# Knowledge Discovery Objects and Queries in Distributed Knowledge Systems

Zbigniew W. Raś and Jiyun Zheng

University of North Carolina, Dept. of Comp. Science, Charlotte, N.C. 28223, USA
email: ras@uncc.edu or jzheng@uncc.edu

**Abstract.** The development of many knowledge discovery methods (see [14], [7], [16]) provided us with good foundations to build a kd-Query Answering System ($kdQAS$) for Distributed Knowledge Systems ($DKS$). By $DKS$ we mean a number of autonomous processing elements (called knowledge systems) that are interconnected by a computer network and that cooperate in their assigned tasks. A knowledge-system we see as a relational database coupled with a discovery layer which is simplified in this paper to a set of rules.

Queries handled by $kdQAS$ are more general than SQL. Also, the queried objects are far more complex than tuples in a relational database. To distinguish them from objects and queries in DBMS, we introduce kd-objects and kd-queries respectively. In general, by $kd-object$ we mean any set of tuples and rules. By $kd-query$ we mean a predicate which queries kd-object in $DKS$ and returns another kd-object for an answer. Our kd-objects may not exist a priori, thus querying them at one site of $DKS$ may require generation, at run time, of new kd-objects either at the same site or at other sites of $DKS$. So, querying has to major roles: generation of new kd-objects and retrieval of the ones which were generated before.

In relational databases, the result of a query is a relation that can be queried further. This is typically referred to as a closure principle, and it should be one of the most important design principles for $kdQAS$. Our kd-queries satisfy such a closure principle.

**Key Words:** incomplete information system, cooperative query answering, rough sets, multi-agent system, knowledge discovery.

## 1 Introduction

In many research fields, such as military, medical, manufacturing, and educational, similar databases are kept at many sites. Each database stores information about local events and the information is expressed in attributes compatible among databases. When similar databases are designed, their events are

described in terms of the same attributes with some minor or major exceptions. Values of the same attribute may have different generality among databases. The procedures used to collect the data do not have to be the same among databases which means their operational semantics can be different. Also, some attributes might be missing in one database but they occur in many others. Missing attributes lead to problems. Medical doctor may query a database in his hospital to find all patients having certain symptoms, only to realize that one component of that description is a result of a medical test which is missing in the database schema and the same the query cannot be answered. The same query would work on many other databases but the doctor is interested in identifying the patients in his hospital. In this paper we develop a theory for intelligent answering these "locally unreachable" queries.

The task of integrating independently built databases is complicated not only by the differences between the data contents but also by the differences in structure and semantics of the information they contain. The problem is exacerbated when one needs to provide access to such a system for the end-users. For more than 10 years research has been devoted to the question of information retrieval from heterogonous distributed databases. This research has sought to provide integrated access to such databases and has focused on distributed databases, multidatabases, federated databases and their interoperability. The main purpose of integrated access is to enable a number of heterogeneous distributed databases to be queried as if they were a single homogeneous database. Common practice in integrating database systems involves manual integration of each database schema into a global schema [1]. This approach does not work when the number of database systems is large. Navathe and Donahoo [9] propose to allow the database designers to develop a metadata description of their database schema. The collection of metadata descriptions can be automatically processed by a schema builder to create a partially integrated global schema. The heterogeneity problem can be eliminated (see [8]) by using an intermediate model that controls the knowledge translation from a source database or knowledgebase. The intermediate model they developed is based on the concept of abstract knowledge representation. It has two components: a modeling behavior which separates the knowledge from its implementation, and a performative behavior which establishes context abstraction rules over the knowledge. In this paper we propose to handle the heterogeneity problem among independently built databases through the use of discovery layers.

First, we introduce the notion of a Distributed Information System ($DIS$) which is the main vehicle for development of a Distributed Knowledge System ($DKS$). Basically to transform $DIS$ to $DKS$ we need to add a discovery layer to every site of $DIS$. Discovery layers are homogeneous and for simplicity reason they are simplified in this paper to sets of rules. The content of discovery layers may constantly change because by querying one site of $DKS$, its other sites can be asked for help to answer the query. Rules in this paper are seen as operational definitions providing a commonly sharable information among independently built information systems. So, the transfer of rules from one site of $DKS$ to

another site does not cause much problems.

All distributed DBMSs support deduction of information. Our goal is to build a knowledge discovery based Query Answering System ($kdQAS$) for each site of $DKS$ which satisfies the closure principle (the response for a query can be queried further). The access to $kdQAS$ is through WWW.

Predicate logic is the vehicle chosen to represent knowledge in $DKS$ and queries in $kdQAS$. Many other representations are, of course, possible. We have chosen predicate logic because of the need to manipulate queries and rules syntactically without changing their semantical meaning. This syntactical manipulation of queries will be handled by $kdQAS$. By designing an axiomatic system which is sound and complete we are certain that queries we manipulate will not change their semantical meaning. Clearly, this property is very much needed. Without it, we may be looking for an answer to queries which are semantically different from the queries asked by the user. Such a situation has to be avoided.

## 2 Distributed Information Systems

In this section, we introduce the notion of a Distributed Information System ($DIS$) which is the main vehicle for development of a Distributed Knowledge System ($DKS$). Basically to transform $DIS$ to $DKS$ we need to add a discovery layer to every site of $DIS$. A discovery layer is simplified in our paper to the set of rules. Its content may constantly change because by querying $DKS$ we may discover rules at one site and store them at other sites of $DKS$.

In this paper, we consider two types of queries called local and global (locally unreachable). Global queries are queries which can be resolved only through the interaction of sites (exchanging knowledge between them) in $DKS$. Local queries are resolved entirely by a single site of $DKS$.

So, let us start with basic definitions.

By an information system $S$ we mean a sequence $(X, A, V, h)$, where $X$ is a finite set of objects, $A$ is a finite set of attributes, V is the set-theoretical union of domains of attributes from $A$, and $h$ is a classification function which describes objects in terms of their attribute values (see [12], [13]). We assume that:

- $V = \bigcup\{V_a : a \in A\}$ is finite,
- $V_a \cap V_b = \emptyset$ for any $a, b \in A$ such that $a \neq b$,
- $h : X \times A \longrightarrow V \cup 2^V$ where $h(x, a) \in V_a$ or $h(x, a) = V_a$ for any $x \in X$ and $a \in A$.

Attribute $a$ is called incomplete in $S$ if there is $x \in X$ such that $h(x, a) = V_a$. By $In(X, a)$ we mean the set $\{x \in X : h(x, a) = V_a\}$. We will be referring to this set when we give the definition of $kdQAS$.

Let $S_1 = (X_1, A_1, V_1, h_1)$, $S_2 = (X_2, A_2, V_2, h_2)$ be information systems. We say that $S_2$ is a subsystem of $S_1$ if $X_2 \subseteq X_1$, $A_2 \subseteq A_1$, $V_2 \subseteq V_1$ and $h_2 \subseteq h_1$.

By $h_2 \subseteq h_1$ we mean that either $h_2(x,a) = h_1(x,a)$ or [ $h_2(x,a) \in V_a$ and $h_1(x,a) = V_a$ ].

We use a table-representation of a classification function $h$ which is naturally identified with an information system $S = (X, A, V, h)$. For simplicity reason, instead of a set $V_a$ we place the blank symbol which is interpreted here as *all values in $V_a$ are possible*. For example, let $S = (X, A, V, h)$ is an information system where $X = \{a_1, a_6, a_8, a_9, a_{10}, a_{11}, a_{12}\}$, $A = \{C, D, E, F, G\}$ and $V = \{e_1, e_2, e_3, f_1, f_2, g_1, g_2, g_3, c_1, c_2, d_1, d_2\}$. Additionally, we assume here that $V_E = \{e_1, e_2, e_3\}$, $V_F = \{f_1, f_2\}$, $V_G = \{g_1, g_2, g_3\}$, $V_C = \{c_1, c_2\}$, and $V_D = \{d_1, d_2\}$. Then, the function $h$ defined by Table 1 is identified with information system $S$.

| $X_2$ | $F$ | $C$ | $D$ | $E$ | $G$ |
|---|---|---|---|---|---|
| $a1$ | $f1$ | $c1$ | $d2$ | $e2$ | $g1$ |
| $a6$ | $f2$ | | $d2$ | $e3$ | $g2$ |
| $a8$ | $f1$ | $c2$ | | | $g1$ |
| $a9$ | $f2$ | $c1$ | | | $g1$ |
| $a10$ | $f2$ | $c2$ | $d2$ | $e3$ | $g1$ |
| $a11$ | $f1$ | | $d1$ | $e3$ | $g2$ |
| $a12$ | $f1$ | $c1$ | $d1$ | $e3$ | $g1$ |

**Table 1.** Information System $S$

By a Distributed Information System [13] ($DIS$) we mean a pair $DS = (\{S_i\}_{i \in I}, L)$ where:

- $S_i = (X_i, A_i, V_i, h_i)$ is an information system for any $i \in I$,
- $L$ is a symmetric, binary relation on the set I,
- $I$ is a set of sites.

Systems $S_{i1}, S_{i2}$ (or sites $i1, i2$) are called neighbors in a distributed information system $DS$ if $(i1, i2) \in L$. The transitive closure of $L$ in $I$ is denoted by $L^+$.

A distributed information system $DS = (\{S_i\}_{i \in I}, L)$ is consistent if:

- $(\forall i)(\forall j)(\forall x \in X_i \cap X_j)(\forall a \in A_i \cap A_j)[(x,a) \in Dom(h_i) \cap Dom(h_j) \longrightarrow h_i(x,a) = h_j(x,a)]$.

We assume here that any site of $DIS$ can be queried either for objects or for knowledge. Knowledge in this paper is simplified to a set of rules. Syntactically, a query is built from values of attributes belonging to $V = \bigcup\{V_i : i \in I\}$. A query is called local for a site $i$, if it is built from values in $V_i$. Otherwise, it is called global (locally unreachable) for $i$. Both, local and global queries will be

handled by kd-Query Answering System ($kdQAS$). In order to resolve a global query at site $i$, a transfer of newly discovered knowledge at other sites of $DIS$ to a site $i$ will be needed. This knowledge is stored in discovery layer of site $i$. If a queried information system $S$ in $DIS$ is incomplete, then a new query has to be invoked and answered first. To be more precise, system $S$ is queried first for certain consistent rules to be discovered locally at $S$ and at its remote sites. Next, this newly discovered set of consistent rules is treated as a new local query which, when applied to the system $S$, transforms $S$ to a more complete system. In the final step, this new system is queried by the original query. If this original query is global and it is submitted to site $i$, a transfer of newly discovered knowledge from other sites of $DIS$ to a site $i$ is needed.

In relational databases the result of a query is a relation which can be queried further. Clearly, our $kdQAS$ should have a similar property. To achieve this, we will extend $DIS$ to a Distributed Knowledge System ($DKS$) where each site is defined as an information system coupled with a discovery layer simplified in this paper to a set of rules. Before we proceed any further, let us give an example of a $kd$-query.

For instance, $SQL$-type query

select $*$ from $Flights$
where $airline = "Delta"$
and $departure\_time = "morning"$
and $departure\_airport = "Charlotte"$
and $aircraft = "Boeing"$

is global (locally unreachable) for a database
$Flights(airline, departure\_time, arrival\_time, departure\_airport, arrival\_airport)$
because of the attribute $aircraft$. In order to resolve it, a transfer of newly discovered definitions of $aircraft = "Boeing"$ from other sites of $DIS$ to a site $i$ is needed. So, this query can be called a knowledge discovery query ($kd$-query).

We begin with a definition of $s(i)$-terms and their standard interpretation $M_i$ in a distributed information system $DS = (\{S_j\}_{j \in I}, L)$, where $S_j = (X_j, A_j, V_j, h_j)$ and $V_j = \bigcup\{V_{ja} : a \in A_j\}$, for any $j \in I$.

By a set of $s(i)$-terms we mean a least set $T_i$ such that:

- $\mathbf{0}, \mathbf{1} \in T_i$,
- $(a, w) \in T_i$ for any $a \in A_i$ and $w \in V_{ia}$,
- $\sim (a, w) \in T_i$ for any $a \in A_i$ and $w \in V_{ia}$,
- if $t_1, t_2 \in T_i$, then $(t_1 + t_2), (t_1 * t_2) \in T_i$.

We say that:

- $s(i)$-term $t$ is *atomic* if $t \in \{(a,w), \sim(a,w), \mathbf{0}, \mathbf{1}\}$ where $a \in B_i \subseteq A_i$ and $w \in V_{ia}$
- $s(i)$-term $t$ is *positive* if it is of the form $\prod\{(a,w) : a \in B_i \subseteq A_i \text{ and } w \in V_{ia}\}$
- $s(i)$-term $t$ is *primitive* if it is of the form $\prod\{t_j : t_j \text{ is atomic }\}$
- $s(i)$-term is in *disjunctive normal form* (DNF) if $t = \sum\{t_j : j \in J\}$ where each $t_j$ is primitive.

Standard interpretation $M_i$ of $s(i)$-terms in a distributed information system $DS = (\{S_j\}_{j \in I}, L)$ is defined as follows:

- $M_i(\mathbf{0}) = \emptyset$, $M_i(\mathbf{1}) = X_i$,
- $M_i((a,w)) = \{x \in X_i : w = h_i(x,a)\}$ for any $w \in V_i$,
- $M_i(\sim(a,w)) = \{x \in X_i : \sim(w \in h_i(x,a))\}$ for any $w \in V_i$,
- if $t_1, t_2$ are s(i)-terms, then
  $M_i(t_1 + t_2) = M_i(t_1) \cup M_i(t_2)$,
  $M_i(t_1 * t_2) = M_i(t_1) \cap M_i(t_2)$.

## 3 Distributed Knowledge Systems

In this section we introduce the notion of $i$-rules, $kd$-objects and, $kd$-queries. We define a Distributed Knowledge System ($DKS$) and introduce the notion of its consistency. We also provide a basic architecture of $DKS$. Finally, we describe the process of querying $kd$-objects at site $i$ of $DKS$.

The definition of $s(I)$-terms is similar to the definition of $s(i)$-terms with only one difference. Namely, the set $V_i$ in the definition of $s(i)$-terms is replaced by the set $V = \bigcup\{V_j : j \in I\}$. The meaning of $s(I)$-terms, which forms the foundations for $kdQAS$, is clarified after $kd$-objects and $kd$-queries are defined. It depends on the site of $DKS$ it is interpreted in.

By $(k,i)$-rule in $DS = (\{S_j\}_{j \in I}, L)$, $k, i \in I$, we mean a triple $(c,t,s)$ such that:

- $c \in V_k - V_i$,
- $t, s$ are $s(k)$-terms in DNF and they both belong to $T_k \cap T_i$,
- $M_k(t) \subseteq M_k(c) \subseteq M_k(t+s)$.

By $(i,i)$-rule in $DS = (\{S_j\}_{j \in I}, L)$, $i \in I$, we mean a triple $(c,t,s)$ such that:

- $c \in V_{ia}$,
- $t, s$ are $s(i)$-terms in DNF built from values of attributes belonging to $V_i - V_{ia}$,
- $M_i(t) \subseteq M_i(c) \subseteq M_i(t+s)$.

For simplicity reason both $(i,i)$-rules and $(k,i)$-rules are called $i$-rules.

System $DS = (\{(S_i, D_i, kdQAS_i, Agent_i)\}_{i \in I}, L)$, where (for any $i \in I$):

- $D_i$ is a discovery layer simplified to a consistent set of $i$-rules,
- $S_i$ is an information system
- $kdQAS_i$ is a query answering system for a site $i$,
- $Agent_i$ is a set of knowledge discovery based client/server protocols.

is called a Distributed Knowledge System ($DKS$).

If there is $i \in I$ such that $S_i$ is incomplete, then $DKS$ is called incomplete Distributed Knowledge System. Figure 1 shows its basic architecture (WWW interface is added).

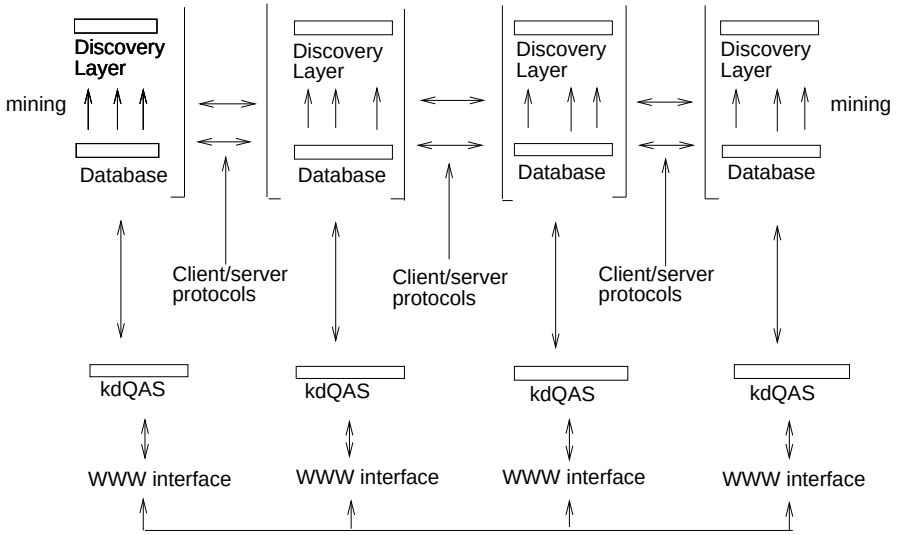


**Fig. 1.** Distributed Knowledge System

By $kd$-object at site $i$ of $DS = (\{(S_i, D_i, kdQAS_i, Agent_i)\}_{i \in I}, L)$, we mean any subsystem of $(S_i, D_i)$ or saying another words any subsystem of $S_i$ coupled with a consistent set of $i$-rules.

By $kd$-query at site $i$, $i \in I$, we mean either any $s(I)$-term or any consistent set of $(i, i)$-rules.

Now, we describe the process of querying a $kd$-object at site $i$ of $DKS$. In this paper we consider four options for a $kd$-query $q$:

- $q$ is a *primitive* $s(i)$-term,
- $q$ is an $s(i)$-term in DNF,
- $q$ is a *primitive* $s(I)$-term,

– $q$ is an $s(I)$-term in DNF.

Let us assume that $q_i$ is a *primitive* $s(i)$-term. First, $kdQAS_i$ identifies all incomplete attributes among attributes used in $q_i$. Let us say that $a_{i1}, a_{i2}, ..., a_{ik}$ is the list of all these attributes. In the second step, $kdQAS_i$ finds all certain rules at $S_i = (X_i - In(X_i, a_{ij}), A_i, V_i, h_i)$ describing attribute $a_{ij}$ in terms of attributes from $A_i - \{a_{ij}\}$. Let's denote these rules by $R_{ij}$. This process is repeated for every $j \in \{1, 2, 3, ..., k\}$. In the third step, $kdQAS_i$ applies the rules in $R_{ij}$ to find the value of $a_{ij}$ for a maximal number of objects from $(In(X_i, a_{ij}), A_i, V_i, h_i)$. These values are stored in a temporary matrix $M_{ij}$, for every $j \in \{1, 2, 3, ..., k\}$. In the fourth step, all values stored in temporary matrices $M_{ij}$ are moved to corresponding locations in a system $S_i = (X_i, A_i, V_i, h_i)$ to replace some of its null values. Let us denote the resulting information system by $S_i^1$. At this point, $kdQAS_i$ goes back again to the first step and the process continues to iterate until all newly generated temporary matrices are empty. Let us assume that after $m$ iterations, the process will stop and denote the resulting information system by $S_i^m$. In the final step, $kdQAS_i$ finds all objects in $S_i^m$ satisfying the description $q_i$.

If $q_i$ is an $s(i)$-term in DNF, then the strategy described for primitive $s(i)$-terms is repeated for every disjunct of $q_i$.

Assume now that $q_i$ is a *primitive* $s(I)$-term which is global for $S_i$. First, $kdQAS_i$ identifies all incomplete attributes in $A_i$ among attributes used in $q_i$. Let us say that $a_{i1}, a_{i2}, ..., a_{ik}$ is the list of all these attributes. In the second step, $kdQAS_i$ finds all certain rules at $S_i = (X_i - In(X_i, a_{ij}), A_i, V_i, h_i)$ describing attribute $a_{ij}$ in terms of attributes from $A_i - \{a_{ij}\}$. Let's denote these rules by $R_{ij}$. This process is repeated for every $j \in \{1, 2, 3, ..., k\}$. In the third step, $kdQAS_i$ applies the rules in $R_{ij}$ to find the value of $a_{ij}$ for a maximal number of objects from $(In(X_i, a_{ij}), A_i, V_i, h_i)$. These values are stored in a temporary matrix $M_{ij}$, for every $j \in \{1, 2, 3, ..., k\}$. In the fourth step, all values stored in temporary matrices $M_{ij}$ are moved to corresponding locations in a system $S_i = (X_i, A_i, V_i, h_i)$ to replace some of its null values. Let us denote the resulting information system by $S_i^1$. At this point, $kdQAS_i$ goes back again to the first step and the process continues to iterate until all newly generated temporary matrices are empty. Let us assume that after $m$ iterations, the process will stop and denote the resulting information system by $S_i^m$. Now, $kdQAS_i$ identifies all attributes used in $q_i$ which do not belong to $A_i$ (we call them concepts for the site $i$). $Agent_i$ sends request to other sites of $DKS$ to find rules describing all these concepts in terms of attributes from $A_i$. These newly discovered rules are used to approximate query $q_i$ by a new local query $p_i$ for $S_i$. Also, these rules are stored in the discovery layer $D_i$ which can be used (for new global queries) by $kdQAS_i$ before any help from $Agent_i$ is requested. In the final step, $kdQAS_i$ finds all objects in $S_i^m$ satisfying the description $p_i$.

If $q_i$ is an $s(I)$-term in DNF, then the strategy described for primitive $s(I)$-terms is repeated for every disjunct of $q_i$.

## 4 Interpretation of Primitive *kd*-Queries

In this section we propose a class of $i$-based operational semantics for a $kd$-query $q$, assuming that $q$ is an $s(I)$-term. Next, for this class of operational semantics we give a complete and sound set of axioms and rules.

Standard interpretation $M_i$, introduced in a previous section, shows how to interpret $i$-queries in $DIS$. Now, we propose how to interpret non-local queries (called global) bounded in this paper to the class of *primitive* $s(I)$-terms. We call them primitive $kd$-queries.

Parentheses will be used, if necessary, in the obvious way. As will turn out later, the order of a sum or product is immaterial. So, we will abbreviate finite sums and products as $\sum\{t_j : j \in J\}$ and $\prod\{t_j : j \in J\}$, respectively. Intentionally, $s(I)$-terms are names of certain features of parts being processed by $kdQAS$, more complex than those expressed by constants.

Let $M_i$ be a standard interpretation of $s(i)$-terms in $DS = (\{S_j\}_{j \in I}, L)$.

Let $C_i = \bigcup\{V_k : k \in I - \{i\}\} - V_i$ is a set of concepts for $S_i$. By $i$-based operational semantics for $s(I)$-terms in $DS = (\{(S_i, D_i)\}_{i \in I}, L)$, $S_i = (X_i, A_i, V_i, h_i)$ and $V_i = \bigcup\{V_{ia} : a \in A_i\}$, we mean the interpretation $M_{i,K_i}$ such that:

- $M_{i,K_i}(\mathbf{0}) = \emptyset$, $M_{i,K_i}(\mathbf{1}) = X_i$
- for any $w \in V_i$,
  $M_{i,K_i}(w) = M_i(w)$,
  $M_{i,K_i}(\sim w) = X_i - M_{i,K_i}(w)$

- for any $w \in C_i$,
  $M_i(w) = \{x \in X_i : (\exists t, s)((w, t, s) \in D_i \wedge x \in M_i(t))\}$
  $M_i(\sim w) = \{x \in X_i : (\exists t, s)((w, t, s) \in D_i \wedge x \notin M_i(s))\}$

- for any $s(I)$-terms $t_1, t_2$
  $M_{i,K_i}(t_1 + t_2) = M_{i,K_i}(t_1) \cup M_{i,K_i}(t_2)$,
  $M_{i,K_i}(t_1 * t_2) = M_{i,K_i}(t_1) \cap M_{i,K_i}(t_2)$,
  $M_{i,K_i}(\sim (t_1 + t_2)) = (\sim M_{i,K_i}(t_1)) \cap (\sim M_{i,K_i}(t_2))$,
  $M_{i,K_i}(\sim (t_1 * t_2)) = (\sim M_{i,K_i}(t_1)) \cup (\sim M_{i,K_i}(t_2))$,
  $M_{i,K_i}(\sim\sim t) = M_{i,K_i}(t)$.

- for any $s(I)$-terms $t_1, t_2$
  $M_{i,K_i}(t_1 = t_2) = (\text{ if } M_{i,K_i}(t_1) = M_{i,K_i}(t_2) \text{ then } T \text{ else } F)$
  where $T$ stands for $True$ and $F$ for $False$

From the point of view of $S_i$ the interpretation $M_{i,K_i}$ represents a pessimistic approach to query evaluation. It means that $M_{i,K_i}(t)$ is interpreted as a set of objects in $X_i$ which have the property $t$ for sure. We are not retrieving here objects which might have property $t$.

Let us adopt the following set $A$ of Axiom Schemata:

$A1$. Substitutions of the axioms of distributive lattices for $s(I)$-terms and the axioms of equality

$A2$. $\sim w * w = \mathbf{0}$ for any constant $w$

$A3$. $\sim w + w = \mathbf{1}$ for any $w \in V_i$

$A4$. for each $w \in V_i$ there is a subset $w_1, w_2, ..., w_n$ of $V_i$ such that $\sim w = w_1 + w_2 + ... + w_n$

$A5$. $v_1 * v_2 = 0$ if $v_1, v_2 \in V_{ia}$ for some $a \in A_i$

$A6$. for any $s(I)$-term t, $\sim \mathbf{0} = \mathbf{1}$, $\sim \mathbf{1} = \mathbf{0}$, $\mathbf{1} + t = \mathbf{1}$, $\mathbf{1} * t = t$, $\mathbf{0} * t = \mathbf{0}$, $\mathbf{0} + t = t$, $\sim\sim t = t$

$A7$. for any $w \notin V_i$ $w = \sum\{t : [w, t, s] \in D_i\}$

$A8$. for any $w \notin V_i$ $\sim w = \sum\{t : [\sim w, t, s] \in D_i\}$

$A9$. $\sim (t_1 + t_2) = (\sim t_1) * (\sim t_2)$

$A10$. $\sim (t_1 * t_2) = (\sim t_1) + (\sim t_2)$

$A11$. Substitutions of the propositional calculus axioms

The rules of inference for our formal system are the following:

$R1$. from $(\alpha \Rightarrow \beta)$ and $\alpha$ we can deduce $\beta$ for any formulas $\alpha, \beta$

$R2$. from $t_1 = t_2$ we can deduce $t(t_1) = t(t_2)$, where $t(t_1)$ is a $s(I)$-term containing $t_1$ as a subterm and $t(t_2)$ comes from $t(t_1)$ by replacing some of the occurrences of $t_1$ with $t_2$.

We write $A \vdash (t_1 = t_2)$ if there exists a derivation from a set $A$ of formulas as premises to the formula $(t_1 = t_2)$ as the conclusion.

We write $A \models (t_1 = t_2)$ to denote the fact that $A$ semantically implies $(t_1 = t_2)$, that is, for any $i$-standard interpretation $M_{i,K_i}$ of $s(I)$-terms in $DKS$ we have $M_{i,K_i}(t_1 = t_2) = T$.

**Theorem 1.** (Completeness). For any $s(I) - terms$ $t_1, t_2$, if $A \models (t_1 = t_2)$ then $A \vdash (t_1 = t_2)$.

The above completeness theorem gives us the set of axioms which is sound and sufficient to transform any global $s(I)$-term to its equivalent DNF. So, the set of $kd$-queries does not have to be bounded to primitive terms.

## 5 Conclusion

This paper presents a methodology and theoretical foundations of a kd-Query Answering System for $DKS$ which is partially implemented at UNC-Charlotte on a cluster of SPARC 20 workstations.

## References


1. Batini, C., Lenzerini, M., Navathe, S., "A comparative analysis of methodologies for database schema integration", in *ACM Computing Surveys*, Vol 18, No. 4, 1986, 325-364
2. Bosc, P., Pivert, O., "Some approaches for relational databases flexible querying", in *Journal of Intelligent Information Systems*, Kluwer Academic Publishers, Vol. 1, 1992, 355-382
3. Chu, W.W., "Neighborhood and associative query answering", in *Journal of Intelligent Information Systems*, Kluwer Academic Publishers, Vol. 1, 1992, 355-382
4. Chu, W.W., Chen, Q., Lee, R., "Cooperative query answering via type abstraction hierarchy", in *Cooperating Knowledge-based Systems* (ed. S.M. Deen), North Holland, 1991, 271-292
5. Cuppers, F., Demolombe, R., "Cooperative answering: a methodology to provide intelligent access to databases", in *Proceedings 2nd International Conference on Expert Database Systems*, Virginia, USA, 1988
6. Gaasterland, T., Godfrey, P., Minker, J., "An overview of cooperative answering", *Journal of Intelligent Information Systems*, Kluwer Academic Publishers, Vol. 1, 1992, 123-158
7. Grzymala-Busse, J., *Managing uncertainty in expert systems*, Kluwer Academic Publishers, 1991
8. Maluf, D., Wiederhold, G., "Abstraction of representation for interoperation", in *Proceedings of Tenth International Symposium on Methodologies for Intelligent Systems*, LNCS/LNAI, Springer-Verlag, No. 1325, 1997, 441-455
9. Navathe, S., Donahoo, M., "Towards intelligent integration of heterogeneous information sources", in *Proceedings of the Sixth International Workshop on Database Re-engineering and Interoperability*, 1995
10. Pawlak, Z., "Rough Sets - theoretical aspects of reasoning about data", Kluwer Academic Publishers, 1991
11. Pawlak, Z., "Rough sets and decision tables", in *Proceedings of the Fifth Symposium on Computation Theory*, Springer Verlag, Lecture Notes in Computer Science, Vol. 208, 1985, 118-127
12. Ras, Z., "Resolving queries through cooperation in multi-agent systems", in *Rough Sets and Data Mining*, (Eds. T.Y. Lin, N. Cercone), Kluwer Academic Publishers, 1997, pp. 239-258
13. Ras, Z., Joshi, S., "Query approximate answering system for an incomplete DKBS", in *Fundamenta Informaticae Journal*, IOS Press, Vol. 30, No. 3/4, 1997, 313-324
14. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, "Advances in Knowledge Discovery and Data Mining", AAAI Press/MIT Press, 1996
15. Ras, Z., "Collaboration control in distributed knowledge-based systems", in *Information Sciences Journal*, Elsevier, Vol. 96, No. 3/4, 1997, pp. 193-205
16. Skowron, A., "Boolean reasoning for decision rules generation", in *Methodologies for Intelligent Systems, Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems*, (eds. J. Komorowski, Z. Ras), Lecture Notes in Artificial Intelligence, Springer Verlag, No. 689, 1993, 295-305


This article was processed using the LaTeX macro package with LLNCS style