Query Folding*

Xiaolei Qian
Computer Science Laboratory
SRI International
qian@csl.sri.com

Abstract

Query folding refers to the activity of determining if and how a query can be answered using a given set of resources, which might be materialized views, cached results of previous queries, or queries answerable by another database. We investigate query folding in the context where queries and resources are conjunctive queries. We develop an exponential-time algorithm that finds all foldings, and a polynomial-time algorithm for the subclass of acyclic queries. Our results can be applied to query optimization in centralized databases, to query processing in distributed databases, and to query answering in federated databases.

1 Introduction

Query folding refers to the activity of determining if and how a query can be answered using a given set of resources. These resources might be materialized views, cached results of previous queries, or even queries answerable by another database. Query folding is important because the base relations referred to in a query might be stored remotely and hence too expensive to access, or might not be available for access because of temporary network disconnection, or might be conceptual relations only and hence not existent physically. Query folding has applications in query optimization in centralized databases [4], query processing in distributed databases [6], and query answering in federated databases [7].

Example 1 Let us consider a patient record database consisting of the following relations.

Patients	(patient_id, clinic, dob, insurance)
Physician	(physician_id, clinic, pager_no)
Drugs	(drug_name, generic?)
Notes	(note_id, patient_id, physician_id, note_text)
Allergy	(note_id, drug_name, allergy_text)
Prescription	(note_id, drug_name, prescription_text).

*This work was supported in part by U.S. Department of Defense Advanced Research Projects Agency and U.S. Air Force Rome Laboratory under contracts F30602-92-C-0140 and F30602-94-C-0198, and in part by the National Science Foundation under grant ECS-94-22688.

Suppose that the database maintains a materialized view defined by

```
CREATE VIEW Drug_Allergy (patient_id, drug_name, text)
SELECT      patient_id, drug_name, allergy_text
FROM        Notes, Allergy
WHERE       Notes.note_id = Allergy.note_id.
```

A user might issue the following query to get the ids of patients at the Palo Alto Clinic who are allergic to an experimental drug `xd_2001`.

```
SELECT patient_id, allergy_text
FROM   Patients, Notes, Allergy
WHERE  Patients.patient_id = Notes.patient_id
       AND Notes.note_id = Allergy.note_id
       AND clinic = palo_alto
       AND drug_name = xd_2001.
```

Using the view, the above query can be folded to

```
SELECT patient_id, text
FROM   Patients, Drug_Allergy
WHERE  Patients.patient_id = Drug_Allergy.patient_id
       AND clinic = palo_alto
       AND drug_name = xd_2001.
```

This new query could be more efficient to evaluate than the original query. □

Query containment is a special case of query folding. To determine whether a query is contained in another query, we could determine instead whether the second query can be answered using a view defined by the first query. The problem of containment for conjunctive queries is known to be NP-complete [3]. Several subclasses of conjunctive queries have been identified that have polynomial-time containment algorithms [1, 2, 5].

Thus, the query-folding problem is at least NP-hard. Actually, it is shown recently to be NP-complete for conjunctive queries and resources [8], and for conjunctive queries and resources with binding patterns [10].

Solutions to query folding for conjunctive queries and resources with built-in predicates have been developed in [4, 12, 14]. The algorithms employed all use exhaustive search strategies that are exponential-time in complexity, and sometimes use unnecessary pruning conditions that do not guarantee to find all foldings. In addition, these algorithms compute only strong foldings, which are foldings that are equivalent to the original query. The algorithms in [12, 14] compute only complete foldings, which are foldings that depend solely on the resources. When strong and complete foldings do not exist, partial foldings that are contained in the original query or depend partially on the resources could be very useful in practice, especially in the distributed environment.

We consider the query-folding problem for conjunctive queries and resources. Section 2 provides some preliminary definitions. We show in Section 3 how to derive folding rules from resources, and

how to compute strong foldings from partial foldings. In Section 4 an exponential-time algorithm is developed that finds all complete or partial foldings. In Section 5, we give a polynomial-time algorithm for a large and natural subclass of conjunctive queries. It is similar in spirit to arc consistency algorithms for constraint satisfaction problems [9]. For the query containment problem, our algorithm degenerates to a containment algorithm for a new subclass of conjunctive queries that is incomparable to but more natural than the subclasses identified in [1, 2, 5]. Finally, we briefly discuss potential applications of query folding, and conclude the paper with Section 6.

2 Conjunctive Queries

2.1 Preliminaries

We consider queries and resources that are expressible as conjunctive queries or project-select-join queries where the selection conditions are restricted to equality. We assume that predicates are divided into three disjoint groups: base predicates, resource predicates, and query predicates. A *conjunctive query* Q has the form

$$h \quad :- \quad p_1, \ldots, p_n$$

where $h, p_1, \ldots, p_n$ are atomic formulas whose arguments are variables or constants, h is the ***head***, and $p_1, \ldots, p_n$ is the ***body***. Variables in the head are ***distinguished***. We assume that every distinguished variable also appears in the body. As a convention, we use $X, Y, \ldots$ for distinguished variables, $W, U, \ldots$ for other variables, and $A, B, \ldots$ for constants. We also use head(Q) and body(Q) to denote the head and body of Q, respectively, and var(h) to denote the list of variables in h. The SQL query in Example 1 is a conjunctive query

$$\begin{aligned} q(X, Y) \quad :- \quad & \text{patients}(X, \texttt{palo\_alto}, W_1, W_2), \\ & \text{notes}(W_3, X, W_4, W_5), \text{allergy}(W_3, \texttt{xd\_2001}, Y). \end{aligned}$$

The value of a conjunctive query in a database is the value of its head obtained by evaluating its body in that database. A conjunctive query Q is contained in another conjunctive query Q', denoted as $Q \subseteq Q'$, if the value of Q is a subset of the value of Q' in all possible databases. Q and Q' are equivalent, denoted as $Q \equiv Q'$, if $Q \subseteq Q'$ and $Q' \subseteq Q$.

2.2 Hypergraph Representation

A conjunctive query can be represented by a hypergraph as follows. Every variable in the body is represented by a node, and a node is ***distinguished*** if it represents a distinguished variable. A hyperedge is a set of nodes. Every conjunct in the body is represented by a hyperedge that contains the set of variables in the conjunct, in which case we say that the conjunct is associated with the hyperedge.

Example 2 Consider the following conjunctive query

$$\begin{aligned} q(X, Y) \quad :- \quad & \text{notes}(W_1, X, W_2, W_3), \text{allergy}(W_1, Y, W_4), \\ & \text{notes}(W_5, X, W_6, W_7), \text{prescription}(W_5, Y, W_8) \end{aligned}$$

which computes patients X and drugs Y such that X is prescribed to Y and is treated with allergy to Y. Its hypergraph is shown in Figure 1. □

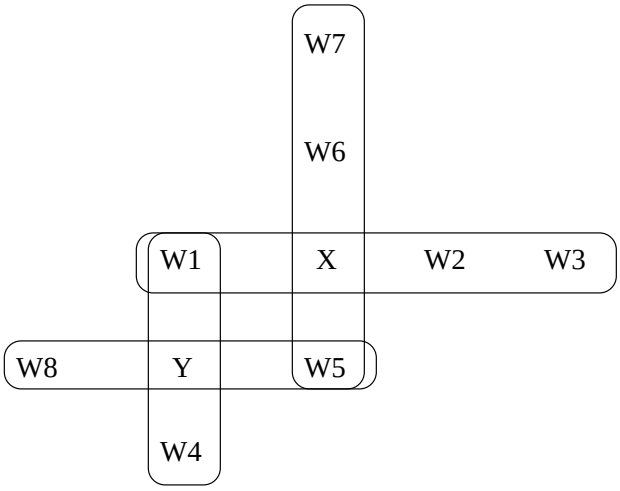

Figure 1: Hypergraph for Example 2

A conjunctive query is *acyclic* if its hypergraph representation is acyclic. We briefly recall the definitions of GYO-reductions of hypergraphs and acyclic hypergraphs from [13, Section 11.12]. Given two hyperedges p and q, if nodes in the set difference $p - q$ appear in no other hyperedges, then p is an *ear*. The *GYO-reduction* of a hypergraph is obtained by removing ears repeatedly until no more ears exist. A hypergraph is *acyclic* if its GYO-reduction is empty (i.e., contains no hyperedges). For example, the hypergraph in Figure 1 is cyclic.

3 The Query-Folding Problem

3.1 Folding Rules

A *resource* is a conjunctive query whose head contains a resource predicate and whose body contains base predicates only. Similarly, a *query* is a conjunctive query whose head contains a query predicate and whose body contains base predicates only. Let Q be a query, and $\mathcal{R} = \{R_1, \ldots, R_n\}$ be a set of resources. We assume that no two resources have the same resource predicate, and there are no variables in common between Q and R_i for $1 \leq i \leq n$, or between R_i and R_j for $1 \leq i, j \leq n$ and $i \neq j$. Hence, every resource (query) completely defines its resource (query) predicate.

A *partial folding* of Q using $\mathcal{R}$ is a conjunctive query Q' such that $Q' \subseteq Q$ and the body of Q' contains one or more resource predicates defined in $\mathcal{R}$. A partial folding of Q using $\mathcal{R}$ is *maximal* if there are no other partial foldings of Q using $\mathcal{R}$ that contain fewer base predicates. A *complete folding* of Q using $\mathcal{R}$ is a partial folding of Q using $\mathcal{R}$ whose body contains resource predicates only. A *strong folding* of Q using $\mathcal{R}$ is a partial folding Q' of Q using $\mathcal{R}$ such that $Q \subseteq Q'$.[1]

[1]Using results from [11], our definitions can be generalized directly to foldings that are unions of conjunctive

Example 3 Consider the following resources.

$$\begin{aligned} r_1(X_1, X_2, X_3) &:- \text{notes}(U_1, X_1, U_2, U_3), \text{allergy}(U_1, X_2, X_3) \\ r_2(Y_1, Y_2, Y_3, Y_4) &:- \text{notes}(V_1, Y_1, Y_2, V_2), \text{prescription}(V_1, Y_3, V_3), \text{drugs}(Y_3, Y_4). \end{aligned}$$

A complete folding of the query in Example 2 using these resources is

$$q(X, Y) :- r_1(X, Y, W), r_2(X, W_1, Y, W_2).$$

□

Consider a resource of the form

$$r :- p_1, \ldots, p_n.$$

Let $X_1, \ldots, X_l$ be the distinguished variables, and $Y_1, \ldots, Y_m$ be the rest of variables. Since the resource predicate is completely defined by the body of the resource, the resource implies the formula

$$(\forall X_1, \ldots, X_l)(r \rightarrow (\exists Y_1, \ldots, Y_m)(p_1 \wedge \cdots \wedge p_n)).$$

After skolemization, we derive n ***folding rules*** below.

$$\begin{aligned} p_1[f_1(X_1, \ldots, X_l)/Y_1, \ldots, f_m(X_1, \ldots, X_l)/Y_m] &:- r \\ &\vdots \\ p_n[f_1(X_1, \ldots, X_l)/Y_1, \ldots, f_m(X_1, \ldots, X_l)/Y_m] &:- r \end{aligned}$$

where $f_1, \ldots, f_m$ are skolem functions with arguments $X_1, \ldots, X_l$. Notice that, for a folding rule, the head is a single conjunct containing a base predicate, the body is a single conjunct containing a resource predicate, and every variable in the head also appears in the body. For example, from the resources in Example 3, we derive the following folding rules.

$$\begin{aligned} \text{notes}(f_1(X_1, X_2, X_3), X_1, f_2(X_1, X_2, X_3), f_3(X_1, X_2, X_3)) &:- r_1(X_1, X_2, X_3) \\ \text{allergy}(f_1(X_1, X_2, X_3), X_2, X_3) &:- r_1(X_1, X_2, X_3) \\ \text{notes}(g_1(Y_1, Y_2, Y_3, Y_4), Y_1, Y_2, g_2(Y_1, Y_2, Y_3, Y_4)) &:- r_2(Y_1, Y_2, Y_3, Y_4) \\ \text{prescription}(g_1(Y_1, Y_2, Y_3, Y_4), Y_3, g_3(Y_1, Y_2, Y_3, Y_4)) &:- r_2(Y_1, Y_2, Y_3, Y_4) \\ \text{drugs}(Y_3, Y_4) &:- r_2(Y_1, Y_2, Y_3, Y_4). \end{aligned}$$

We can reformulate the query-folding problem as follows. Given a query Q, a set of resources, and a set of folding rules $\mathcal{F}$ derived from the resources, a partial folding of Q using $\mathcal{F}$ is a conjunctive query, obtained by rewriting Q using $\mathcal{F}$, that contains one or more resource predicates but no skolem terms. For example, readers can verify that rewriting the query in Example 2 using the folding rules above leads to the query in Example 3 (modulo variable renaming).

queries. In other words, a union of conjunctive queries is a partial folding of Q iff every conjunctive query in the union is a partial folding of Q. A union of conjunctive queries is a strong folding of Q iff every conjunctive query in the union is a partial folding of Q, and at least one conjunctive query in the union is a strong folding of Q.

3.2 Strong Foldings

A strong folding of a query is a partial folding that contains the original query. To show that one query is contained in another query, we can show that the second query can be folded using the first query as the resource.

Lemma 1 *Let Q and Q_r be two conjunctive queries. $Q_r \subseteq Q$ iff Q has a folding Q_f using Q_r as the resource such that body(Q_f) = head(Q_r).*

Proof Let $\mathcal{F}$ be the set of folding rules derived from Q_r. First suppose that Q_f is a folding of Q using Q_r as the resource, and $\text{body}(Q_f) = \text{head}(Q_r)$. Obviously, $Q_r \subseteq Q_f$. Since Q_f is a rewriting of Q using $\mathcal{F}$, $Q_f \subseteq Q$. Hence, $Q_r \subseteq Q$.

Second suppose that $Q_r \subseteq Q$. Without loss of generality, assume that Q_r and Q have the same distinguished variables. Let body(Q) be of the form $p_1, \ldots, p_n$. Using results from [3], there is a substitution σ such that Q_r has the form

$$\text{head}(Q_r) \quad :- \quad p_1\sigma, \ldots, p_n\sigma, p'_1, \ldots, p'_m.$$

In deriving $\mathcal{F}$ from Q_r, let δ be the substitution that replaces the non-distinguished variables in body(Q_r) by skolem terms. In $\mathcal{F}$, we have folding rules of the form

$$\begin{aligned} p_1\sigma\delta \quad &:- \quad \text{head}(Q_r) \\ &\vdots \\ p_n\sigma\delta \quad &:- \quad \text{head}(Q_r). \end{aligned}$$

Hence, we can rewrite Q using these folding rules, leading to a conjunctive query Q_f where $\text{body}(Q_f) = \text{head}(Q_r)$. Since δ does not substitute distinguished variables in Q_r (and hence Q) by skolem terms, Q_f does not contain skolem terms. Thus, Q_f is a folding of Q using Q_r as the resource. □

Let Q_p be a partial folding of query Q using a set of resources $\mathcal{R}$. Also let Q_r be the query obtained from Q_p by replacing every resource predicate by its definition in $\mathcal{R}$. Since $Q_p \equiv Q_r$, Q_p is a strong folding of Q iff $Q \subseteq Q_r$, and according to Lemma 1, iff Q_r has a folding Q_f using Q as the resource such that $\text{body}(Q_f) = \text{head}(Q)$.

For example, if we expand the resource predicates in the query in Example 3 by their definitions, and take the conjunctive query in Example 2 as the resource, then the expanded query does not have complete foldings. Hence, the query in Example 3 is not a strong folding of the query in Example 2.

Lemma 1 gives a straightforward way of determine whether a partial folding is a strong folding. Therefore, we do not explicitly consider strong foldings any more in the rest of the paper.

4 A Query Folding Algorithm

Let Q be a query, G_Q be the hypergraph representing Q, and $\mathcal{F}$ be a set of folding rules. Our query-folding algorithm computes complete or partial foldings of Q using $\mathcal{F}$. It consists of two steps.

Initialization The first step is to compute a label for every hyperedge in G_Q. Given hyperedge $e \in G_Q$ and conjunct p associated with e, its label L_e is a relation with attributes $\mathrm{var}(p)$. We denote the list of attributes of L_e by $\mathrm{attr}(L_e)$. For every $F \in \mathcal{F}$ such that p unifies with $\mathrm{head}(F)$ with a most general unifier (mgu) σ, there is a tuple in L_e consisting of two parts: tuple $\mathrm{var}(p)\sigma$ and expression $\mathrm{body}(F)\sigma$, where the second part is used to store a folding of p. We denote such a tuple by $\mathrm{var}(p)\sigma|\mathrm{body}(F)\sigma$. The value of L_e is the set

$$\left\{ \begin{array}{ll} \mathrm{var}(p)\sigma|\mathrm{body}(F)\sigma\colon & F \in \mathcal{F} \\ & \wedge \sigma = \mathrm{mgu}(\mathrm{head}(F), p) \\ & \wedge \mathrm{body}(F)\sigma \text{ contains no skolem terms} \end{array} \right\}.$$

If we are seeking partial foldings of Q using $\mathcal{F}$, then we add a ***default*** tuple $\mathrm{var}(p)|p$ to L_e.

Folding Generation Given two hyperedges e and e' with labels L_e and L'_e, respectively, let $\mathcal{Y} = \mathrm{attr}(L_e) \cap \mathrm{attr}(L'_e)$ and $\mathcal{Z} = \mathrm{attr}(L'_e) - \mathrm{attr}(L_e)$. The ***u-join*** of L_e and L'_e, denoted as $L_e \overset{u}{\bowtie} L'_e$, is a relation with attributes $\mathrm{attr}(L_e), \mathcal{Z}$ defined by the set

$$\left\{ \begin{array}{ll} (t, t'[\mathcal{Z}])\sigma|(u, u')\sigma\colon & t|u \in L_e \\ & \wedge t'|u' \in L'_e \\ & \wedge \sigma = \mathrm{mgu}(t[\mathcal{Y}], t'[\mathcal{Y}]) \\ & \wedge (u, u')\sigma \text{ contains no skolem terms} \end{array} \right\}.$$

The second step is to actually construct the set of foldings. It proceeds by u-joining the labels of all hyperedges in an arbitrary order. The set of foldings is found in the second parts of tuples in the result:

$$\{u\colon t | u \in \mathrm{folding}(G_Q) \wedge u \text{ contains a resource predicate}\}$$

where $\mathrm{folding}(G_Q)$ is defined in Figure 2.

Algorithm. Folding
INPUT. A hypergraph
OUTPUT. A relation
procedure folding(G): relation
  let e be a hyperedge in G with label L_e
  $G' \leftarrow G - e$;
  if G' is empty **then return** L_e **else return** $L_e \overset{u}{\bowtie} \mathrm{folding}(G')$.

Figure 2: Folding

Let us apply the algorithm to Example 3. Assuming that we are seeking complete foldings, the first step computes the labels associated with the hyperedges in Figure 1, as shown in Figure 3, and the second step computes the u-join of these labels, which contains one tuple whose second component is $r_1(X_1, X_2, X_3), r_2(X_1, Y_2, X_2, Y_4)$. Hence, the query in Example 2 has one complete folding, which is the same as the folding in Example 3 (modulo variable renaming).

W_1	X	W_2	W_3	
$f_1(X_1, X_2, X_3)$	X_1	$f_2(X_1, X_2, X_3)$	$f_3(X_1, X_2, X_3)$	$r_1(X_1, X_2, X_3)$
$g_1(Y_1, Y_2, Y_3, Y_4)$	Y_1	Y_2	$g_2(Y_1, Y_2, Y_3, Y_4)$	$r_2(Y_1, Y_2, Y_3, Y_4)$

W_1	Y	W_4	
$f_1(X_1, X_2, X_3)$	X_2	X_3	$r_1(X_1, X_2, X_3)$

W_5	X	W_6	W_7	
$f_1(X_1, X_2, X_3)$	X_1	$f_2(X_1, X_2, X_3)$	$f_3(X_1, X_2, X_3)$	$r_1(X_1, X_2, X_3)$
$g_1(Y_1, Y_2, Y_3, Y_4)$	Y_1	Y_2	$g_2(Y_1, Y_2, Y_3, Y_4)$	$r_2(Y_1, Y_2, Y_3, Y_4)$

W_5	Y	W_8	
$g_1(Y_1, Y_2, Y_3, Y_4)$	Y_3	$g_3(Y_1, Y_2, Y_3, Y_4)$	$r_2(Y_1, Y_2, Y_3, Y_4)$

Figure 3: Example 3

Theorem 2 *A conjunctive query Q', where body(Q') contains one or more resource predicates, is a partial folding of Q using $\mathcal{F}$ iff there is some t such that $t|$body$(Q') \in$ folding(G_Q) (modulo variable renaming).*

Proof First, suppose that $t|\text{body}(Q') \in \text{folding}(G_Q)$ for some t. We prove by induction on the length of body(Q) that Q' is a partial folding of Q.

1. The base case is when body(Q) is an atomic formula p. G_Q has one hyperedge e with label L_e, and $t|\text{body}(Q') \in L_e$. Since body($Q'$) contains a resource predicate, there is a folding rule $F \in \mathcal{F}$ and a mgu σ such that p unifies with head(F) and body(Q') is body(F)σ. Hence, Q' is obtained by rewriting body(Q) using F. Since body(F)σ does not contain skolem terms, Q' does not contain skolem terms.

2. Suppose that body(Q) has the form $p_1, p_2, \ldots, p_n$. Let $e \in G_Q$ be the hyperedge with which conjunct p_1 is associated, L_e be the label of e, and $G = G_Q - e$. Also let $\mathcal{Z} = \text{var}(p_2, \ldots, p_n) - \text{var}(p_1)$. There are $t_1|u_1 \in L_e$ and $t_2|u_2 \in \text{folding}(G)$ such that $t = t_1, t_2[\mathcal{Z}]$ and body(Q') $= u_1, u_2$. Either $u_1 = p_1$ or u_1 is obtained by rewriting p_1 using some $F \in \mathcal{F}$. In the latter case, u_1 does not contain skolem terms. By induction hypothesis, u_2 is obtained by rewriting $p_2, \ldots, p_n$ using $\mathcal{F}$, and u_2 does not contain skolem terms. Thus, Q' is obtained by rewriting body(Q) using $\mathcal{F}$, and Q' does not contain skolem terms.

Second, suppose that Q' is a partial folding of Q obtained by rewriting body(Q) using $\mathcal{F}$. Without loss of generality, assume that body(Q) has the form $p_1, \ldots, p_m, \ldots, p_n$ for some $1 \leq m \leq n$, where rewrite rules are applied to $p_1, \ldots, p_m$ only. Since every rewriting step replaces exactly one conjunct, there are m (not necessarily different) folding rules $F_1, \ldots, F_m \in \mathcal{F}$ used in the rewriting, such that p_i unifies with head(F_i) with mgu σ_i, and var(p_i)$\sigma_i|$body(F_i)σ_i is in the label of the

hyperedge in G_Q with which p_i is associated, for $1 \leq i \leq m$. Recall that $\text{var}(p_i)|p_i$ is a default tuple in the label of the hyperedge in G_Q with which p_i is associated, for $m+1 \leq i \leq n$. Hence, $\text{body}(Q')$ is

$$(\text{body}(F_1), \ldots, \text{body}(F_m), p_{m+1}, \ldots, p_n)\sigma_1 \ldots \sigma_m$$

and $\text{var}(\text{body}(Q))\sigma_1 \ldots \sigma_m | \text{body}(Q')$ is in $\text{folding}(G_Q)$. □

Now let us consider the cost of computing foldings. Since unification is linear time, labels in G_Q can be computed in time polynomial in the sizes of Q and $\mathcal{F}$. The size of a label is bounded by the size of $\mathcal{F}$, and there are as many hyperedges as there are conjuncts in $\text{body}(Q)$. Hence, the cost of computing $\text{folding}(G_Q)$ is exponential in the size of $\mathcal{F}$.

5 Query Folding for Acyclic Queries

5.1 Existence of Folding

Let Q be an acyclic query, G_Q be the hypergraph representing Q, $\mathcal{R}$ be a set of resources (notice that resources can be cyclic), and $\mathcal{F}$ be a set of folding rules derived from $\mathcal{R}$. We compute the labels in G_Q using $\mathcal{F}$ in the same way as in Section 4.

Given two hyperedges e and e' with labels L_e and L'_e, respectively, let $\mathcal{Y} = \text{attr}(L_e) \cap \text{attr}(L'_e)$. The ***u-semijoin*** of L_e and L'_e, denoted as $L_e \overset{u}{\ltimes} L'_e$, is defined by the set

$$\left\{ \begin{array}{ll} t\sigma | u\sigma: & t|u \in L_e \\ & \wedge t'|u' \in L'_e \\ & \wedge \sigma = \text{mgu}(t[\mathcal{Y}], t'[\mathcal{Y}]) \\ & \wedge u\sigma \text{ contains no skolem terms.} \end{array} \right\}.$$

A hypergraph is ***pairwise-consistent*** if, for every pair of hyperedges e, e' and their labels L_e, L'_e where $e \cap e' \neq \emptyset$, we have that $L_e \overset{u}{\ltimes} L'_e = L_e$. Figure 4 gives an algorithm for reducing an acyclic hypergraph to a pairwise-consistent acyclic hypergraph.

Example 4 Pairwise consistency is necessary but not sufficient for the existence of foldings of cyclic queries. Consider the query

$$q(X, Y) \quad :- \quad \text{patients}(W_1, W_2, W_3, W_4), \text{notes}(X, W_1, W_5, Y), \text{physician}(W_5, W_2, W_6)$$

which computes the notes for those patients who are seen by physicians at the same clinic. If we have the following two resources

$$\begin{array}{rcl} r_1(X_1, X_2) & :- & \text{patients}(B_1, A_1, U_1, U_2), \text{notes}(X_1, B_1, C_1, X_2), \text{physician}(C_1, A_2, U_3) \\ r_2(Y_1, Y_2) & :- & \text{patients}(B_2, A_2, V_1, V_2), \text{notes}(Y_1, B_2, C_2, Y_2), \text{physician}(C_2, A_1, V_3) \end{array}$$

where A_i, B_i, C_i for $1 \leq i \leq 2$ are distinct constants, then the query does not have complete foldings using these resources. However, if we use the folding rules derived from these resources to compute

Algorithm. Reduction
INPUT. An acyclic hypergraph
OUTPUT. A pairwise-consistent acyclic hypergraph
procedure reduction(G): hypergraph
  if G is empty **then return** G
  else let e be an ear in G with label L_e
    $G' \leftarrow G - e$;
    foreach $e' \in G'$ with label L'_e **where** $e \cap e' \neq \emptyset$ **do** $L'_e \leftarrow L'_e \overset{u}{\ltimes} L_e$;
    $G' \leftarrow$ reduction(G');
    foreach $e' \in G'$ with label L'_e **where** $e \cap e' \neq \emptyset$ **do** $L_e \leftarrow L_e \overset{u}{\ltimes} L'_e$;
    return $G' + e$.

Figure 4: Reduction

the labels of the hypergraph of the query, and apply the algorithm of Figure 4 to it, we would get a non-empty and pairwise-consistent hypergraph. □

Theorem 3 *There exists a complete folding of acyclic query Q using folding rules $\mathcal{F}$ iff no hyperedges in reduction(G_Q) have empty labels.*

Proof Suppose that Q' is a complete folding of Q using $\mathcal{F}$. According to Theorem 2, there is some t such that $t|\text{body}(Q') \in \text{folding}(G_Q)$. Recall that folding$(G_Q)$ is a relation over attributes var(body(Q)). For every hyperedge e in G_Q and its associated conjunct p, $t[\text{var}(p)]$ is a tuple in the label of e in reduction(G_Q). Thus, no hyperedges in reduction(G_Q) have empty labels.

Suppose that no hyperedges in $G = \text{reduction}(G_Q)$ have empty labels. We show by induction on the number of hyperedges in G that folding(G) is not empty.

1. The base case when G contains one hyperedge is obvious.
2. Let $e \in G$ be an ear, L_e be the label of e, and $G' = G - e$. Let $\mathcal{Y} = \text{attr}(L_e) \cap \text{attr}(\text{folding}(G'))$. For every tuple $t \in L_e$, there is a tuple $t' \in \text{folding}(G')$ such that $t[\mathcal{Y}] = t'[\mathcal{Y}]$, and vice versa. Since L_e is not empty, and by induction hypothesis folding(G') is not empty, folding$(G) = L_e \overset{u}{\bowtie} \text{folding}(G')$ is not empty.

Let $t|u \in \text{folding}(G)$. Since folding$(G) \subseteq$ folding(G_Q), u is a folding of Q according to Theorem 2. For every hyperedge in G and every tuple $t'|u'$ in its label, u' is an expression containing resource predicates only. Hence, u contains resource predicates only and is a complete folding of Q. □

Example 5 Consider the acyclic query below.

$$q(X, Y) \text{ :- } \text{allergy}(X, W_1, W_2), \text{drugs}(W_1, W_3), \text{notes}(X, W_4, W_5, W_6), \text{patients}(W_4, Y, W_7, W_8)$$

which computes the notes from clinics that describe allergic reactions. Suppose that we have the following two resources

$$\begin{aligned} r_1(X_1, X_2) &:- \text{allergy}(X_1, U_1, U_2), \text{drugs}(U_1, X_2), \text{notes}(X_1, U_3, U_4, U_5) \\ r_2(Y_1, Y_2) &:- \text{notes}(Y_1, V_1, V_2, V_3), \text{patients}(V_1, Y_2, V_4, V_5), \text{drugs}(V_6, V_7) \end{aligned}$$

from which we derive the following folding rules

$$\begin{aligned} \text{allergy}(X_1, f_1(X_1, X_2), f_2(X_1, X_2)) &:- r_1(X_1, X_2) \\ \text{drugs}(f_1(X_1, X_2), X_2) &:- r_1(X_1, X_2) \\ \text{notes}(X_1, f_3(X_1, X_2), f_4(X_1, X_2), f_5(X_1, X_2)) &:- r_1(X_1, X_2) \\ \text{notes}(Y_1, g_1(Y_1, Y_2), g_2(Y_1, Y_2), g_3(Y_1, Y_2)) &:- r_2(Y_1, Y_2) \\ \text{patients}(g_1(Y_1, Y_2), Y_2, g_4(Y_1, Y_2), g_5(Y_1, Y_2)) &:- r_2(Y_1, Y_2) \\ \text{drugs}(g_6(Y_1, Y_2), g_7(Y_1, Y_2)) &:- r_2(Y_1, Y_2). \end{aligned}$$

To fold the query completely using these folding rules, Figure 5 gives the labels of its hypergraph, after reduction using the algorithm in Figure 4. According to Theorem 3, this query has a complete folding. □

X	W_1	W_2	
X_1	$f_1(X_1, X_2)$	$f_2(X_1, X_2)$	$r_1(X_1, X_2)$

W_1	W_3	
$f_1(X_1, X_2)$	X_2	$r_1(X_1, X_2)$

X	W_4	W_5	W_6	
X_1	$g_1(X_1, Y_2)$	$g_2(X_1, Y_2)$	$g_3(X_1, Y_2)$	$r_2(X_1, Y_2)$

W_4	Y	W_7	W_8	
$g_1(X_1, Y_2)$	Y_2	$g_4(X_1, Y_2)$	$g_5(X_1, Y_2)$	$r_2(X_1, Y_2)$

Figure 5: Example 5

Theorem 4 *There does not exist a partial folding of acyclic query Q using folding rules $\mathcal{F}$ iff every hyperedge in reduction(G_Q) has a singleton label.*

Proof For every conjunct p in body(Q), the label of the hyperedge that represents p contains at least one tuple var$(p)|p$. Consequently, folding(reduction(G_Q)) contains at least one tuple var(body(Q))|body(Q).

Suppose that every hyperedge in reduction(G_Q) has a singleton label. Hence, there is exactly one tuple var(body(Q))|body(Q) $\in$ folding(reduction(G_Q)), meaning that Q does not have a partial folding using $\mathcal{F}$.

Suppose that there is a hyperedge e in reduction(G_Q) whose label L_e contains more than one tuple. Let $t|u \in L_e$ such that u contains a resource predicate. Thus, there is $t'|u' \in$ folding(reduction(G_Q)) such that u' contains a resource predicate, meaning that u' is a partial folding of Q. □

Example 6 Consider the acyclic query below.

$$\begin{aligned} q(X,Y) \quad :- \quad & \text{patients}(X, W_1, W_2, \texttt{medicare}), \text{notes}(W_3, X, W_4, W_5), \\ & \text{prescription}(W_3, Y, W_6), \text{drugs}(Y, \texttt{no}) \end{aligned}$$

which computes the medicare patients who are prescribed to non-generic drugs. Suppose that we have the following resource

$$r(X_1, X_2, X_3) \quad :- \quad \text{notes}(U_1, X_1, X_2, U_2), \text{prescription}(U_1, X_3, U_3), \text{drugs}(X_3, U_4)$$

from which we derive the following folding rules

$$\begin{aligned} \text{notes}(f_1(X_1, X_2, X_3), X_1, X_2, f_2(X_1, X_2, X_3)) \quad &:- \quad r_1(X_1, X_2, X_3) \\ \text{prescription}(f_1(X_1, X_2, X_3), X_3, f_3(X_1, X_2, X_3)) \quad &:- \quad r_1(X_1, X_2, X_3) \\ \text{drugs}(X_3, f_4(X_1, X_2, X_3)) \quad &:- \quad r_1(X_1, X_2, X_3). \end{aligned}$$

To fold the query partially using these folding rules, Figure 6 gives the labels of its hypergraph, after reduction using the algorithm in Figure 4. According to Theorem 4, this query has a partial folding. □

Theorem 5 *The problem of whether there exists a folding of acyclic query Q using folding rules $\mathcal{F}$ is decidable in time polynomial in the sizes of Q and $\mathcal{F}$.*

Proof Let c be the number of conjuncts in Q, s_c be the size of the largest conjunct in Q, $f = |\mathcal{F}|$, and s_f be the size of the largest folding rule in $\mathcal{F}$. The size of Q is $O(c \times s_c)$, and the size of $\mathcal{F}$ is $O(f \times s_f)$.

For every hyperedge $e \in G_Q$ and its associated conjunct p, the cost of computing its label L_e is $O(f \times (s_f + s_c))$, since unification is linear time. Hence, the cost of computing labels in G_Q is $O(c \times f \times (s_f + s_c))$.

Let e be an ear in G_Q. The size of L_e is $O(f \times s_f)$. The cost of computing reduction(G_Q) is $O(c \times (s_c^2 + (f \times s_f)^2))$, plus the cost of computing reduction($G_Q - e$). Thus, the cost of computing reduction(G_Q) is $O(c^2 \times (s_c^2 + (f \times s_f)^2))$.

Finally, the cost of checking if every hyperedge in reduction(G_Q) has an empty or singleton label is $O(c)$. Therefore, the total cost of deciding if there exists a folding of Q using $\mathcal{F}$ is bounded by

$$O(f \times (c \times s_c) + (c \times s_c)^2 + (c \times f \times s_f)^2)$$

X	W_1	W_2	
X_1	W_1	W_2	patients(X_1, W_1, W_2,`medicare`)

W_3	X	W_4	W_5	
W_3	X_1	W_4	W_5	notes(W_3, X_1, W_4, W_5)
$f_1(X_1, X_2, X_3)$	X_1	X_2	$f_2(X_1, X_2, X_3)$	$r(X_1, X_2, X_3)$

W_3	Y	W_6	
W_3	X_3	W_6	Prescription(W_3, X_3, W_6)
$f_1(X_1, X_2, X_3)$	X_3	$f_3(X_1, X_2, X_3)$	$r(X_1, X_2, X_3)$

Y	
X_3	drugs(X_3,`no`)

Figure 6: Example 6

which is polynomial in the sizes of Q and $\mathcal{F}$. □

5.2 Compute Foldings

Given an acyclic query Q, its hypergraph G_Q, and a set of folding rules $\mathcal{F}$ (extended appropriately if partial foldings are sought), our algorithm computes complete or maximal partial foldings of Q using $\mathcal{F}$. It consists of three steps.

Initialization The first step is the same as in Section 4.

Hypergraph Reduction The second step is to reduce G_Q to a pairwise-consistent hypergraph, by using the algorithm of Figure 4. If the resulting G_Q is empty (if we are seeking complete foldings) or every hyperedge in G_Q has a singleton label (if we are seeking partial foldings), then there is no need to proceed to the next step: there does not exist a folding of Q. Otherwise, if we are seeking partial foldings, then we remove the default tuple from the label of every non-singleton hyperedge in G_Q.

Folding Generation The last step is similar to the second step of Section 4, except that u-join is replaced by ordinary join, and the order of join is the reverse of the GYO-reduction of G_Q. The set of foldings is found in the second parts of tuples in the join result: $\{u\colon t | u \in \text{folding}(\text{reduction}(G_Q))\}$.

For Example 5, joining the relations in Figure 5 we obtain a complete folding.

$$q(X_1, Y_2) \quad :- \quad r_1(X_1, X_2), r_2(X_1, Y_2).$$

Note that this folding cannot be obtained by pattern matching the query to the two resources sequentially in any order, since the second and the third conjuncts in the body of the query are needed in both matchings.

For Example 6, we first remove the default tuples from the second and third relations in Figure 6, and then join the relations, which gives us a maximal partial folding.

$$q(X_1, X_3) \quad :- \quad \text{patients}(X_1, W_1, W_2, \texttt{medicare}), r(X_1, X_2, X_3), \text{drugs}(X_3, \texttt{no}).$$

Note that the drugs conjunct of this query cannot be removed, even though it matches the drugs conjunct in the resource, because U_4 is not a distinguished variable in the resource.

Now let us consider the cost of computing foldings. From Theorem 5, the cost of the first two steps is polynomial in the sizes of Q and $\mathcal{F}$. Let U be the set of foldings of Q using $\mathcal{F}$. Since reduction(G_Q) is pairwise consistent, the size of every label in reduction(G_Q) is bounded by the size of U, and the size of every intermediate join result in the third step is also bounded by the size of U. Since the number of joins is bounded by the size of Q, the cost of the third step is polynomial in the sizes of U and Q. Hence, the total cost of computing foldings for acyclic queries is polynomial in the sizes of Q, $\mathcal{F}$, and U.

6 Conclusion

The query-folding problem is the problem of determining if and how a query can be answered using a given set of resources. We have investigated this problem in the context where queries and resources are conjunctive queries. In particular, we have developed a simple, exponential-time algorithm that finds all foldings, and a polynomial-time algorithm for the large and natural subclass of queries that are acyclic.

Our results show that query containment is a special case of query folding (Lemma 1). For the class of acyclic queries, our algorithm in Section 5.2 degenerates to a polynomial-time containment algorithm. This class is incomparable to and more natural than the classes of queries identified in [1, 2, 5] as having polynomial-time containment algorithms. For example, a polynomial-time algorithm is developed in [5] for the class of fan-out free queries. There are acyclic queries that are not fan-out free, for example,

$$q(X_1) \quad :- \quad r(Y_1, X_1), r(Y_1, Y_2), r(Y_3, X_1), r(Y_3, Y_4)$$

and there are cyclic queries that are fan-out free, for example,

$$q(X_1, X_2) \quad :- \quad r(Y_1, Y_2), r(Y_1, X_1), r(Y_2, X_1), r(Y_2, X_2).$$

Query folding has obvious applications in centralized databases. For example, databases often maintain materialized views, and a query can be answered by accessing views instead of base relations if the query can be folded using the views [4]. In multiple query answering, the result of a query can be used to at least partially answer another query if the second query can be folded using the first one. Query folding is even more important in a distributed environment. In a client-server application, views and queries might be cached at the client site. Client queries can be answered

more efficiently if they can be folded using the cached data [6]. In the situation of a disconnected network, a query can still be answered at least partially if it can be folded using views and queries maintained at available sites.

In a federated environment containing multiple heterogeneous, autonomous, and legacy data sources, a data source might be capable of answering only limited kinds of queries [7]. We can envision a data source being described using a set of statements of the form[2]

$$q_1, \ldots, q_m \quad :- \quad p_1, \ldots, p_n$$

where $q_1, \ldots, q_m$ is a conjunctive query in the federated schema, and $p_1, \ldots, p_n$ is a conjunctive query in a data source. This statement says that a federated query $q_1, \ldots, q_m$ can be answered by a source query $p_1, \ldots, p_n$. From this statement, we can define a view in the federated database

$$r \quad :- \quad q_1, \ldots, q_m.$$

The job of the federated query processor is to determine if and how federated queries can be answered using these views, or equivalently, to determine if and how federated queries can be folded using these views.

Acknowledgment

The author is indebted to Louiqa Raschid, Alon Levy, Yannis Papakonstantinou, V. S. Subrahmanian, Daniela Florescu, and Pierre Huyn for helpful discussions.

References

[1] A. V. Aho, Y. Sagiv, and J. D. Ullman. Efficient optimization of a class of relational expressions. *ACM Transactions on Database Systems*, 4(4):435–454, December 1979.

[2] A. V. Aho, Y. Sagiv, and J. D. Ullman. Equivalence of relational expressions. *SIAM Journal of Computing*, 8(2):218–246, 1979.

[3] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proceedings of the Ninth ACM Symposium on Theory of Computing*, pages 77–90, 1977.

[4] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 190–200, 1995.

[5] D. S. Johnson and A. Klug. Optimizing conjunctive queries that contain untyped variables. *SIAM Journal of Computing*, 12(4):616–640, November 1983.

[2] Notice that these statements are more general than the site descriptions in [7].

[6] A. M. Keller and J. Basu. A predicate-based caching scheme for client-server database architectures. In *Proceedings of the International Conference on Parallel and Distributed Information Systems*, 1994.

[7] A. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, 1995.

[8] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proceedings of the Fourteenth ACM Symposium on Principles of Database Systems*, pages 95–104, 1995.

[9] A. K. Mackworth. The logic of constraint satisfaction. *Artificial Intelligence*, 58(1–3):3–20, December 1992.

[10] A. Rajaraman, Y. Sagiv, and J. D. Ullman. Answering queries using templates with binding patterns. In *Proceedings of the Fourteenth ACM Symposium on Principles of Database Systems*, pages 105–112, 1995.

[11] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *Journal of the ACM*, 27(4):633–655, October 1980.

[12] O. G. Tsatalos, M. H. Solomon, and Y. E. Ioannidis. The GMAP: A versatile tool for physical data independence. In *Proceedings of the Twentieth International Conference on Very Large Data Bases*, pages 367–378. Morgan Kaufmann, 1994.

[13] J. D. Ullman. *Principles of Database and Knowledge Base Systems*, volume 2. Computer Science Press, 1989.

[14] H. Z. Yang and P.-Å. Larson. Query transformation for PSJ-queries. In *Proceedings of the Thirteenth International Conference on Very Large Data Bases*, pages 245–254. Morgan Kaufmann, 1987.