The site relations $\mathcal{R}$ appear only in the antecedents of the rules. Intuitively, a set of rules in $\mathcal{SL}_0$ enables us to compute extensions of the KB relations in $\mathcal{E}$ and $\mathcal{D}$, given the relations in $\mathcal{R}$. Given a set of rules, we add the formulas required by the predicate completion [6] of the predicates in $\mathcal{E}$ (but not $\mathcal{R}$!). Intuitively, this means that extensions of these relations can be computed *only* using the given rules.[5] In our example, we can describe the following two travel agents:

$$Agent0DB(src, dest, cost, date) \Rightarrow Quote(Agent0, United, src, dest, cost, date).$$

$$Agent1DB(al, dest, cost, date) \Rightarrow Quote(Agent1, al, NYC, dest, cost, date).$$

$$Agent1DB(al, dest, cost, date) \Rightarrow (cost \leq \$1000).$$

Agent0 is an agent that sells tickets only on United Airlines. Agent1 specializes in cheap deals out of New York City. Note that the third rule specifies a constraint on the information in Agent1DB, and is allowed in $\mathcal{SL}_0$. We also have three sites containing directory information:

$$212Residential(name, telNo) \Rightarrow Dir(name, 212, telNo).$$

$$212Residential(name, telNo) \Rightarrow Residential(name).$$

$$212Business(name, telNo) \Rightarrow Dir(name, 212, telNo).$$

$$212Business(name, telNo) \Rightarrow Business(name).$$

$$800Dir(name, telNo) \Rightarrow Dir(name, 800, telNo).$$

The site 212Residential (212Business) contains the residential (business) customers in the 212 area code. The site 800Dir contains all the toll-free numbers.

Answering a specific query first requires that the agent determine which sites contain information that can be used to answer the query. Having found the set of relevant sites, the next problem is to devise an optimal plan for accessing the relevant sites. After computing the relevant extensions of the KB relations, the agent can compute the answer to the query. This paper focuses on the problem of determining the relevant sites, as we formally define below. Clearly, we can always consider all the site relations and get all the relevant information. However, the challenge is to determine the *minimal* portions of the site relations that are relevant to answering the query.[6]

Definition 3.1: Given a query q and a set of rules Δ in $\mathcal{SL}_0$, a subset $R \wedge C$ of a site relation R, where C is a constraint, is said to be the *minimal portion* of R relevant to q, if:

- For any fact ϕ of R that satisfies C, there is some instantiation of the site relations that is consistent with Δ, such that ϕ is used in a derivation of an answer to q, and
- There is no instantiation of the sites that is consistent with Δ, and such that a fact of R that doesn't satisfy C is used in a derivation of q. ∎

In our example, since we are looking for a flight on AirFrance, Agent0 will be deemed irrelevant, and therefore, Agent0DB will be ignored. Similarly, the 800 directory listing database will not be queried. Moreover, since the first argument of *Quote* must be an instance of the concept *TravelAgent*, and since *TravelAgent* is subsumed by *Business*, which is disjoint from *Residential*, only the 212 business directory listing will be considered for the query.

The following theorem shows that it is possible to precisely determine the minimal portions of the site relations that are relevant to a given query.

[3] Note that the predicates in $\mathcal{I}$ (and therefore, the query predicate) may be recursively defined.

[4] Recall that concepts are unary relations and therefore, a site can contain the instances of a concept.

[5] Note that if we want to model incomplete information about some of the relations in $\mathcal{E}$, then we can remove the predicate completion axioms for those relations.

[6] Clearly, for some site relations the minimal portion can be empty, indicating that the site relation does not contain any relevant information.

Theorem 3.2: *Let Δ be a set of rules in $\mathcal{SL}_0$ describing the relationship between the KB relations $\mathcal{E}$ and $\mathcal{D}$ and the external site relations $\mathcal{R}$. Let q be a query. For each relation $R_i \in \mathcal{R}$, it is possible to determine a constraint C_i such that $R_i \wedge C_i$ is the minimal portion of R_i that is relevant to q.*

The key to proving the theorem is to show that the query-tree algorithm presented in [13] can be extended to solve this problem, since the language for expressing constraints (concept descriptions and order constraints) satisfies the requirements of the query-tree algorithm outlined in [13]. Finding minimal portions of the sites is done in two steps. The first step determines which portions of the knowledge base relations are needed to solve the query, and the second step determines which portions of the site relations are needed to compute the relevant portions of the knowledge base relations. Because of space limitations, the full proof is omitted here.

It should be realized that the problem of finding relevant sites is a crucial problem for information gathering agents. In addition to its importance for answering queries, it also forms a basic building block for other tasks of information agents. For example,

- Processing updates on the knowledge base requires updating relevant site relations and hence, determining the relevant sites.
- Efficiently monitoring queries over time requires determining precisely which external site relations should be monitored.
- Maintaining consistency among site relations again requires that we determine which sites contain information relevant to a given consistency condition.

4 The Site Description Language $\mathcal{SL}_1$

Although the site description language $\mathcal{SL}_0$ is quite expressive, it has some limitations. In practice, information may reside redundantly in many sites. Therefore, an important challenge is to find a set of site portions such that each portion is minimal and such that the overall set does not contain redundancies. This point is illustrated by the experiments described in [10] which show that reducing redundant information gathering leads to significant savings.

A major limitation of $\mathcal{SL}_0$ is that we cannot express knowledge about the sites that enables finding such a non-redundant set. For instance, suppose that in our example, the query was to find some flight from NYC to Paris on United Airlines. There is no way in $\mathcal{SL}_0$ to express the fact that Agent0 contains *all* the information about flights on United. Had we been able to express this knowledge, we could have determined that Agent0 by itself would suffice for answering this query, and hence, that knowledge of Agent1 would have been redundant.

The reason we cannot express this knowledge is that in $\mathcal{SL}_0$, the site relations cannot appear in the consequents of the rules. A natural extension of $\mathcal{SL}_0$ would be to relax that restriction. However, allowing such rules arbitrarily raises several problems. For example, a rule of the form:

$$r_1 : E(\bar{x}) \wedge C(\bar{x}) \Rightarrow R(\bar{x})$$

where E is a KB relation, C is a constraint and R is a site relation, tells us that all facts of E that satisfy C can be found in R. However, the above rule by itself does not tell us how to *compute* the extension of E given the facts in R, because given a fact in R, r_1 does not imply that it is a fact in E.

A possible solution to this problem is to add the predicate completion axioms of R. However, that leads to two problems. For example, consider the rules:

$$r_2 : E_1(x) \Rightarrow R(x).$$
$$r_3 : E_2(x) \Rightarrow R(x).$$

Predicate completion of R will add the formula $R(x) \Rightarrow E_1(x) \vee E_2(x)$. Given a fact in R (which is the external site relation) we cannot determine whether it is in the extension of E_1 or E_2, but only that it is in their union. Consequently, facts in R can be used *only* in queries that require the union of E_1 and E_2. Answering arbitrary queries involving E_1 and E_2 becomes much harder because the inference engine needs to reason with disjunctive information. Another problem of predicate completion is illustrated by the following rules:

$$E(x) \wedge (x < 1) \Rightarrow R(x).$$
$$R(x) \wedge (x < 2) \Rightarrow E(x).$$

These rules state that facts in R for which $x < 2$ are in E, and that R is known to have *all* the facts of E for which $x < 1$. Note that the second rule is expressed in $\mathcal{SL}_0$. The predicate completion of R will entail that R contains *only* facts for which $x < 1$, which is clearly not an intuitive consequence of these rules.[7]

In the extended language $\mathcal{SL}_1$ which we describe below, we add additional rules which are weaker than the predicate completion axioms, and enable us to compute the KB relations from the site relations. The language $\mathcal{SL}_1$ contains the formulas in $\mathcal{SL}_0$ and restricted forms of Horn rules in which the site relations appear in the consequents. Formally, except for rules in $\mathcal{SL}_0$, knowledge about the relation R is specified by a set of rules of the form:

$$P_1(\bar{x}) \wedge C_1 \Rightarrow R(\bar{x}).$$
$$\vdots$$
$$P_k(\bar{x}) \wedge C_k \Rightarrow R(\bar{x}).$$

where P_i is a relation in $\mathcal{E}$ or $\mathcal{D}$, and C_i is a constraint. Furthermore, (for reasons we explain below) we restrict the rules such that the constraints are pairwise mutually exclusive, i.e., for each pair, $1 \leq i < j \leq k$, it must be the case that $C_i \wedge C_j$ is unsatisfiable. These rules specify that R contains complete information about the facts of P_i that satisfy C_i. Given the rules above, we add the following rules that enable us to use R to compute the extensions of the KB relations:

$$R(\bar{x}) \wedge C_1 \Rightarrow P_1(\bar{x}).$$
$$\vdots$$
$$R(\bar{x}) \wedge C_k \Rightarrow P_k(\bar{x}).$$

[7] An alternative solution is to assume that rules for computing $\mathcal{E}$ and $\mathcal{D}$ are *explicitly* specified, using $\mathcal{SL}_0$ rules. Therefore, non-$\mathcal{SL}_0$ rules are used only to specify completeness information.

Intuitively, this means that if we stated that R contains all the facts of P_i that satisfy C_i, we take that also to mean that any fact in R that satisfies C_i indeed belongs to P_i. Since the C_i's are pairwise mutually exclusive, we obtain intuitive results.[8]

The language $\mathcal{SL}_1$ has several important properties. First, it enables us to express knowledge stating that a certain site contains complete information of a certain type. Second, given arbitrary formulas in $\mathcal{SL}_1$ and a set of site relations, it is possible to uniquely determine the extensions of the KB relations. Finally, given a query, we show that an agent can find a *non-redundant site set* for computing the answer to the query. That is, the agent can find a set of minimal portions of sites such that there are no redundancies among these portions. This notion is made formal as follows:

A *site set* is a set $\{(R_1, C_1), \ldots, (R_n, C_n)\}$, denoting the set of facts of the site relations R_i satisfying the constraints C_i.

Definition 4.1: Given a query q, and a set of rules Δ in $\mathcal{SL}_1$, a site set $S = \{(R_1, C_1), \ldots, (R_n, C_n)\}$ is said to be a non-redundant site set w.r.t. q if:

- For each fact $\phi \in S$ there is some instantiation of the site relations that satisfies Δ, such that ϕ is part of a derivation of an answer to the query,
- Each superset of S will yield the same answer to the query as S, and
- There is no subset of S that is a non-redundant site set w.r.t. q. ∎

Note that this definition is stronger than Definition 3.1. There we only required that the portions of a relation be minimal in the sense that each fact in it can be used in *some* derivation of the query. Definition 4.1 considers the whole site set and requires that none of its facts be redundant. Therefore, facts that are replicated in multiple sites will be retrieved from at most one external site.

In our example, if we have a server with the listing of *all* travel agents in the greater NYC area (area codes 212 and 718), we can specify its contents as follows:

$$Dir(name, ac, telNo) \wedge TravelAgent(name) \wedge ((ac = 212) \vee (ac = 718)) \Rightarrow NYTravel(name, ac, telNo).$$

Given our query that restricts travel agents to the 212 area code, we can use this knowledge to determine that only the NYTravel directory listing is needed.

Allowing Specialized Sites

In the $\mathcal{SL}_1$ rules we described above, the site relations contain the same attributes of the KB relations. Often, a site relation will be specialized w.r.t. a KB relation and therefore will contain fewer attributes. For example, Agent0DB does not contain the name of the agent as an attribute, nor the name of the airline. To express completeness knowledge about such sites, we allow for rules of the following form with a few restrictions:

$$E(\bar{x}, \bar{y}) \wedge C \Rightarrow R(\bar{x}).$$

In this rule, the site relation R projects out the arguments $\bar{y}$ of E. We impose restrictions on the rule that guarantee that the variables $\bar{y}$ can be determined uniquely, given $\bar{x}$. One simple restriction is to require that all elements of $\bar{y}$ be constants. Another possibility is to allow the rule to specify linear functions of $\bar{x}$ that uniquely determine $\bar{y}$. Adding the restricted completion axioms can be done as before.

As an example of using such rules, we can specify that Agent0 has *all* the flights of United Airlines:

$$Quote(Agent0, United, src, dest, cost, date) \Rightarrow Agent0DB(src, dest, cost, date).$$

It can be shown that Theorem 3.2 holds also for the site description language $\mathcal{SL}_1$. This means that given a query, it is still possible to determine the minimal portions of each site relation that are needed to answer the query. However, recall that $\mathcal{SL}_1$ was designed to express additional knowledge which would enable the agent to find a non-redundant site set for answering a given query (which is a stronger notion than minimal site portions). The following theorem specifies the exact conditions under which it is possible to do so.

Theorem 4.2: *Let Δ be a set of rules in $\mathcal{SL}_1$, describing the relationship between the site relations $\mathcal{R}$ and our knowledge base relations $\mathcal{E}$ and $\mathcal{D}$. Given a query q,*

- *In general, determining whether a site set S is a non-redundant site set w.r.t. q is undecidable.*
- *If the rules in Δ and in the query q are non-recursive and the description language used for defining concepts contains union and negation (and subsumption is still decidable), then it is possible to find a non-redundant site set w.r.t. q.*

The first half of the theorem follows by a reduction from the equivalence problem of Datalog programs, which is known to be undecidable [18]. The algorithm for finding a non-redundant site set in the non-recursive case proceeds in two steps. In the first, we find a non-redundant subset of the KB relations needed to solve the query, extending the algorithm described in [14]. In the second, we use the rules in Δ that specify completeness information about the external sites to find a minimal non-redundant covering set of the site relations that contain the required KB relations.

5 Conclusions and Related Work

This paper formally investigated the problem of designing an information gathering agent. In order for an information agent to perform effectively, the relationship between the relations in the knowledge base and the external sites must be described in a formal language with well defined semantics. We identified and focused on the crucial problem of determining which minimal set of external sites are needed to answer a given query. We presented an expressive site description language $\mathcal{SL}_1$ that can be used to capture complex relationships between the KB relations and the external sites. A noteworthy feature of the language is its ability to express completeness information about certain sites. We have also shown

[8] Note that if the constraints were not pairwise exclusive, as in rules r_2 and r_3, we would infer that some facts are in the intersection of the KB relations.

that it is possible to determine a minimal set of sites that are needed to answer the query. An interesting aspect of our analysis is the use of a rich constraint language, incorporating concept descriptions and order constraints, which are likely to be very useful in this domain.

Integration of knowledge bases and databases has also been considered in [17; 7; 1]. The work most closely related to ours is described in the SIMS project for integrating multiple information sources [1; 2]. The representation of the domain in their system is based on a description logic (LOOM [16]). Answering a query proceeds in two steps: finding the relevant sites and accessing them. In SIMS, both components of the problem are posed as planning problems. One important result of our formal analysis of the problem is showing that the first step, finding the set of relevant sites, need not be posed as a planning problem and can be done efficiently using our methods. However, the second task, accessing the relevant sites, can possibly benefit by being posed as a planning problem. Our representation and site description languages are considerably more expressive than those used in SIMS. For example, SIMS does not allow arbitrary n-ary relations, and therefore, they have to map each external relation to a concept in LOOM. This can be done only when the relation has a primary key (i.e., an attribute that uniquely determines the rest of the attributes of the tuple). Although one can always conceptually add another such attribute to a relation, modeling a relation in such a way is unnatural.

Several agent languages have been considered in the literature [19; 20; 21; 22]. The main focus of previous work is on providing languages for formalizing agent behavior, such as goals, intentions or capabilities. In contrast, our work deals specifically with the task of information gathering and the special issues that arise in this context. The problem of information gathering has also been considered as a problem for planning with incomplete information [11], focusing on the formalization of the interactions between actions that affect the state of the world and those that affect the state of knowledge of the planner. Etzioni et al. [10] deal with the issue of tractable reasoning about some kinds of complete information in the context of planning. They present a language in which they can infer efficiently complete information about certain relations, given that they have complete information about other relations. An interesting question is to extend their techniques to deal explicitly with constraint information, to make it suitable for the language used in our work.

This paper lays the foundations for exploring several avenues of research. First, extending the representation and site description languages to capture other kinds of information should be explored. In particular, a tighter integration between the description logic and the n-ary relations and rules is desirable. Richer models of sites are also of interest. For example, a site can contain an arbitrary knowledge base, not merely relations. A site can also have an associated access cost, that should be considered in determining which sites to access. Finally, given the set of relevant sites, formulating an efficient query evaluation plan poses many interesting challenges.

References

[1] Yigal Arens, Chin Y. Chee, Chun nan Hsu, and Craig A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems*, 1994.

[2] Yigal Arens and Craig A. Knoblock. Planning and reformulating queries for semantically-modeled multidatabase systems. In *Proceedings of the First International Conference on Information and Knowledge Management, Baltimore, MD*, 1992.

[3] Alex Borgida, Ronald Brachman, Deborah McGuinness, and Lori Resnick. CLASSIC: A structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 59–67, 1989.

[4] Ronald J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[5] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[6] K. Clark. Negation as failure. In J. Minker and H. Gallaire, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[7] Christine Collet, Michael N. Huhns, and Wei-Min Shen. Resource integration sing a large knowledge base in carnot. *IEEE Computer*, pages 55–62, 1991.

[8] Francesco Donini, M. Lenzerini, D. Nardi, and A. Schaerf. A hybrid system integrating datalog and concept languages. In *Working notes of the AAAI Fall Symposium on Principles of Hybrid Reasoning*, 1991.

[9] W. Litwin et al. MSQL: A multidatabase language. *Information Sciences*, 49(1–3):59–101, October–December 1990.

[10] Oren Etzioni, Keith Golden, and Daniel Weld. Tractable closed world reasoning with updates. In *Proceedings of KR-94. To appear.*, 1994.

[11] Oren Etzioni, Steve Hanks, Daniel Weld, Denise Draper, Neal Lesh, and Mike Williamson. An approach to planning with incomplete information. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 115–125, 1992.

[12] Michael R. Genesereth. An agent-based framework for software interoperability. In *Proceedings of the Software Technology Conference, Los Angeles, CA*, 1992.

[13] Alon Y. Levy and Yehoshua Sagiv. Constraints and redundancy in Datalog. In *The Proceedings of the Eleventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Diego, CA.*, pages 67–80, 1992.

[14] Alon Y. Levy and Yehoshua Sagiv. Queries independent of updates. In *Proceedings of the 19th VLDB Conference, Dublin, Ireland*, pages 171–181, 1993.

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

[16] MacGregor R. M. A deductive pattern matcher. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 403–408, 1987.

[17] Donald P. McKay, Timothy W. Finin, and Anthony O'Hare. The intelligent database interface: Integrating ai and database systems. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 1990.

[18] Oded Shmueli. Decidability and expressiveness aspects of logic queries. In *Proceedings of the 6th ACM Symposium on Principles of Database Systems*, pages 237–249, 1987.

[19] Yoav Shoham. AGENT0: a simple agent language and its interpreter. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 704–709, 1991.

[20] Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.

[21] Munindar P. Singh. Towards a formal theory of communication for multiagent systems. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 69–74, 1991.

[22] Becky Thomas. *PLACA, An Agent Oriented Programming Language.* PhD thesis, Stanford University, Stanford, CA, 1993.

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