A Bucket-Based Approach to Query Rewriting Using Views in the Presence of Inclusion Dependencies*

Qingyuan Bai

School of Computing Science, Faculty of Mathematics and Computer Science, Fuzhou University, Fuzhou, Fujian, 350002, P.R.China
Email: baiqy@fzu.edu.cn or qqyybb@yahoo.com

Jun Hong

School of Computer Science, Queen's University Belfast, Belfast, BT7 1NN, UK
Email: j.hong@qub.ac.uk

Michael F. McTear and Hui Wang

School of Computing and Mathematics, University of Ulster, Newtownabbey,
Co. Antrim, Belfast, BT37 0QB, UK
Email: {mf.mctear, h.wang}@ulster.ac.uk

A number of algorithms based on the use of either buckets or inverse rules have been proposed to address the problem of query rewriting using views. Some inverse rule-based algorithms have considered this problem in the presence of inclusion dependencies. However, no bucket-based algorithms have considered the influence from inclusion dependencies, resulting in missing some query rewritings under this condition. In a bucket-based algorithm, if a view does not contain any subgoals of a query, then the algorithm cannot form any bucket over the view. In this paper, we utilize inclusion dependencies to overcome this deficiency. We present two novel algorithms in the buckets framework. In the first algorithm, we apply a chase procedure/rule to a query to get a set of equivalent or contained queries relative to inclusion dependencies, and then generate rewritings for each of the revised queries. In the second algorithm, we apply a chase procedure/rule to such a view that does not contain subgoals of a query but still contains subgoals to which the chase procedure/rule can be applied. We prove that both algorithms can find a maximally-contained rewriting relative to inclusion dependencies. Hence, the problem of missing rewritings in the previous bucket-based algorithms is avoided.

ACM Classification: H.2.4 (Information Systems): Systems—query processing; H.2.5 (Information Systems): Heterogeneous Databases

1. INTRODUCTION

With the presence of more and more data sources on the Internet, data integration systems over the Web have been continuously growing in the last few years. These systems aim to support seamless access to autonomous, heterogeneous data sources. A mediator-based data integration system has

* Supported by the scholarship of Vice-Chancellor of University of Ulster in UK and the Research Foundation of Fuzhou University in China

Manuscript received: 16 February 2005
Communicating Editor: Mirka Miller

been proposed by Wiederhold (1992) to deal with the autonomy and heterogeneity of data sources. In a mediator-based data integration system, there are two types of schemas, i.e., a mediated schema and a set of data source schemas. The mediated schema is used to make queries and describe the contents of data sources. Because the actual data is stored in the data sources, we need to reformulate user queries over the mediated schema into new queries over the data source schemas. This process is called query reformulation which is an important step of query processing in a data integration system. In general, there are two main approaches to describing the relationships between a mediated schema and data source schemas, i.e., Global As View (GAV for short) and Local As View (LAV for short). As stated by Levy (2001), in the GAV approach a mediated schema is defined over data source schemas, while in the LAV approach, data sources are defined over a mediated schema. The LAV approach is suitable for a data integration system in a dynamic environment because it is easy to delete/add a data source from/to a data integration system without making major changes in a mediated schema. Query reformulation in the LAV approach is also called query rewriting using views which has recently received considerable attention because of its relevance to a wide variety of database applications. In this paper we take the LAV approach.

Two assumptions can be made on view definitions, i.e., Closed World Assumption and Open World Assumption (OWA for short). As a result, there are two types of query rewritings, i.e., equivalent rewritings and contained rewritings. The former can provide the same answers as the original query while the latter can provide only a subset of the answers to the original query. Usually, in the Web environment, data sources are autonomous and heterogeneous. We take the OWA assumption only since we are more interested in finding the set of contained rewritings whose union, called a maximally-contained rewriting, can provide the best possible answers to a user query.

The problem of query rewriting using views is: given a query Q and a set of views $\mathbf{V}=\{V_1,\ldots,V_n\}$, a rewriting of Q is a query expression whose body predicates are from $\mathbf{V}$ only. There have been several rewriting algorithms, such as the bucket algorithm (Levy *et al*, 1996a; 1996b), the inverse rule algorithms (Qian, 1996; Duschka and Genesereth, 1997), the Shared Variables Bucket (SVB for short) algorithm (Mitra, 2001), and the MiniCon algorithm (Pottinger and Levy, 2000). These algorithms are based on the use of either inverse rules or buckets. Some algorithms in the inverse rules framework (Grant and Minker, 2002; Gryz, 1999) have considered the problem of query rewriting using views in the presence of inclusion dependencies (INDs for short) in a mediated schema. However, no bucket-based algorithms have considered the influence from INDs, resulting in missing some query rewritings in the presence of inclusion dependencies as shown in the following example.

Example 1 (from Gryz, 1999): *Assume that a mediated schema consists of the following relations: patient(name, dob, address, insurer), procedure(patient_name, physician_name, procedure_name, time), insurer(company, address, phone), event(event_name, description, patient_name, location).*

Assume that all procedures with patients' names are also recorded in the event table, that is,

procedure(procedure _name, patient_name) $\subseteq$ *event(event_name, patient_name).*

We use the rules in the Datalog query language to represent conjunctive queries and views as used in Ullman (1997). The left hand side of a rule is called the head of the rule while the right hand side is called the body of the rule. Each term in the body is called a subgoal.

Suppose that there are two views (For simplicity, we assume that a view represents a data source throughout this paper.):

$V_1(X,Z,W)$:- procedure(X,U_1,Z,W).

$V_2(X,U,S)$:- patient(X,S_1,U,S_2), insurer(S_2,S,S_3).

Assume that there is a query:

$Q(E)$:- patient(W_0,W_1,W_2,W_3), event(E,W_4, W_0,W_5).

The MiniCon algorithm cannot generate any rewriting of Q because no view contains the relation event. However, as shown later, we can generate the following rewriting of Q by making use of the given IND:

$Q'(E)$:- $V_1(W_0,E,W)$, $V_2(W_0,W_2,S)$.

The key idea in this paper is to introduce a chase procedure/rule of INDs and apply it to a query or views. We present two novel algorithms in the buckets framework. In the first algorithm, we apply the chase procedure/rule to a query to get a set of equivalent or contained queries relative to inclusion dependencies, and then generate rewritings for each of the revised queries. In the second algorithm, we apply the chase procedure/rule to such a view that does not contain any subgoals of a query but still contains subgoals to which the chase procedure/rule can be applied.

Throughout the rest of the paper, we make the following assumptions:

1. Because the problem of interaction between functional dependencies and inclusion dependencies is undecidable, we consider only the influence of inclusion dependencies itself.
2. Each view definition should satisfy inclusion dependencies in a mediated schema, which follows from the papers of Gryz (1999), Grant and Minker (2002), and Duschka *et al* (2000). Moreover, we do not consider how to maintain inclusion dependencies, how to compute the closure of inclusion dependencies, etc., which are the topics in the theories of relational database systems.
3. We consider the queries without comparisons only. Our approaches can be applied to the queries containing comparisons as long as we make a test of comparison implication before forming a bucket.

The rest of the paper is organized as follows. In the next section, the preliminaries for the problem of query rewriting using views and inclusion dependencies are given. In Section 3 we give a brief overview of related work on the algorithms for query rewriting using views in the LAV approach. In Section 4 we present two bucket-based algorithms for the problem of query rewriting using views in the presence of inclusion dependencies. In Section 5 we analyze our algorithms with respect to computational complexity and correctness. In Section 6 some experimental results are given. Finally, we conclude the paper.

2. PRELIMINARIES

2.1 Query Containment and Query Rewriting Using Views

A conjunctive query without any arithmetic comparisons has the form:

$$Q(\bar{X}) :- R_1(\bar{X}_1),\ldots,R_k(\bar{X}_k)$$

where $R_1(\bar{X}_1),\ldots,R_k(\bar{X}_k)$ are database relations in a mediated schema. We require that the query be safe, i.e., $\bar{X} \subseteq \bar{X}_1 \cup \ldots \cup \bar{X}_k$. The variables in $\bar{X}$ are distinguished variables, and others are existential ones. An existential variable is a shared variable if it appears in more than one subgoal. A view is a named query.

We say that a query Q_1 is contained in the Q_2, denoted by $Q_1 \subseteq Q_2$, if the answer to Q_1 is a subset of the answer to Q_2 for any database instance. Containment mapping provides a necessary and sufficient condition for testing query containment of conjunctive queries. A mapping φ from $Vars(Q_2)$ to $Vars(Q_1)$ is a containment mapping if (1) φ maps every subgoal in the body of Q_2 to a subgoal in the body of Q_1, and (2) φ maps the head of Q_2 to the head of Q_1. The query Q_2 contains Q_1 if and only if there is a containment mapping from Q_2 to Q_1. The query Q_1 is equivalent to Q_2 if and only if $Q_1 \subseteq Q_2$ and $Q_2 \subseteq Q_1$.

Note that the views are not assumed to contain all the tuples in their definitions since the data sources are autonomous. Moreover, we cannot always find an equivalent rewriting of the query using views because data sources may not contain all of the answers to the query. Instead, we consider a maximally-contained rewriting (Levy, 2001).

Definition 1 (Contained rewriting) Q' is a contained rewriting of a query Q using the views $\mathbf{V}=V_1,...,V_n$ with respect to a query language $\mathbf{L}$ if the expansion of Q', denoted by Q'^{EXP}, is contained in Q, i.e., $Q'^{EXP} \subseteq Q$, where Q'^{EXP} is obtained by replacing all the views in Q' by their definitions, and existential variables in a view definition are replaced by fresh variables.

Definition 2 (Maximally-contained rewriting) Q' is a maximally-contained rewriting of a query Q using the views $\mathbf{V}=V_1,...,V_n$ with respect to a query language $\mathbf{L}$ if

(1) Q' is a contained rewriting of Q, i.e., $Q'^{EXP} \subseteq Q$,
(2) there is no other rewriting Q_1 of Q, such that $Q_1^{EXP} \subseteq Q$ and $Q'^{EXP} \subset Q_1^{EXP}$.

2.2 Inclusion Dependencies:

Definition 3 (Inclusion Dependencies (Abiteboul *et al*, 1995))
An m-ary inclusion dependency (IND) is a formal statement of the form $R[A_1,..., A_m] \subseteq S[B_1,..., B_m]$ where R and S are relation schemes, $A_1,..., A_m$ and $B_1,..., B_m$ are attributes in R and S respectively. A database obeys the IND $R[A_1,..., A_m] \subseteq S[B_1,..., B_m]$ if for every subtuple $<a_1,...,a_m>$ that occurs in columns $A_1,..., A_m$ of some tuple in relation R, there is a tuple of relation S that contains $<a_1,...,a_m>$ in columns $B_1,..., B_m$.

When the relations in a mediated schema satisfy a set Δ of inclusion dependencies, we define our notion of query containment as query containment relative to Δ (Gryz, 1999).

Definition 4 (Query Rewriting in the Context of Inclusion Dependencies)
Query Q_1 is contained in query Q_2 relative to Δ, denoted by $Q_1 \subseteq_\Delta Q_2$, if for each database instance D satisfying the inclusion dependencies in Δ, $Q_1(D) \subseteq Q_2(D)$.

Definition 5 (Maximally-Contained Rewriting in the Context of Inclusion Dependencies)
Q' is a maximally-contained rewriting of a query Q using views $V_1,...,V_n$ with respect to a query language $\mathbf{L}$ relative to Δ, if

(1) Q' is a contained rewriting of Q relative to Δ, i.e., $Q'^{EXP} \subseteq_\Delta Q$,
(2) there is no other rewriting Q_1 of Q, such that $Q_1^{EXP} \subseteq_\Delta Q$ and $Q'^{EXP} \subset_\Delta Q_1^{EXP}$.

3. RELATED WORK

3.1 Algorithms Based on the Use of Buckets

A bucket-based algorithm consists of two steps. In the first step a single-subgoal bucket is created for a subgoal R in a query Q over a view V if the following condition (C_1) is satisfied.

(C_1) Any distinguished variables in Q should be mapped to distinguished variables in R. An existential (non-shared) variable in Q could be mapped to either a distinguished variable or an existential variable in R.

In this case, we say V covers the subgoal R in Q. When a shared variable within R in Q is mapped to a non-distinguished variable in V, it needs to form a bucket containing multiple subgoals which is based on the following condition (C_2).

(C_2) A set of subgoals of Q containing a shared variable should be covered by a view.

In the second step, rewritings are generated by combining a view from each bucket.

In fact, the bucket algorithm (Levy *et al*, 1996a; 1996b) considers forming a set of single-subgoal buckets only. As a result, rewritings need to be verified using the containment test. Compared with the bucket algorithm, the SVB algorithm (Mitra, 2001) creates the above two types of buckets. The MiniCon algorithm (Pottinger and Levy, 2000) proceeds in the same way. The difference between the MiniCon algorithm and the SVB algorithm is that the former introduce a head homomorphism on a view so that it can generate more rewritings in some cases than the latter.

Now we show the MiniCon algorithm in detail because we are going to extend the algorithm.

Example 2 Suppose that there are three views.

$V_1(A, C)$:- $r(A,C)$, $s(W_1,C)$.
$V_2(B)$:- $s(B,U_2)$.
$V_3(A)$:- $r(A,W_3)$, $s(W_3,U_3)$.

A query is made as follows:

$Q(X)$:- $r(X,K)$, $s(K,J)$.

Hereafter, we denote the variables of a query Q and a view V by Vars(Q) and Vars(V) respectively. We use subgoals(Q) to refer to the set of subgoals in Q.

In the first step, the MiniCon algorithm tries to form a set of buckets named as MiniCon Descriptions (MCDs for short) for each subgoal of Q over V_i, i=1,2,3. An MCD over a view V is a tuple of the form of (h_C, $V(Y)_C$, φ_C, G_C), where h_C is a head homomorphism on V, $V(Y)_C$ is the result of applying h_C to V, φ_C is a partial mapping from Vars(Q) to h_C(Vars(V)), and G_C is a set of subgoals of Q which are covered by $h_C(V)$ associated with φ_C. A head homomorphism h_C on a view V is a mapping h_C from Vars(V) to Vars(V) that is the identity on the existential variables, but may equate distinguished variables.

Two types of MCDs are considered. An MCD containing a single subgoal is based on the condition (C_1) while an MCD containing multiple subgoals is based on the condition (C_2). In this example we can form three MCDs, two of which contain a single subgoal while the third contains a set of subgoals.

h_C(homomorphism)	V(selected view)	φ_C(mappings)	G_C(subgoal of Q)
$A \rightarrow A$, $C \rightarrow C$	$V_1(X, K)$	$X \rightarrow A$, $K \rightarrow C$	r(X,K)
$B \rightarrow B$	$V_2(K)$	$K \rightarrow B$, $J \rightarrow U_2$	s(K,J)
$A \rightarrow A$	$V_3(X)$	$X \rightarrow A$, $K \rightarrow W_3$, $J \rightarrow U_3$,	r(X,K), s(K,J)

Table 1: Three MCDs formed in Example 2

In this paper, we simply treat a homomorphism as the identity operator, i.e., an identity on all the variables of V; otherwise it may cause violation with INDs.

After all the MCDs are created, the MiniCon algorithm generates all possible query rewritings by combining views from the MCDs as follows.

Suppose that there are k MCDs, denoted by C_i, i=1,...,k, such that $i \neq j$, $G_i \cap G_j = \varnothing$, and $G_1 \cup G_2 \cup ... \cup G_k$ = subgoals(Q), where G_i is the field G_C in C_i, then the conjunction of views from C_i, i=1,...,k, is a contained rewriting of Q. In this example we can generate two rewritings of Q as follow:

$Q'(X) \text{ :- } V_1(X, K), V_2(K).$
$Q''(X) \text{ :- } V_3(X).$

There are two main advantages of the MiniCon algorithm over the bucket algorithm. The first advantage is that the shared variables of Q are considered when unifying. The second is that none of the obtained rewritings is redundant.

3.2 Algorithms Based on the Use of Inverse Rules

The key idea of the inverse rules algorithms is to construct a set of rules called inverse rules that invert the view definitions. In the inverse rules, the existential variables in the view definitions are replaced with Skolem functions in the heads of the inverse rules. The rewriting of a query is simply the composition of the query and the inverse rules of views using either the transformation method (Duschka and Genesereth, 1997), or the u-join method (Qian, 1996), or the resolution method (Grant and Minker, 2002).

Gryz (1999), Grant and Minker (2002) have discussed the problem of query rewriting using views in the presence of inclusion dependencies. In (Gryz, 1999), a chase procedure/rule for inclusion dependencies in (Johnson and Klug, 1984) is introduced. First, Gryz minimized Q and got a minimal query Q' which is Δ-equivalent to Q given a query Q and a set Δ of INDs. Second, for each subgoal p_i (i=1,2,…,n) in Q', Gryz tried to get a set of Δ-equivalent or Δ-contained subgoals ps_{ij} (i=1,2,…,n, j=1,2,…,m). Then a set of Δ-equivalent or Δ-contained queries of Q' are generated by combining Δ-equivalent or Δ-contained subgoals from the sets ps_{ij} (i=1,2,…,n, j=1,2,…,m). Finally, for each of Δ-equivalent or Δ-contained queries of Q', an inverse rule algorithm is used to generate the Δ-equivalent or Δ-contained rewritings of Q'. These rewritings are also Δ-equivalent or D-contained rewritings of Q.

In Grant and Minker (2002), an IND is described by a Clark Completion rule. For example, the IND in Example 1 can be described as:

$\text{event}(X'_3, f_1, X'_1, g) \leftarrow \text{procedure}(X'_1, X'_2, X'_3, X'_4).$

This rule is then used, associated with a set of Clark Completion rules of the given views and query, to generate query rewritings by the resolution method.

In this paper, we assume that a given query and a given set of views have all been minimized, because the minimization of a query is irrelevant to the issue of query rewriting.

4. QUERY REWRITING USING VIEWS IN THE PRESENCE OF INCLUSION DEPENDENCIES

In the context of databases, there exist inclusion dependencies in a database schema, which provide special relationships between relations. As data sources in data integration systems are defined over the relations in a mediated schema, these dependencies also reveal some relationships between data sources. Thus, the topic in this paper has practical significance.

As stated in Section 2, the conditions (C_1) and (C_2) are basic criteria for forming an MCD in the MiniCon algorithm. We note that the MiniCon algorithm fails to form an MCD for a given query Q over a view V if any of the following cases is true:

(1) V contains no subgoal in Q, e.g., no MCD is formed over V_1 in Example 1.
(2) There is a violation of the condition (C_1).
(3) There is a violation of the condition (C_2).

If the case (1) above occurs, we apply a chase procedure/rule of INDs to a query or views so that we can generate all rewritings.

4.1 Chase Procedure/Rule of Inclusion Dependencies

In the next section, we need to be able to refer to not only the set Δ of INDs as stated for a given database, but also its closure Δ^* which is defined as the set of all the INDs implied by Δ. Henceforth, we assume that the set Δ is closed under its consequences (that is $\Delta = \Delta^*$). Note that conjunctive views are named conjunctive queries. Therefore, the following definitions are applicable to views too.

Definition 6 The Chase$_\Delta$ Procedure/Rule (Johnson and Klug, 1984; Gryz, 1999)

The Chase$_\Delta$ Procedure:
If an IND $R[A_1,..., A_m] \subseteq S[B_1,..., B_m]$ is in Δ, and $R(X_1,...,X_M)$, $M \geq m$, is a subgoal of a query Q, and $S(Y_1,...,Y_N)$, $N \geq m$, does not appear in Q, we say this IND is applicable. Each step of a chase procedure consists of an application of the chase rule to a given IND and a subgoal $R(X_1,...,X_M)$ to which it is applicable.

The Chase$_\Delta$ Rule:
Let an IND and a subgoal R be as above. The chase$_\Delta$ rule is to add a new subgoal $S(Y_1,...,Y_N)$ to Q, where $S[B_1,..., B_m] = R[A_1,..., A_m]$ and for S[C], $C \neq B_i$, $1 \leq i \leq m$, is a new variable that does not appear in Q.

Definition 7 Chase$_\Delta$ Reachable (Johnson and Klug, 1984; Gryz, 1999)
Let Q and Q' be queries. Q is chase$_\Delta$ reachable from Q', denoted by Q=chase$_\Delta$(Q'), if $Q \equiv Q'$ or there exist $Q_1,..., Q_n$, such that:
(1) $Q' \equiv Q_1$,
(2) Q_{i+1} is derived from Q_i, $1 \leq i \leq n-1$, by applying the above single chase$_\Delta$ rule to it, and
(3) $Q \equiv Q_n$.

Definition 8 Scope of a Subgoal (Gryz, 1999)
Let $Q(X)$:- $S_1(X_1),\ldots,S_k(X_k)$ be a query. The scope of the subgoal S_i, $i=1,\ldots,k$, is the set of attributes in S_i corresponding to distinguished variables, constants, and join attributes in Q.

The concept of scope plays an important role in the application of a chase$_\Delta$ procedure/rule to a query or a view. Let us have a look at the following examples.

Example 3 Suppose that we have views and a query as follows:
(a) $V_1(X)$:- R(X, y). // scope(R) = {X}
$V_2(X)$:- S(X, y). // scope(S) = {X}
Q(X) :- S(X, y).
An IND is $\{R[X] \subseteq S[X]\}$.

It is obvious that $V_1(X)$ can provide part of the answers for Q even though it does not contain the subgoal S. This is achieved by making use of the given IND, i.e., one query rewriting is:

Q''(X) :- V_1(X).

Using any previous bucket-based algorithm, we can get the rewriting:

Q'(X) :- V_2(X).

However, we still need to consider how to get Q''(X) because the union of Q'(X) and Q''(X) is a maximally-contained rewriting of Q relative to INDs.

(b) V(Y) :- R(x, Y). // scope(R) = {Y}
Q(X) :- S(X, y). // scope(S) = {X}
An IND is {R[X] $\subseteq$ S[X]}. No rewriting can be obtained because scope(R) $\cap$ scope(S) = $\emptyset$.

(c) V(X) :- R(X, y, z). // scope(R) = {X}
Q(X) :- S(X, y, z, k), T(X, y). // scope(S) = {X, y}
An IND is {R[X, Y] $\subseteq$ S[X, Y]}. Also no rewriting can be obtained because V cannot cover the conjunction of R and T.

In Gryz (1999), an algorithm for minimizing a query is given. A query Q is minimal if and only if Q does not contain both R and S if R and S appear in a IND: R[**A**] $\subseteq$ S[**B**] or in the forms of R[**A**] $\subseteq$ T[**B**], T[**B**] $\subseteq$ S[**C**]. Due to transitivity of INDs, if R[**A**] $\subseteq$ T[**B**] and T[**B**] $\subseteq$ S[**C**], we have R[**A**] $\subseteq$ S[**C**]. As stated in Section 3, we assume that the given query is a minimal query relative to Δ. Because a view is a named query, we also assume that each view is a minimal query relative to Δ.

4.2 The BFIND_1 Algorithm

In Gryz (1999), an algorithm for finding a set of Δ-contained or Δ-equivalent queries of a given query is given as follows.

Algorithm find_contained_query(Q, Δ) (or find_equiv_query (Q, Δ))

Input: Q and Δ.

Output: A set of Δ-contained (or Δ-equivalent) queries, $Q_1,\ldots,Q_M$, of Q.

```
1:  Body(Q):=$S_1,\ldots, S_k$.
2:  For i=1 to k do
3:      A = scope($S_i$)
4:      Atoms$_i$ ← { $S_i$ }
5:      For each R[B] $\subseteq$ $S_i$[A] $\in \Delta$ (or R[B] $\equiv$ $S_i$[A] $\in \Delta$ ), then // B = scope(R)
6:          For each attribute C of R such that C $\notin$ B
7:              R[C]:= new_var; // new_var means new variables
8:          Endfor;
9:          Atoms$_i$ ←Atoms$_i$ $\cup$ {R}
10:     Endfor;
11: Endfor.
12: $Q_j$ = Combine(Atoms$_i$, i=1,…,k), j=1,…, M;
13: Return $Q_1,\ldots,Q_M$.
```

In the BFIND_1 algorithm, we also use the algorithm find_contained_query(Q, Δ) (or find_equiv_query (Q, Δ)) to find a set of Δ-contained or Δ-equivalent queries of a query Q. We then apply the MiniCon algorithm to generate rewritings for each Q_i, i=1,…,M.

Algorithm BFIND_1: Query Rewriting Using Views in the Presence of Inclusion Dependencies

Input: A conjunctive query Q, a set of the views $V_1,V_2,...,V_n$, a set of inclusion dependencies Δ in the form of $R[A_1,..., A_m] \subseteq S[B_1,..., B_m]$.

Output: Q_Δ', a maximally-contained rewriting of Q relative to inclusion dependencies Δ.

Method:

Step 1: Find a set of Δ-equivalent or Δ-contained queries of Q.

Call Algorithm find_contained_query(Q, Δ) (or find_equiv_query (Q, Δ))

Step 2: Generate a set of rewritings of each of queries obtained in Step 1 by using the MiniCon algorithm.

Let us continue with Example 1. First, we can get a set of Δ-contained or Δ-equivalent queries of Q as follows:

$Q_1(E)$:- patient(W_0, W_1, W_2, W_3), event(E, W_4, W_0, W_5).
$Q_2(E)$:- patient(W_0, W_1, W_2, W_3), procedure(E, W_7, W_0, W_8).

We then use the MiniCon algorithm to generate the rewritings of Q_1 and Q_2, respectively, over V_1 and V_2. We form two MCDs for the subgoals of Q_2 over V_1 and V_2 as follows:

h_C	V	φ_C	G_C
$X \to X, Z \to Z, W \to W$	$V_1(E,W_0,W_8)$	$E \to X, W_7 \to U_1, W_0 \to Z, W_8 \to W$	procedure
$X \to X, U \to U, S \to S$	$V_2(W_0,W_2,S)$	$W_0 \to X, W_1 \to S_1, W_2 \to U, W_3 \to S_2$	patient

Table 2: MCDs for Q_2 over V_1 and V_2

We can get a rewriting of Q2 as follows:

$Q_\Delta(E)$:- $V_1(E,W_0,W_8)$, $V_2(W_0,W_2,S)$.

No rewriting is obtained for Q_1 over V_1 and V_2 because neither V_1 nor V_2 contains the subgoal event. Thus, the above $Q_\Delta(E)$ is a maximally-contained rewriting of Q.

4.3 The BFIND_2 Algorithm

In the BFIND_2 algorithm, we apply the $chase_\Delta$ procedure/rule to views which contain $chase_\Delta$ reachable subgoals, instead of the given query.

Continue with Example 1. Even though V_1 does not contain the subgoal *event*, it contains a $chase_\Delta$ reachable subgoal *procedure*. We apply the $chase_\Delta$ procedure/rule to V_1 and get a Δ-equivalent view:

V'_1(X, Z, W) :- procedure(X, U_3, Z, W), event(Z, D, X, L).

We form the following MCDs over V'$_1$ and V$_2$:

h_C	V	φ_C	G_C
X→X, Z→Z, W→W	V'$_1$(W$_0$,E,W)	E→Z, W$_4$→D, W$_0$→X, W$_5$→L	event
X→X, U→U, S→S	V$_2$(W$_0$,W$_2$,S)	W$_0$→X, W$_1$→S$_1$, W$_2$→U, W$_3$→S$_2$	patient

Table 3: The MCDs for the subgoals in Q over V'$_1$ and V$_2$ in Example 1

We then generate a rewriting of Q using V'$_1$ and V$_2$ as follows:

Q'_Δ(E):- V'$_1$(W$_0$,E,W), V$_2$(W$_0$,W$_2$,S).

From the MiniCon algorithm, we know that Q'_Δ(E) is a maximally-contained rewriting of Q using V'$_1$ and V$_2$. However, from Definitions 6 and 7, we know that V'$_1$(X,Z,W) is Δ-equivalent to V$_1$(X,Z,W). Thus, the following query obtained by replacing V'$_1$ with V$_1$ is a maximally-contained rewriting of Q relative to inclusion dependencies.

Q_Δ(E):- V$_1$(W$_0$,E,W), V$_2$(W$_0$,W$_2$,S).

Now we formally present the BFIND_2 algorithm.

Suppose that there is an IND: R[A$_1$,…,A$_m$] ⊆ S[B$_1$,…,B$_m$], and for the sake of simplicity, $A_i = B_i$, i=1,…,m, i.e., R[**C**] = S[**C**], **C** = [A$_1$,…,A$_m$]. Let V be a view containing a subgoal R which appears in an IND on the left hand side. We say that V contains a chase$_\Delta$ reachable subgoal R. When a given query Q contains a subgoal S in the right hand side of an IND, we need to apply a chase$_\Delta$ procedure/rule to all views that contain R. We get chase$_\Delta$(V) by adding S to the body of V. However, in order to make sure that chase$_\Delta$(V) can cover S of Q, the following conditions should be satisfied:

(C$_3$): ***A*** *⊆* ***C****, where* ***A*** *= scope(S) = {a set of attributes in S corresponding to distinguished variables, constants, and shared variables in Q}.*
(C4): ***A*** *⊆* ***B****, where* ***B*** *= scope(R) = {a set of attributes in R corresponding to distinguished variables, constants, and shared variables in V}.*

In fact, this condition is also needed in the algorithm find_contained_query (or find_equiv_query) in Gryz (1999). Otherwise, it is not guaranteed that the obtained queries are Δ-equivalent or Δ-contained queries of the original query.

If the condition (C$_3$) is not satisfied, then it is not necessary to consider making use of INDs because some information of Q will be lost. If the condition (C$_4$) is not satisfied, it does not make sense to apply a chase procedure/rule to views because even though chase$_\Delta$(V) contains the subgoal S, the condition (C$_1$) or (C$_2$) of a containment mapping cannot be satisfied.

The BFIND_2 algorithm consists of two stages described as follows.

Algorithm BFIND_2: Query Rewriting Using Views in the Presence of Inclusion Dependencies

Input: A conjunctive query Q, a set of the views V$_1$,V$_2$,...,V$_n$, a set of inclusion dependencies Δ in the form of R[A$_1$,..., A$_m$] ⊆ S[B$_1$,..., B$_m$].

Output: Q_Δ', a maximally-contained rewriting of Q relative to inclusion dependencies Δ.

Method:

Step 1: Forming the MCDs using the MiniCon Algorithm
For each subgoal of Q, we try to form a MCD over views. If a view contains the chase_Δ reachable subgoals and the conditions (C_3) and (C_4) are satisfied, we apply a chase_Δ procedure/rule to the view. We then try to form a set of MCDs over the revised view.

Step 2: Generating a maximally-contained rewriting using the obtained MCDs.
The MCDs formed in Step 1 are used to generate query rewritings in the same way as in the MiniCon algorithm. The union of all the query rewritings generated is a maximally-contained rewriting relative to inclusion dependencies Δ.

5. COMPUTATIONAL COMPLEXITY AND CORRECTNESS OF OUR ALGORITHMS

5.1 Computational Complexity of Our Algorithms

The computational complexity in Step 1 of the BFIND_2 algorithm is the same as the MiniCon algorithm when forming the MCDs over the views to which no chase_Δ procedure/rule needs to be applied. The computation in applying a chase_Δ procedure/rule to a view is to check whether the two conditions (C_3) and (C_4) are satisfied. In the worst case, it is $|Q|*|\Delta|*n$, where $|Q|$, $|\Delta|$, and n are the number of subgoals of Q, the number of inclusion dependencies in Δ and the number of views respectively. The computation in Step 2 of the BFIND_2 algorithm is the same as the MiniCon algorithm. Thus, the total increased computation has only polynomial time complexity, i.e., $|Q|*|\Delta|*n$.

In Gryz (1999), the computational complexity of the algorithm find_contained_query (or find_equiv_query) is $|Q|*|\Delta|$. However, it needs to consider query rewriting for each of the Δ-equivalent or Δ-contained queries over a set of views. In the worst case, the total cost of computing a maximally-contained rewriting of a query using views in the presence of INDs in (Gryz, 1999) is M times NP-complete, where M is the number of Δ-equivalent or Δ-contained queries of Q. The computational complexity of the BFIND_1 algorithm is the same as this.

5.2 Correctness of Our Algorithms

We now prove that our algorithms are correct in terms of soundness and completeness, which is based on the following theorem.

Theorem 1 (Johnson and Klug, 1984) Let Q and Q' be queries. $Q \subseteq_\Delta Q'$ (or $Q \equiv_\Delta Q'$) if and only if $\text{chase}_\Delta(Q) \subseteq \text{chase}_\Delta(Q')$ (or $\text{chase}_\Delta(Q) \equiv \text{chase}_\Delta(Q')$).

The correctness of the BFIND_1 algorithm follows from the result in Gryz (1999) and the MiniCon algorithm. The proof is given as follows. In Step 1 of the BFIND_1 algorithm, we get a set of Δ-equivalent or Δ-contained queries of Q, denoted by Q_i, i=1,…,M. The proof is given in Gryz (1999). For each Q_i, i=1,…,M, we generate a set of contained rewritings using the MiniCon algorithm, denoted by Q'_{ij}, i=1,…,M, j=1,…,N_i. For any i_0, $1 \le i_0 \le M$, the union of Q'_{i0j}, j=1,…,N_{i0}, is a maximally-contained rewriting of Q_{i0}. Thus, the union of Q'_{ij}, i=1,…,M, j=1,…,N_i, is a maximally-contained rewriting of Q relative to INDs.

In the BFIND_2 algorithm, if a view V contains a chase_Δ reachable subgoal R and the conditions (C_3) and (C_4) are satisfied, we can apply a chase_Δ procedure/rule (We assume that an IND is: $R[\mathbf{C}] \subseteq S[\mathbf{C}]$) to V, denoted by $V'=\text{chase}_\Delta(V)$. The head of V' is the same as the head of V. In the body of V', there is a conjunction of R and S. From the Theorem 1 we have that $V' \equiv_\Delta V$. For the sake of simplicity, we assume that when using the BFIND_2 algorithm, we apply a $chase_\Delta$ procedure/rule to the views $V_i, \ldots, V_{i+j}$ and generate an Δ-rewriting in the form of:

$Q'(X) :- V'_i,\ldots,V'_{i+j}, V_R.$

where $V'_i = chase_\Delta(V_i),\ldots,V'_{i+j} = chase_\Delta(V_{i+j})$, V_R is a set of views to which no $chase_\Delta$ procedure/rule needs to be applied. Q' is a maximally-contained rewriting of Q in terms of answering Q using $V'_i, \ldots, V'_{i+j}$, and V_R, which follows from the MiniCon algorithm. After replacing $V'_i,\ldots,V'_{i+j}$ with $V_i,\ldots,V_{i+j}$, we have:

$Q''(X) :- V_i,\ldots,V_{i+j}, V_R.$

Let Q'^{exp} and Q''^{exp} be expansions of Q' and Q'' respectively obtained by replacing all the views in Q' and Q'' by their definitions. Because $V'_i = chase_\Delta(V_i),\ldots,V'_{i+j} = chase_\Delta(V_{i+j})$, we have $Q'^{exp} = chase_\Delta(Q''^{exp})$. Thus, we have $Q'^{exp} \equiv_\Delta Q''^{exp}$. Because Q' is a maximally-contained rewriting of Q relative to Δ, Q'' is also a maximally-contained rewriting of Q relative to Δ.

6. EXPERIMENTAL RESULTS

We present some experimental results (from Example 4 to Example 8) to show that the BFIND_2 algorithm can generate all the Δ-equivalent or Δ-contained rewritings of a given Q. Some of the rewritings are generated by the algorithm using inclusion dependencies, and they cannot be generated by any of the existing bucket-based algorithms including the MiniCon algorithm. Table 4 shows the numbers of rewritings generated by the MiniCon algorithm and by the BFIND_2 algorithm. We use JBuilder programming language to implement our algorithm. After a set of views and inclusion dependencies based on the relations in a mediated schema are given, we can make arbitrary queries.

The details of these examples are listed below, along with the explanations of the experimental results.

Example 4

(a)

1. $V_1(X,Y) :- R(X,Y,z_1).$
   $V_2(X) :- T(X,y_2).$
2. $Q(X) :- S(X, y, z), T(X, w).$
3. IND: $\{R[X, Y] \subseteq S[X, Y]\}$

	The MiniCon algorithm	The BFIND_2 algorithm*
Example 4(a)	0	1
Example 4(b)	0	2
Example 5	1	3
Example 6	0	3
Example 7	0	1
Example 8	0	2

Table 4: The numbers of rewritings generated by the MiniCon algorithm and by the BFIND_2 algorithm respectively

* Using the BFIND_1 algorithm, we can also get the same results.

4. We need to apply a chase$_\Delta$ procedure/rule to V_1 and get an MCD for subgoal S of Q over V_1. Another MCD for subgoal R of Q over V2 can be formed without considering IND. Thus, no rewriting is generated by the MiniCon algorithm. But, one rewriting is obtained by the BFIND_2 algorithm as follows:

   Q'$_\Delta$(X) :- V1(X, Y), V2(X).

(b)
1. V_1(X, Y) :- R_1(X, Y, z_1).
   V_2(X, Y) :- R_2(X, Y, z_2, z_3).
   V_3(X) :- T(X, y_2).
2. Q(X) :- S(X, y, z), T(X, w).
3. IND: {R_1[X, Y] $\subseteq$ S[X, Y], R_2[X, Y] $\subseteq$ S[X, Y]}
4. An MCD for subgoal T over V_3 can be formed without considering INDs. In addition we apply a chase$_\Delta$ procedure/rule to V_1 and V_2 respectively, and get other two MCDs for subgoal S of Q over V_1 and V_2 respectively. Thus, no rewriting is generated by the MiniCon algorithm. But, two rewritings are obtained by the BFIND_2 algorithm as follows:

   Q'$_\Delta$(X) :- V_1(X, Y), V_3(X).
   Q''$_\Delta$(X) :- V_2(X, Y), V_3(X).

Example 5
1. V_1(X) :- R_1(X, y_1, z_1), T_1(y_1, z_1).
   V_2(X) :- R_2(X, y_2), T_2(X).
   V_3(K) :- R_3(x_3,y_3,z_3, K), T_3(K).
   V_4(X) :- S(X,y_4).
   V_5(X) :- T(X), T_2(X).
2. Q(X) :- S(X, y), T(X).
3. IND: {R1[X, Y] Õ S[X, Y], R2[X] Õ S[X], R3[X] Õ S[X]}
4. Two MCDs are formed by the MiniCon algorithm. However, we can form two other MCDs by making use of INDs. Thus, there are up to four MCDs formed by the BFIND_2 algorithm. As a result, only one rewriting is generated by the MiniCon algorithm. However, three rewritings are obtained by the BFIND_2 algorithm; two of them are Δ-contained rewritings shown as follows:

   Q'$_\Delta$ (X) :- V_1(X), V_5(X).
   Q''$_\Delta$ (X) :- V_2(X), V_5(X).
   Q'''(X) :- V_4(X), V_5(X).

Example 6
1. V_1(X) :- R_1(X, y_1), S_2(X, y_1). // $\mathbf{B}_1$={X, Y}
   V_2(X) :- R_1(X, y_2), R_2(X, y_2). // $\mathbf{B}_1$={X, Y}, $\mathbf{B}_2$={X, Y}
   V_3(X, Y) :- R_1(X, Y). // $\mathbf{B}_1$={X, Y}
   V_4(X) :- R_1(X, y_4), R_2(X, y_5). // $\mathbf{B}_1$={X}, $\mathbf{B}_2$={X}
   V_5(X) :- S_1(X, y_6), R_2(X, y_6). // $\mathbf{B}_2$={X, Y}
2. Q(X) :- S_1(X, y), S_2(X, y). // $\mathbf{A}_1$={X, Y}, $\mathbf{A}_2$={X, Y}
3. IND: {R_1[X, Y] $\subseteq$ S_1[X, Y], R_2[X, Y] $\subseteq$ S_2[X, Y]}
4. No MCD is formed by the MiniCon algorithm. Except V_4, for V_1, V_2, V_3, and V_5, the conditions (C_3) and (C_4) are satisfied. Therefore, we can form four MCDs over them. Thus, no rewriting

can be generates by the MiniCon algorithm. However, the BFIND_2 algorithm can generate three D-contained rewritings of Q as follows:

$Q'_\Delta(X)$:- $V_1(X)$.
$Q''_\Delta(X)$:- $V_2(X)$.
$Q'''_\Delta(X)$:- $V_5(X)$.

In this example, using the approach in Gryz (1999) or using the BFIND_1 algorithm, there are up to four Δ-equivalent or Δ-contained queries of Q to be obtained as follows:

$Q_1(X)$:- $S_1(X, y)$, $S_2(X, y)$.
$Q_2(X)$:- $R_1(X, y)$, $S_2(X, y)$.
$Q_3(X)$:- $R_1(X, y)$, $R_2(X, y)$.
$Q_4(X)$:- $S_1(X, y)$, $R_2(X, y)$.

Then for each of the above queries, the inverse rule algorithm in Qian (1996) or the MiniCon algorithm is used. Altogether these algorithms will be applied four times.

Example 7

1. $V_1(X)$:- $R_1(X, y_1, z_1)$, $R_2(y_1, k_1, j_1)$, $R_3(k_1, m_1)$. // $\mathbf{B}_1$={X, Y}, $\mathbf{B}_2$={Y, K}, $\mathbf{B}_3$={K}
   $V_2(X)$:- $R_1(X, y_2, z_2)$, $R_2(y_2, k_2, j_2)$. // $\mathbf{B}_1$={X, Y}, $\mathbf{B}_2$={Y, K}
2. $Q(X)$:- $S_1(X, y, z)$, $S_2(y, k, j)$, $S_3(k, m)$. // $\mathbf{A}_1$={X, Y}, $\mathbf{A}_2$={Y, K}, $\mathbf{A}_3$={K}
3. IND: $\{R_1[X, Y] \subseteq S_1[X, Y], R_2[Y, K] \subseteq S_2[Y, K], R_3[K] \subseteq S_3[K]\}$
4. We can generate one Δ-contained rewriting of Q using the BFIND_2 algorithm as follows:

   $Q'_\Delta(X)$:- $V1(X)$.

Example 8

1. $V_1(X,Y)$:- $R_1(X,Y,z_1)$.
   $V_2(X,Y)$:- $R_2(X,Y,z_2)$.
   $V_3(X)$:- $R_3(X,y_3)$.
   $V_4(X)$:- $R_4(X,y_4)$.
   $V_5(X)$:- $R_5(X,y_5)$.
   $V_6(X)$:- $R_6(X,y_6)$.
   $V_7(X)$:- $R_7(X,y_7)$.
   $V_8(X)$:- $R_8(X,y_8)$.
   $V_9(X)$:- $R_9(X,y_9)$.
   $V_{10}(X)$:- $R_{10}(X,y_{10})$.
2. $Q(X)$:- $S(X, y, z)$.
3. IND: $\{R_1[X, Y] \subseteq S[X, Y], R_2[X, Y] \subseteq S[X, Y]\}$
4. Using the BFIND_2 algorithm, we apply a chase_Δ procedure/rule to V_1 and V_2 only, and generate two Δ-contained rewritings of Q shown as follows:

   $Q'_\Delta(X)$:- $V_1(X, Y)$.
   $Q''_\Delta(X)$:- $V_2(X, Y)$.

   We do not need to apply a chase_Δ procedure/rule to other views. However, using the approach in Gryz (1999) or the BFIND_1 algorithm, we first get three Δ-equivalent or Δ-contained queries of Q as follows:

   $Q_1(X)$:- $S(X, y, z)$.
   $Q_2(X)$:- $R_1(X, Y, z)$.

$Q_3(X)$:- $R_2(X, Y, z)$.

For Q_i, i=1,2,3, we have to consider its rewriting problem over the ten views.

7. CONCLUSIONS

We have extended the MiniCon algorithm to exploit inclusion dependencies in order to generate all the rewritings for a query. A chase$_\Delta$ procedure/rule is introduced in Gryz (1999). The algorithm in Gryz (1999) is to first get a set of Δ-equivalent or Δ-contained queries of the original query, and then to apply an inverse rule algorithm to get rewritings for each of the Δ-equivalent or Δ-contained queries over a set of views. In this paper, we presented two algorithms to deal with the problem of query rewriting using views in the presence of inclusion dependencies. The first algorithm is similar to the one in Gryz (1999). That is, the query is changed while views remain unchanged. After a set of Δ-equivalent or Δ-contained queries of the original query are obtained, the MiniCon algorithm is used. In the second algorithm, we applied a chase$_\Delta$ procedure/rule to the views that contain chase$_\Delta$ reachable subgoals, instead of the given query. In other words, the query is not changed while some views are applied by a chase$_\Delta$ procedure/rule. We gave two conditions to ensure the chased views were available for forming the MCDs over those revised views. Theoretic analysis and experimental results have shown that the second algorithm is more efficient than the first algorithm. We proved that the union of all the obtained rewritings is a maximally-contained rewriting with respect to inclusion dependencies.

REFERENCES

ABITEBOUL, S., HULL, R. and VIANU, V. (1995): Foundations of databases, Addison-Wesley Publishing Company.

DUSCHKA, O.M. and GENESERETH, M.R. (1997): Answering recursive queries using views. *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Tucson, AZ, USA, 109–116, ACM Press.

DUSCHKA, O.M., GENESERETH, M.R. and LEVY, A.Y. (2000): Recursive query plans for data integration. *Journal of Logic Programming*, special issue on Logic Based Heterogeneous Information Systems 43(1):49–73.

GRANT, J. and MINKER, J. (2002): A logic-based approach to data integration. TLP 2(3):323–368.

GRYZ, J. (1999): Query rewriting using views in the presence of inclusion and inclusion dependencies. *Information System* 24(7):597–612.

JOHNSON, D.S. and KLUG, A. (1984): Testing containment of conjunctive queries under functional and inclusion dependencies. *Journal of Computer and System Sciences* 28:167–189.

LEVY, A. Y. (2001): Answering queries using views: A survey. *The VLDB Journal* 10(4):270–294.

LEVY, A.Y., RAJARAMAN, A. and ORDILLE, J.J. (1996a): Querying heterogeneous information sources using source descriptions. *Proceedings of the 22nd International Conference on Very Large Data Bases*, Mumbai, India, 251–262, Morgan Kaufmann.

LEVY, A.Y., RAJARAMAN, A. and ORDILLE, J.J. (1996b): Query-answering algorithms for information agents. *Proceedings of the 13th National Conference on Artificial Intelligence* and the *8th Innovative Applications of Artificial Intelligence Conference*, Portland, USA, 1:40-47, AAAI/MIT Press.

MITRA, P. (2001): An algorithm for answering queries efficiently using views. *Proceedings of the 12th Australasian Database Conference*, Queensland, Australia, 99–106, ACM Press.

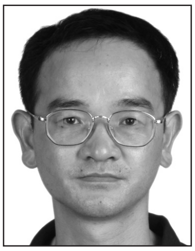
POTTINGER, R. and LEVY, A.Y. (2000): A scalable algorithm for answering queries using views. *Proceedings of the 26th International Conference on Very Large Data Bases*, Cairo, Egypt, 484–495, Morgan Kaufmann.

QIAN, X. (1996): Query folding. *Proceedings of the 12th IEEE International Conference on Data Engineering*, New Orleans, USA, 48–55, IEEE Computer Society Press.

ULLMAN, J. D. (1997): Information integration using logical views. *Proceedings of the 6th International Conference on Database Theory*, Delphi, Greece, 19–40, Springer-Verlag Press.

WIEDERHOLD, G. (1992): Mediators in the architecture of future information systems. *IEEE Computer*, 25(3): 38–49.

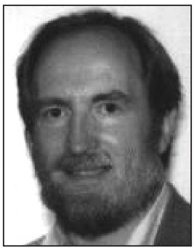
BIOGRAPHICAL NOTES


Qingyuan Bai

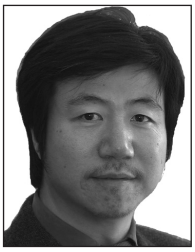
Dr Qingyuan Bai is an associate professor of Computer Science at Fuzhou University in China. He received his BSc and MSc degrees both in Computational Mathematics from Fuzhou University in 1985 and 1988 respectively, and PhD in Computer Science from University of Ulster in UK in 2005. His research interests are in the areas of Data Integration, Web Mining, Semantic Web, and so on. He has published over 20 papers in these areas in journals and international conferences. He has so far been involved in a number of research projects funded by the National Science Foundation of China and other sources.

Jun Hong

Dr Jun Hong is a senior lecturer of Computer Science at Queen's University Belfast in the UK. He received his BSc and MSc degrees both in Computer Science and PhD in Artificial Intelligence. His research interests are in the areas of AI planning, Web mining, semantic integration, data integration systems and reasoning under uncertainty. He has published over 50 papers in these areas, including a number of papers published in some of the most prestigious journals and international conferences in Artificial Intelligence. He has so far been involved in a number of research projects funded by the EU and other sources. He has been on the Program Committees of a number of international conferences. He is the Program Co-chair of BNCOD 2006 – the 23rd British National Conference on Databases.


Michael McTear

Michael McTear is Professor of Knowledge Engineering with a special research interest in spoken language technologies. He graduated in German Language and Literature from Queens University Belfast in 1965, was awarded an MA in Linguistics at the University of Essex in 1975, and a PhD at the University of Ulster in 1981. He has been Visiting Professor at the University of Hawaii and at the University of Koblenz, Germany. He has published several books and journal papers in the areas of spoken dialogue systems, natural language processing, user modelling, and language acquisition. His most recent book 'Spoken Dialogue Technology' is published by Springer Verlag (2004).


Hui Wang

Dr Hui Wang received his BSc(Computer Science) and MSc (Artificial Intelligence) from Jilin University of China in 1985 and 1988 respectively, and DPhil(Informatics) from the University of Ulster in 1996. He worked at the University of Ulster as a lecturer from 1996 to 2002, and a senior lecturer from 2002. His research interests include machine learning, data/text mining, uncertainty reasoning, spatial reasoning, and information retrieval. He has authored or co-authored over 70 scientific articles in these areas in journals and conferences, and edited a special issue of Decision Support Systems Journal.