Abstract: Traditional query processing provides exact answers to queries. It usually requires that users fully understand the database structure and content to issue a query. Due to the complexity of the database applications, so called locally non-reachable queries can be posed which traditional query answering systems can not handle. In this paper a query rough-answering system for a multi-agent system is presented to rectify these problems.

Keywords: intelligent information system, query answering system, rough sets, multi-agent system, knowledge discovery.

1 Introduction

By a cooperative knowledge-based system (CKBS) we mean a collection of autonomous knowledge-based systems called agents (sites) which are capable of interacting with each other. Each agent is represented by an information system (collection of data) and a knowledge base called here a dictionary (collection of rules). In [6] and [12], each site of CKBS can learn from its neighbors the descriptions of all unknown attribute values used in queries entering that site. These descriptions are not precise and they only provide lower and upper approximations of attribute values. Semantically, we can see them as rough sets [8]. We store them in dictionaries. Additionally, we assume that some descriptions can be provided by experts and added, if necessary, in the form of rules to appropriate dictionaries of CKBS. In [10] we have investigated the problem of repairing inconsistent dictionaries. In this paper we assume that the query answering system (QAS) for a CKBS can use only consistent rules. This way, QAS does not have to deal with uncertain answers.

Any site of CKBS can be a source of a local or a global query. By a local query for a site \(i\) we mean a query entirely built from values of attributes local for \(i\). Local queries need only to access the information system of the site where they were issued and they are completely processed on the system associated with that site. In order to resolve a global query for a site \(i\) (built from values of attributes not necessarily local for \(i\)) successfully, the query answering system has to access local dictionaries or/and information systems at other sites of CKBS because of the need of additional data.

In this paper, we outline a query rough-answering system for CKBS. This is different from solving queries on a conventional distributed relational database in the sense that it uses rules to resolve unknown attributes. We assume that along with a query the user has to provide a list of sites from which objects satisfying the query should be retrieved. Our query rough-answering system gives a set of objects from all these sites which certainly satisfy the query and another set of objects which possibly satisfy the query.

2 Basic Definitions

In this section, we recall [12] the notion of an information system, a distributed information system, a dictionary, and \(s(i)\)-terms which are called local for a site \(i\). We introduce the notion of a rough rule and show the process of building dictionaries.
Information system is defined as 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. 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 \(a \neq b\),
- \(h : X \times A \rightarrow V\) where \(h(x, a) \in V_a\) for any \(x \in X, a \in A\).

We use a table-representation of a classification function \(h\) which is naturally identified with an information system \(S = (X, A, V, h)\). For instance, let us assume that \(S_2 = (X_2, A_2, V_2, h_2)\) is an information system where \(X_2 = \{a_1, a_5, a_9, a_{10}, a_{11}, a_{12}\}\), \(A_2 = \{C, D, E, F, G\}\) and \(V_2 = \{e_1, e_2, e_3, f_1, f_2, g_1, g_2, g_3, c_1, c_2, d_1, d_2\}\). Additionally, we assume 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 \(h_2\) defined by Table 1 is identified with an information system \(S_2\).

<table>
<thead>
<tr>
<th>(X_2)</th>
<th>(F)</th>
<th>(C)</th>
<th>(D)</th>
<th>(E)</th>
<th>(G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a_1)</td>
<td>(f_1)</td>
<td>(c_1)</td>
<td>(d_2)</td>
<td>(e_1)</td>
<td>(g_1)</td>
</tr>
<tr>
<td>(a_6)</td>
<td>(f_2)</td>
<td>(c_1)</td>
<td>(d_2)</td>
<td>(e_3)</td>
<td>(g_2)</td>
</tr>
<tr>
<td>(a_8)</td>
<td>(f_1)</td>
<td>(c_2)</td>
<td>(d_1)</td>
<td>(e_3)</td>
<td>(g_1)</td>
</tr>
<tr>
<td>(a_9)</td>
<td>(f_2)</td>
<td>(c_1)</td>
<td>(d_1)</td>
<td>(e_3)</td>
<td>(g_1)</td>
</tr>
<tr>
<td>(a_{10})</td>
<td>(f_2)</td>
<td>(c_2)</td>
<td>(d_1)</td>
<td>(e_3)</td>
<td>(g_2)</td>
</tr>
<tr>
<td>(a_{11})</td>
<td>(f_1)</td>
<td>(c_2)</td>
<td>(d_1)</td>
<td>(e_3)</td>
<td>(g_1)</td>
</tr>
<tr>
<td>(a_{12})</td>
<td>(f_1)</td>
<td>(c_1)</td>
<td>(d_1)</td>
<td>(e_3)</td>
<td>(g_1)</td>
</tr>
</tbody>
</table>

Table 1: Information System \(S_2\)

By a distributed information system \([12]\) 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 informationsystem 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

\[
(\forall i)(\forall j)(\forall x \in X_i \cap X_j)(\forall a \in A_i \cap A_j)[(x, a) \in \text{Dom}(h_i) \cap \text{Dom}(h_j) \rightarrow h_i(x, a) = h_j(x, a)].
\]

Otherwise \(DS\) is called inconsistent. In this paper we only deal with distributed informationsystems which are consistent.

Now, we recall the notion of a dictionary \(D_{ki}\), \((k, i) \in L^+\), containing approximate descriptions of values of attributes (they are in a form of rules) from \(A_k \rightarrow A_i\) in terms of values of attributes from \(A_k \cap A_i\) (see [12]).

We begin with definitions of \(s(i)\)-terms, \(s(i)\)-formulas 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:

- \(0, 1 \in T_i\),
- \(w \in T_i\) for any \(w \in V_i\),
- if \(t_1, t_2 \in T_i\), then \((t_1 + t_2), (t_1 * t_2), \sim t_1 \in T_i\).

We say that:

- \(s(i)\)-term \(t\) is primitive if it is of the form \(\prod \{w : w \in U_i\}\) for any \(U_i \subseteq V_i\),
- \(s(i)\)-term \(t = \prod \{w : w \in U_i\}\) where \(U_i \subseteq V_i\) is simple if \(U_i \cap V_{in}\) is a singleton set for any \(a \in A_i\),
- \(s(i)\)-term is in disjunctive normal form (DNF)
  if \(t = \prod \{w \in \bigcup \{T_j : j \in J\} \mid \text{each } t_j \text{ is primitive.}\}

By a set of \(s(i)\)-formulas we mean a least set \(F_i\) such that:

- if \(t_1, t_2 \in T_i\), then \((t_1 = t_2) \in F_i\).
- if \(\alpha, \beta\) are \(s(i)\)-formulas, then \(\alpha \lor \beta, \alpha \land \beta, \alpha \Rightarrow \beta, \sim \alpha\) are \(s(i)\)-formulas

Elements in \(L(S_i) = T_i \cup F_i\) represent queries local for a site \(i\). Following Grzymala-Busse [4] we call them reachable in \(S_i\) or simply \(i\)-reachable.

For example, a DNF query

\[
\text{select } * \text{ from Flights}
\]
where \(\text{airline} = "\text{Delta}"\)
and \(\text{departure_time} = "\text{morning}"\)
and \(\text{departure_airport} = "\text{Charlotte}"\)

is reachable in a database

\[
\text{Flights(airline, departure_time, arrival_time, departure_airport, arrival_airport)}.
\]

Standard interpretation \(M_i\) of \(i\)-reachable queries in a distributed information system \(DS = (\{S_j\}_{j \in I}, L)\) is defined as follows:
• \( M_f(\emptyset) = \emptyset \), \( M_f(1) = X_i \)
• \( M_f(w) = \{ x \in X_i : w \in V_{i\alpha} \text{ then } w = h_i(x,a) \} \) for any \( w \in V_i \).
• if \( t_1, t_2 \) are \( s(i) \)-terms, then
  \[
  M_i(t_1 + t_2) = M_i(t_1) \cup M_i(t_2), \\
  M_i(t_1 \cdot t_2) = M_i(t_1) \cap M_i(t_2), \\
  M_i(\sim t_1) = X_i - M_i(t_1), \\
  M_i(t_1 t_2) = \\
  \begin{cases} 
  M_i(t_1) & \text{if } M_i(t_1) = M_i(t_2) \\
  T & \text{else } F
  \end{cases}
  \]
where \( T \) stands for \( True \) and \( F \) for \( False \).

• for any \( I(S) \)-formulas \( \alpha, \beta \)
  \[
  J_S(\alpha \lor \beta) = J_S(\alpha) \lor J_S(\beta), \\
  J_S(\alpha \land \beta) = J_S(\alpha) \land J_S(\beta), \\
  J_S(\alpha \rightarrow \beta) = J_S(\alpha) \rightarrow J_S(\beta), \\
  J_S(\neg \alpha) = \neg J_S(\alpha)
  \]

Let us adopt the following set \( A_L \) of axioms schemata:
• Substitutions of the axioms of Boolean Algebras for terms and the axioms of equality ( \( = \) )
• \( \neg \, w \cdot w = 0 \) for any \( w \in V_i \)
• \( \neg \, w + w = 1 \) for any \( w \in V_i \)
• for each \( w \in V_{i\alpha} \) there is a subset \( \{ w_1, w_2, \ldots, w_n \} \) of \( V_{i\alpha} \) such that
  \( \neg \, w = w_1 + w_2 + \ldots + w_n \)
• for any term \( t \)
  \[
  \begin{align*}
  0 + 1 & = 1, \\
  1 + t & = 1, \\
  1 \cdot t & = t, \\
  0 \cdot t & = 0, \\
  0 + t & = t,
  \end{align*}
  \]
• 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 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 \( A_L \vdash \alpha \) if there exists a derivation from a set \( A_L \) of formulas as premises to the formula \( \alpha \) as the conclusion.

We write \( A_L \models \alpha \) to denote the fact that \( A_L \) semantically implies \( \alpha \), that is, for any standard interpretation \( M_i \) of queries, we have \( M_i(\alpha) = T \).

**Theorem 1** (Completeness Theorem)

For any terms \( t_1, t_2 \in T_i \),
\[
A_L \vdash t_1 = t_2 \text{ iff } A_L \models t_1 = t_2.
\]
For any formula \( \alpha \in F_i \),
\[
A_L \vdash \alpha \text{ iff } A_L \models \alpha.
\]

By \((k,i)\)-rule in \( DS = \{ \{ S_i \} \}_{i \in 1, L} \), \( k, i \in I \), 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 \( T_k \cap T_i \),
• \( M_k(t) \subseteq M_k(c) \subseteq M_k(t + s) \).

We say that \((k,i)\)-rule \((c, t, s)\) is in \( k \)-optimal form if there are no other \( s(k) \)-terms \( t_1, s_1 \in T_k \cap T_i \), both in DNF such that:
• \( M_k(t) \subseteq M_k(t_1) \subseteq M_k(c) \),
• \( M_k(t) \neq M_k(t_1) \),
• \( M_k(\sim (t + s)) \subseteq M_k(s_1) \subseteq M_k(\sim c) \),
• \( M_k(\sim (t + s)) \neq M_k(s_1) \).

For any \((k,i)\)-rule \((c, t, s)\) in \( DS = \{ \{ S_i \} \}_{i \in 1, L} \), we say that:
• \((t \rightarrow c)\) is a \( k \)-certain rule in \( DS \),
• \((t + s \rightarrow c)\) is a \( k \)-possible rule in \( DS \).

Let us assume that \( r_1 = (c_1, t_1, s_1) \), \( r_2 = (c_2, t_2, s_2) \) are \((k,i)\)-rules. We say that: \( r_1, r_2 \) are strongly consistent, if either \( c_1, c_2 \) are values of two different attributes in \( S_k \) or a DNF form equivalent to \( t_1 \cdot t_2 \) does not contain simple conjuncts.

Now, we are ready to recall the notion of a dictionary \( D_{ki} \). Its elements can be seen as approximate descriptions of values of attributes from \( V_k - V_i \) in terms of values of attributes from \( V_k \cap V_i \).

To be more precise, we assume that \( D_{ki} \) is a set of \((k,i)\)-rules such that:
if \((c, t, s) \in D_{ki} \) and \( t_1 = \neg (t + s) \) is a tautology then \((\neg c, t_1, s) \in D_{ki} \) (see \([12]\))

Dictionary \( D_{ki} \) is in \( k \)-optimal form if all its \((k,i)\)-rules are in \( k \)-optimal form.

Let us assume that a distributed information system \( DS = \{ \{ S_i \} \}_{i \in 1, 2} \) has two sites, one represented by Table 1 and the second by Table 2. We assume here that \( V_D = \{ b_1, b_2 \} \), \( V_C = \{ c, c_1, c_2, c_3 \} \), \( V_D = \{ d, d_1 \} \), and \( V_E = \{ c_1, c_2, c_3 \} \).

The following rules can be computed directly from the information system \( S_1 : (b_1, c_1 \cdot d_1), (c_1, c_2 \cdot c_3, c_2 \cdot c_1 + c_3 + d_1 + c_3 + c_2 \cdot d_1 + c_2) \).

The dictionary \( D_{12} \) in 1-optimal form may contain
Table 2: Information System $S_1$

<table>
<thead>
<tr>
<th>X</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>c1</td>
<td>d1</td>
<td>e1</td>
</tr>
<tr>
<td>a2</td>
<td>b1</td>
<td>c2</td>
<td>d1</td>
<td>e2</td>
</tr>
<tr>
<td>a3</td>
<td>b2</td>
<td>c3</td>
<td>d2</td>
<td>e1</td>
</tr>
<tr>
<td>a4</td>
<td>b1</td>
<td>c2</td>
<td>d1</td>
<td>e2</td>
</tr>
<tr>
<td>a5</td>
<td>b1</td>
<td>c3</td>
<td>d3</td>
<td>e1</td>
</tr>
<tr>
<td>a6</td>
<td>b2</td>
<td>c1</td>
<td>d3</td>
<td>e3</td>
</tr>
<tr>
<td>a7</td>
<td>b2</td>
<td>c2</td>
<td>d1</td>
<td>e2</td>
</tr>
</tbody>
</table>

the following (1, 2)-rules: $(b_1, d_1 * e_1 + d_1 * e_3, e_2), (b_2, d_2, e_2)$.

Dictionary $D_{k_i}$ is built at site $k$ and some of its elements (rules), if needed, can be sent to the site $i$ of a distributed information system $DS = \{\{S_i\}_{j \in J}, I\}$, for any $k, i \in I$ (see [6, [12]). Dictionary $D_{k_i}$ will represent beliefs of agent $k$ at site $i$. Elements of dictionaries $D_{k_i}, k \in J$ sent to site $i$ can be stored there. This way site $i$ knows beliefs of all agents from $J$. Clearly, all these beliefs may form a set at site $i$ which is inconsistent. In [10] we proposed a repair strategy for an inconsistent set of rules. In this paper, for simplicity reasons, we assume that all dictionaries are consistent.

3 Query Rough-Answering System

In this section, we present a query rough-answering system for a distributed knowledge-based system.

Let us take an information system $S_k = (X_k, A_k, V_k, f_k)$ where $X_k = \{a_1, a_3, a_4, a_6, a_9, a_{10}, a_{11}\}, V_k = \{b_1, b_2, c_1, c_2, f_1, f_2, f_3, g_1, g_2, g_3, k_1, k_2, l_1, l_2\}, I_k = \{i_1, i_2, i_3, i_4, i_5, i_6\}$, and $f_k$ is defined by Table 3.

Table 3: Information System $S_k$

<table>
<thead>
<tr>
<th>$X_k$</th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
<th>$i_4$</th>
<th>$i_5$</th>
<th>$i_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>h1</td>
<td>c1</td>
<td>f2</td>
<td>g1</td>
<td>k1</td>
<td>l1</td>
</tr>
<tr>
<td>a3</td>
<td>h2</td>
<td>c1</td>
<td>f1</td>
<td>g1</td>
<td>k1</td>
<td>l1</td>
</tr>
<tr>
<td>a4</td>
<td>h1</td>
<td>c1</td>
<td>f2</td>
<td>g2</td>
<td>k1</td>
<td>l1</td>
</tr>
<tr>
<td>a6</td>
<td>h2</td>
<td>c2</td>
<td>f3</td>
<td>g3</td>
<td>k2</td>
<td>l2</td>
</tr>
<tr>
<td>a8</td>
<td>h2</td>
<td>c2</td>
<td>f2</td>
<td>g2</td>
<td>k2</td>
<td>l2</td>
</tr>
<tr>
<td>a9</td>
<td>h1</td>
<td>c1</td>
<td>f1</td>
<td>g1</td>
<td>k1</td>
<td>l1</td>
</tr>
<tr>
<td>a10</td>
<td>h2</td>
<td>c1</td>
<td>f2</td>
<td>g2</td>
<td>k2</td>
<td>l2</td>
</tr>
<tr>
<td>a11</td>
<td>h1</td>
<td>c1</td>
<td>f2</td>
<td>g1</td>
<td>k1</td>
<td>l1</td>
</tr>
</tbody>
</table>

Now, to recall our strategy, let us assume that information system represented by Table 3 is used to generate rules describing $e_1, e_2$ in terms of $\{f_1, f_2, f_3, g_1, g_2, k_1, k_2\}$. So, the attribute $i_2$ is a decision attribute and $\{i_3, i_4, i_5\}$ is a set of classification attributes. We assume here that the request to compute these rules came from another site of a distributed system $DS$. Following Grzymala-Busse in [4], we can easily check that $f_3 * g_3 * k_2 \rightarrow e_2$ is a certain rule and $f_3 * g_3 * k_2 + f_2 * g_2 * k_2 \rightarrow e_2$ is a possible one in $S_k$. Similarly, $f_1 * g_1 * k_1 + f_2 * g_1 * k_1 + f_2 * g_2 * k_1 \rightarrow e_1$ is a certain rule and $f_1 * g_1 * k_1 + f_2 * g_1 * k_1 + f_2 * g_2 * k_1 \rightarrow e_1$ is a possible rule in $S_k$. Now, assuming that $S_k$ is closed (we are not allowed to make any updates or add new tuples), we optimize the rules in $S_k$. As a result, we get two generalized certain rules: $f_3 \rightarrow e_2$ and $k_1 \rightarrow e_1$.

The generalization process for possible rules is more complex. Clearly, $k_1 + f_2 * g_2 * k_2 \rightarrow e_1$ is a possible rule and $\sim (k_1 + f_2 * g_2 * k_2) \rightarrow e_1$ is a certain one. Now, observe that: $\sim (k_1 + f_2 * g_2 * k_2) = \sim k_1 * ((\sim f_2) + ((\sim g_2) + ((\sim k_2)) = k_2 * ((f_1 + k_3)) * (g_1 + k_1) = k_2 * f_1 + k_2 * f_3 + k_2 * g_1$ is a sequence of tautologies. So, $k_2 * f_1 + k_2 * f_3 + k_2 * g_1 \rightarrow e_1$ is a certain rule and the same $\sim (k_2 * f_1 + k_2 * f_3 + k_2 * g_1) \rightarrow e_1$ is a possible one. This rule is equivalent to $(k_1 + f_2 + f_3) * (k_1 + f_1 + f_2) * (k_1 + g_2) \rightarrow e_1$ and finally equivalent to a possible rule $k_1 + (f_2 + g_2) \rightarrow e_1$.

<table>
<thead>
<tr>
<th>$X_m$</th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
<th>$i_4$</th>
<th>$i_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>f2</td>
<td>c1</td>
<td>d1</td>
<td>e1</td>
<td>g1</td>
</tr>
<tr>
<td>a6</td>
<td>f2</td>
<td>c1</td>
<td>d2</td>
<td>e3</td>
<td>g2</td>
</tr>
<tr>
<td>a7</td>
<td>f1</td>
<td>c2</td>
<td>d1</td>
<td>e3</td>
<td>g1</td>
</tr>
<tr>
<td>a11</td>
<td>f1</td>
<td>c1</td>
<td>d2</td>
<td>e3</td>
<td>g1</td>
</tr>
<tr>
<td>a13</td>
<td>f1</td>
<td>c2</td>
<td>d2</td>
<td>e3</td>
<td>g1</td>
</tr>
<tr>
<td>a14</td>
<td>f1</td>
<td>c2</td>
<td>d1</td>
<td>e3</td>
<td>g2</td>
</tr>
<tr>
<td>a15</td>
<td>f1</td>
<td>d1</td>
<td>e1</td>
<td>g1</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Information System $S_m$

Let us assume that an information system $S_m = (X_m, A_m, V_m, f_m)$ where $X_m = \{a_1, a_6, a_7, a_{11}, a_{13}, a_{14}, a_{15}\}, V_m = \{c_1, c_2, c_1, c_2, c_3, d_1, d_2, f_1, f_2, g_1, g_2\}, I_m = \{i_1, i_2, i_3, i_4, i_5\}$, and $f_m$ is defined by Table 4.

System $S_m$ represents one of the sites of $DS$. Now, employing the strategy described in [12], we can generate two globally sound rules from $S_m$: $(d_1, e_1, f_1 * e_3)$ and $(d_2, f_2 * c_3, f_1 * e_3)$. These rules can be added to the knowledge-base $K_{B_k}$ assigned to the site $k$ of our distributed information system $DS$. In [12], we
proposed how to change the semantics of queries at site \( k \) when \( KB_k \) is updated. Clearly, the semantics of \( L(DS) \) is improving if more local objects are retrieved for the same global query.

Let us assume that \((S_k, KB_k), k \in K \) represents one of the sites of a distributed knowledge-based system \( DS \), \( S_k = (X_k, A_k, V_k, f_k) \) and \( J_k \) is a standard interpretation of \( s(k)\)-terms and \( s(k)\)-formulas in \( DS \).

Also, we assume that:

- \( (X_1, Y_1) \cup (X_2, Y_2) = (X_1 \cup X_2, Y_1 \cup Y_2) \),
- \( (X_1, Y_1) \cap (X_2, Y_2) = (X_1 \cap X_2, Y_1 \cap Y_2) \).

By a set of \( L(DS)\)-terms we mean a least set such that:

- constants \( 0 = (0, 0), 1 = (1, 1) \) are \( L(DS)\)-terms,
- if \( v \in \bigcup \{V_k : k \in K\} \) then \( (v, v) \) is an \( L(DS)\)-term
- if \( t_1, t_2 \) are \( L(DS)\)-terms then \( (t_1 + t_2), (t_1 \ast t_2), \sim t_1 \) are \( L(DS)\)-terms

\( L(DS)\)-formulas are defined as a least set such that:

- if \( t_1, t_2 \) are \( L(DS)\)-terms, then \( t_1 = t_2 \) is an \( L(DS)\)-formula
- if \( \alpha, \beta \) are \( L(DS)\)-formulas, then \( \alpha \lor \beta, \alpha \land \beta, \alpha \Rightarrow \beta, \sim \alpha \) are \( L(DS)\)-formulas

\( L(DS)\)-terms and \( L(DS)\)-formulas represent global queries for any of the sites of \( DS \).

By a standard interpretation of \( L(DS)\)-formulas and \( L(DS)\)-terms at site \( k \), we mean a function \( M_k \) such that:

- \( M_k(0) = (0, 0) \),
- \( M_k(1) = (X_k, X_k) \)
- for any \( w \in V_k \), \( M_k((w, w)) = (J_k(w), J_k(w)) \)
- for any \( w \not\in V_k \), \( M_k((w, w)) = \emptyset \)
- for any \( w \not\in V_k \), \( M_k((w, w)) = \emptyset \)
- for any \( w \not\in V_k \), \( M_k(\sim (w, w)) = \emptyset \)
- for any \( w \not\in V_k \), \( M_k(\sim (w, w)) = \emptyset \)

\( M_k(\sim (t + s)) = M_k(\sim t) \cup M_k(\sim s) \)

\( M_k(\sim (t \ast s)) = M_k(\sim t) \cup M_k(\sim s) \)

\( M_k(\sim (t)) = M_k(t) \)

For simplicity reason, we will identify any \( L(DS)\)-term \( (w, w) \) with a constant \( w \) if it does not lead us to a confusion.

Now, if a global query is submitted to a site \( k \) of a distributed knowledge-based system \( DS \), the query rough-answering system for a site \( k \) will convert the query to its equivalent \( DNF \) form. To visualize the case, assume that the site \( k \) is represented by Table 3 and the knowledge-base \( KB_k \) contains two rules: \([d_1, e_1, f_1 \ast e_3], [d_2, f_2 \ast e_3, f_1 \ast e_3] \). Assume that \( \sim (f_3 \ast k_2) + d_1 \ast k_2 + d_2 \ast e_2 \) is a query submitted to the site \( k \). Its equivalent term in \( DNF \) is equal to: \( f_1 + f_2 + k_1 + d_1 \ast k_2 + d_2 \ast e_2 \). Rules describing values of attributes \( d_1, d_2 \) are in a local knowledge-base \( KB_k \). Query rough-answering system for a site \( k \) is using both rules from \( KB_k \) to evaluate the query. We get:

\( M_k(f_1 + f_2 + k_1 + d_1 \ast k_2 + d_2 \ast e_2) = M_k(f_1) \cup M_k(f_2) \cup M_k(k_1) \cup \{M_k(d_1) \cap M_k(k_2)\} \cup \{M_k(d_2) \cap M_k(e_2)\} \)

where:

\( M_k(d_1) = (J_k(e_1), J_k(e_1 + f_1 \ast e_3)) \),
\( M_k(d_2) = (J_k(f_2 \ast e_3), J_k(f_2 \ast e_3 + f_1 \ast e_3)) \),
\( M_k(f_1) = (J_k(f_1), J_k(f_1)) \),
\( M_k(f_2) = (J_k(f_2), J_k(f_2)) \), etc.

Let us adopt the following set \( A_G \) of axioms schemata:

- Substitutions of the axioms of distributive lattices for terms and the axioms of equality
  - \( \sim (w, w) \ast (w, w) = 0 \) for any constant \( w \)
  - \( \sim (w, w) + (w, w) = 1 \) for any \( w \in V_k \)
  - for each \( w \in V_k \) there is a subset \( \{w_1, w_2, ..., w_n\} \) of \( V_k \) such that
  - \( \sim (w, w) = (w_1, w_1) + (w_2, w_2) + ... + (w_n, w_n) \)
  - for any term \( t \),
    - \( \sim 0 = 1, \sim 1 = 0 \),
    - \( 1 + t = 1, 1 \ast t = t \),
    - \( 0 \ast t = 0, 0 + t = t \),
    - \( \sim (t) = t \)
  - for any \( w \not\in V_k \),
    - \( (w, w) = (\sum \{t : (w, t, s) \in KB_k\}, \sum \{t + s : (w, t, s) \in KB_k\}) \)
    - \( \sim (w, w) = (\sum \{t : (\sim w, t, s) \in KB_k\}, \sum \{t + s : (\sim w, t, s) \in KB_k\}) \)
  - for any terms \( t_1, t_2 \),
\[ \sim (t_1 + t_2) = (\sim t_1) * (\sim t_2) \]
\[ \sim (t_1 * t_2) = (\sim t_1) + (\sim t_2) \]

- 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 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 \(t_2\).

**Theorem 2** (Completeness Theorem)

For any \(L(DS)\)-formula \(\alpha, A \vdash \alpha\) iff \(A \models \alpha\).

Our Query Rough-Answering System is replacing a global query by its equivalent DNF form. Next, it finds any locally unknown attributes in the expression and contacts all the potential servers to obtain rules for them.

### 4 Conclusion

This paper presents a methodology and theoretical foundations of a query answering system (QAS) for a distributed knowledge-based system (DKBS) which is partially implemented at UNC-Charlotte on a cluster of SPARC 2 workstations. This approach is different from solving queries on a conventional distributed relational database in the sense that it uses rough-rules to resolve unknown attributes. The rough-rules are obtained from a remote server who has a database containing those attributes and is willing to serve the client machine answering a query containing unknown attributes. The communication protocol used between the client and the server machine is UDP/IP instead of TCP/IP. UDP is used over TCP because the packets exchanged in our case are always very small and do not require fragmentation. The client program calls a remote procedure (RPC) to obtain the rules from the server. Server executes the remote procedure whenever the client calls it. The RPC has an advantage over that of using Socket or TLI because the former hides all necessary communication details and is simple to use.

**References**


