CONDITIONS FOR LOSSLESS JOIN

PRAKASH RAMANAN*

Department of Computer Science, Wichita State University, Wichita, KS 67260-0083

(Received 29 August 2000)

There is a well-known algorithm for determining when a decomposition \( \rho = \{ R_1, R_2, \ldots, R_m \} \) of a database relation scheme has a lossless join with respect to a given set of functional dependencies. We first present a reformulation of this algorithm in terms of set closures. For the special case of \( m = 2 \), there is a well-known explicit condition for losslessness. Our formulation extends this result for general \( m \). Also, our formulation leads to a strong necessary condition for \( \rho \) to be lossless. Separately, we prove a sufficient condition for \( \rho \) to be lossless. Finally, we present a sufficient condition, and a necessary condition for \( \rho \) to be lossless with respect to a set of functional and multivalued dependencies.

Keywords: Database; Data modeling; Dependencies; Decomposition; Lossless join

Computing Reviews Category: H.2.1

1. INTRODUCTION

We consider the question of when a given decomposition of a database relation scheme is lossless (i.e., has a lossless join) with respect to a given set of functional dependencies. A relation scheme is a set of attributes. Let \( R = \{ A_1, A_2, \ldots, A_k \} \) be a relation scheme, where the \( A_i \)'s are attributes. A decomposition of \( R \) is a collection \( \rho = \{ R_1, R_2, \ldots, R_m \} \) of relation schemes such that \( R_i \subseteq R \), for \( 1 \leq i \leq m \), and \( \bigcup_{i=1}^{m} R_i = R \). Let \( F = \{ F_1, F_2, \ldots, F_p \} \) be a set of functional dependencies on \( R \). For \( 1 \leq i \neq j \leq m \), let \( R_{ij} \) denote \( R_i \cap R_j \); let \( R_{i} = \bigcup_{j \neq i} R_{ij} \). For a set \( S \subseteq R \), let \( S^+ \) denote the closure of \( S \) with respect to \( F \).

Aho et al. [1] presented an algorithm for determining if a given decomposition \( \rho \) of \( R \) has a lossless join with respect to a given \( F \). The algorithm is quite intricate, and its actions are not clear at the macro level. In Section 2,
we present a reformulation of this algorithm in terms of set closures (Theorem 2.1). For the special case of \( m = 2 \), there is an explicit condition for losslessness [8, 4, 10]: \( \rho = \{ R_1, R_2 \} \) is lossless with respect to \( F \) iff \( R_{12} \rightarrow R_1 \) or \( R_{12} \rightarrow R_2 \); i.e., \( R^+_1 \) contains either \( R_1 \) or \( R_2 \). Our formulation (Theorem 2.1) extends this result for general \( m \).

Based on Aho et al.’s algorithm, Biskup et al. [3] and Loizou and Thanisch [9] proved the following necessary condition for \( \rho \) to be lossless:

If \( \rho \) is lossless then one of the \( R_i \)’s must contain a key for \( R \); i.e., for some \( i, 1 \leq i \leq m \), \( R_i^+ \) equals \( R \).

Their proof is based on a tedious analysis of Aho et al.’s algorithm. Vardi [12] proved the same result using some advanced tools from dependency theory. In Section 2, using our new formulation, we give a much simpler proof for a much stronger necessary condition (Theorem 2.3).

Also, Biskup et al. [3], Loizou and Thanisch [9] and Vardi [12] proved the following sufficient condition for \( \rho \) to be lossless:

If \( \rho \) preserves the dependencies in \( F \), then \( \rho \) is lossless if one of the \( R_i \)’s contains a key for \( R \).

In Section 3, we present a simpler proof for a weaker (i.e., easier to meet) sufficient condition for \( \rho \) to be lossless (Theorem 3.1).

Finally, in Section 4, we consider the question of when a given decomposition is lossless with respect to a given set \( D \) of functional and multivalued dependencies. Aho et al. [1] presented another algorithm for determining if a decomposition \( \rho \) is lossless with respect to \( D \); again, its actions are not clear at the macro level. For the special case of \( m = 2 \), there is an explicit condition for losslessness [5, 1, 11]: \( \rho = \{ R_1, R_2 \} \) is lossless with respect to \( D \) iff \( R_{12} \rightarrow R_1 \) (by [5], this condition is equivalent to \( R_{12} \rightarrow R_2 \)). Using Aho et al.’s algorithm, we prove that, for general \( m \), one extension of this condition is a sufficient condition (Theorem 4.3 and Corollary 4.4), and another extension is a necessary condition (Theorem 4.5) for \( \rho \) to be lossless with respect to \( D \).

For details about relation scheme, decomposition, functional/multivalued dependencies, and the lossless join property, we refer the reader to [11, Chapter 7].

2. A CONDITION EQUIVALENT TO LOSSLESSNESS

Aho et al. [1] presented an algorithm for determining if a decomposition \( \rho = \{ R_1, R_2, \ldots, R_m \} \) of \( R = \{ A_1, A_2, \ldots, A_k \} \) is lossless with respect to \( F \).
In this section, we first present a reformulation of this algorithm in terms of set closures.

Recall that a functional dependency \( X \rightarrow Y \) on \( R \) means the following: In any legal relation \( r \) for \( R \), if two tuples agree (i.e., have the same value) for all the attributes in \( X \), then the two tuples must also agree for all the attributes in \( Y \). Aho et al.’s algorithm works as follows (the following description is from [11, p. 394]). First construct a table \( T \) with \( m \) rows (\( i \)th row corresponds to \( R_i \)) and \( k \) columns (\( j \)th column corresponds to \( A_j \)). The table entries are initialized as follows:

\[
T(i,j) = \begin{cases} 
    a_{ij} & \text{if } A_j \in R_i \\
    b_{ij} & \text{otherwise}
\end{cases}
\]

Repeatedly “consider” each dependency \( X \rightarrow Y \) in \( F \), until no more changes can be made to the table. Each time we consider \( X \rightarrow Y \), we look for rows that agree in all of the columns for the attributes in \( X \). If we find two (or more) such rows, equate the symbols of those rows for each of the attributes (i.e., columns) in \( Y \). When we equate two symbols, if one of them is \( a_j \), make the other be \( a_j \). If they are \( b_{ij} \) and \( b_{ij} \), make them both \( b_{ij} \) or both \( b_{ij} \). When two symbols are equated, all occurrences of those symbols (in the same column, but in all the rows) in the table become the same.

After modifying the rows of the table as above, if some row consists only of \( a \) symbols, then the decomposition is lossless; otherwise, it is lossy.

While the individual steps of the algorithm are clear, their overall effect at the macro level is not clear. For the special case of \( m = 2 \), there is an explicit condition for losslessness [8, 4, 10]: \( \rho = \{ R_1, R_2 \} \) is lossless with respect to \( F \) iff \( R_{ij} \) contains either \( R_1 \) or \( R_2 \). The following reformulation of Aho et al.’s algorithm extends this result for general \( m \).

**Theorem 2.1** For \( 1 \leq i \neq j \leq m \), initialize \( S_{ij} \) to \( R_{ij}^+ \). Then, repeatedly set

\[
S_{ij} = [S_{ij} \cup (\cup_{k \neq i,j} (S_{ik} \cap S_{kj}))]^+
\]

This is done repeatedly, until no more changes occur to any \( S_{ij} \). Then, \( \rho \) is lossless iff there exists an \( i, 1 \leq i \leq m \), such that \( S_{ij} \) contains \( R_j \), for all \( j \neq i \).

**Proof** The table modification part of Aho et al.’s algorithm can be changed, without affecting the final result, as follows: Instead of separately considering each dependency in \( F \), repeatedly apply the following *procedure* to each pair of rows in \( T \):

Find the set \( S \) of columns in which the two rows agree;

then, for each column in \( S^+ \), equate the two symbols in these rows.
This procedure is applied repeatedly, for all pairs of rows, until no more equivalences can be found.

The final table that results from this modified algorithm would be same as the table that results from Aho et al.'s algorithm (except for the renaming of some bij's). In this final table, for \( 1 \leq i \neq j \leq m \), let \( S_{ij} \) be the set of columns in which rows \( i \) and \( j \) agree. With our modified algorithm, for any two rows \( i \) and \( j \), their entries in some column \( l \) could become equal in one of two ways:

1. Directly, when our procedure is applied to rows \( i \) and \( j \).
2. Indirectly, when our procedure is applied to pairs of rows \( (i, k_1), (k_1, k_2), \ldots, (k_{q-1}, k_q), (k_q, j) \), for some set \( \{k_1, k_2, \ldots, k_q\} \) of distinct row numbers.

Equation (1) captures exactly these two possibilities. In particular, the closure (i.e., \( + \)) takes care of the first possibility. For the second possibility, we can show by induction on \( q \), that there exists \( k \in \{k_1, k_2, \ldots, k_q\} \) (the value of \( k \) depends on the order in which Eq. (1) is applied), such that \( l \in S_{ik} \cap S_{kj} \); hence, \( l \) will be added to \( S_{ij} \). So, the final value of \( S_{ij} \) that results from repeatedly applying Eq. (1) would be the set of columns in which rows \( i \) and \( j \) agree in the final version of \( T \).

In the final table \( T \), row \( i \) would consist only of \( \alpha \) symbols iff \( S_{ij} \) contains \( R_j \), for all \( j \neq i \).

The concept embodied in the statement of Theorem 2.1 can be explained in terms of “information flow” as follows (see Fig. 1 for \( m = 3 \)). The information originating at each \( R_j \) consists of all the attributes in \( R = R_j \cup \bar{R}_j \). We want to see how much of the information originating at \( R_j \) reaches \( R_i \). We can think of \( S_{ij} \) as the “window” between \( R_i \) and \( R_j \). The information that can flow through a window \( S_{ij} \) is \( S_{ij}^c \); note that this information could include attributes not in \( R_i \cup R_j \). The information originating at \( R_j \) can reach \( R_i \) either directly through the window \( S_{ij} \) or indirectly through a sequence of windows \( S_{jk}, S_{kj}, \ldots, S_{ki} \), as discussed in the above proof. Also, as discussed in the above proof, by iteratively enlarging the windows as in Eq. (1), we only need to consider indirect transmission through two windows \( S_{jk} \) and \( S_{kj} \), for some \( k \in \{k_1, k_2, \ldots, k_q\} \).

Then, \( \rho \) is lossless iff there exists an \( i, 1 \leq i \leq m \), such that, in steady state (i.e., when the windows can not be enlarged any further using Eq. (1)), the information originating at \( R_j \) that reaches \( R_i \) contains \( R_j \), for all \( j \neq i \).

Using Theorem 2.1, we will prove a necessary condition for \( \rho \) to be lossless with respect to \( F \). First, we need the following lemma. The following
results can be understood easily in the light of the above information flow concept, though we give more formal proofs.

**Lemma 2.2** For $1 \leq i \neq j \leq m$, let $S_{ij}$ be as specified in Theorem 2.1. Then, $S_{ij} \subseteq R_{ij}^+$.  

**Proof** As specified in Theorem 2.1, the $S_{ij}$’s are computed recursively. The proof of the lemma is by simple induction on the recursion step.

Biskup et al. [3], Loizou and Thanisch [9] and Vardi [12] showed that the following condition is necessary (but not sufficient) for $\rho$ to be lossless: For some $i$, $1 \leq i \leq m$, $R_i^+$ equals $R$. Our next result gives a much stronger necessary condition.

**Theorem 2.3** If $\rho$ is lossless with respect to $F$, there exists $i$, $1 \leq i \leq m$, such that:

- $R_i$ contains a key for $\cup_{j \neq i} R_j$, and
- for all $j \neq i$, $R_j$ contains a key for $R_i$.

**Proof** Let $\rho$ be lossless with respect to $F$. For $1 \leq i \neq j \leq m$, let $S_{ij}$ be as specified in Theorem 2.1. By Lemma 2.2, $S_{ij} \subseteq R_{ij}^+$; by symmetry, $S_{ji} = S_{ji} \subseteq R_{ji}^+$. Theorem 2.1, there exists an $i$, $1 \leq i \leq m$, such that $S_{ij}$ contains $R_j$, for all $j \neq i$. Hence the result follows.

---

**FIGURE 1** Information flow.
3. A SUFFICIENT CONDITION FOR LOSSLESSNESS

In this section, using Aho et al.’s algorithm, we first present a sufficient condition for \( \rho \) to be lossless. Then, we show that this condition is also necessary for \( m = 3 \) (but not for \( m \geq 4 \)).

**Theorem 3.1** For \( 1 \leq i \leq m \), initialize \( S_i \) to \( R_i \). Then repeatedly do the following: Set \( F' = \bigcup_{j=1}^{m} F'_j \), where \( F'_j \) is the projection of \( F \) onto \( S_j \); take the closure of each \( S_j \) with respect to \( F' \). This is done repeatedly, until no more changes occur to any \( S_i \). If for some \( i, 1 \leq i \leq m \), \( S_i \) equals \( R \), then \( \rho \) is lossless.

**Proof** Consider Aho et al.’s algorithm described in Section 2. Let \( ALG_1 \) be the algorithm obtained by modifying that algorithm as follows: When Aho et al.’s algorithm wants to equate two symbols, \( ALG_1 \) will equate them only if one of those symbols is an \( a \). Clearly, in each row \( i \), if the final table obtained by \( ALG_1 \) has \( a_j \) in column \( j \), then the final table obtained by Aho et al.’s algorithm would also have \( a_j \) in column \( j \).

The table modification part of \( ALG_1 \) can be changed, without affecting the final table, as follows: Instead of separately considering each dependency in \( F \), repeatedly apply the following procedure to each pair of rows in \( T \):

- Find the set \( S \) of columns in which the two rows agree;
- then, for each column in \( S^+ \), equate the two symbols in these rows,
- if one of these two symbols is an \( a \).

This procedure is applied repeatedly, for all pairs of rows, until no more equivalences can be found.

Let this modified version of \( ALG_1 \) be called \( ALG_2 \). The final table that results from \( ALG_2 \) would be same as the table that results from \( ALG_1 \). The operation of \( ALG_2 \) does not depend on the \( b \) symbols, because all the \( b \) symbols are distinct to start with, and \( ALG_2 \) never equates any two of them.

Let \( ALG_3 \) be the following algorithm. For \( 1 \leq i \leq m \), initialize \( S_i \) to \( R_i \). For each pair \((i,j), 1 \leq i \neq j \leq m \), repeatedly do the following: Set \( S_i = S_i \cup [S_i \cap (S_i \cap S_j)^+] \), and \( S_j = S_j \cup [S_j \cap (S_i \cap S_j)^+] \). This is done repeatedly, for all pairs \((i,j)\), until no more changes occur.

For each pair \((i,j)\), the action of \( ALG_3 \) on \((S_i, S_j)\) is equivalent to the action of the procedure in \( ALG_2 \) on the pair of rows \((i,j)\). So, the final value of \( S_i \) from \( ALG_3 \) is exactly the set of columns containing an \( a \) symbol in row \( i \) of the final table produced by \( ALG_2 \) (and so, \( ALG_1 \)). As we have seen
What $ALG_3$ does is to repeatedly take the closure of each $S_i$ with respect to the projection $F'_j$ of $F$ onto each $S_j$, until no more changes occur to any $S_i$. This is equivalent to repeatedly taking the closure of each $S_i$ with respect to $F'_j = \bigcup_{j=1}^{m} F'_j$, until no more changes occur to any $S_i$. Hence the result follows.

Biskup et al. [3], Loizou and Thanisch [9] and Vardi [12] proved the following sufficient condition for $\rho$ to be lossless:

$$\text{If } \rho \text{ preserves the dependencies in } F, \text{ then }$$

$$\rho \text{ is lossless if one of the } R_i \text{s contains a key for } R.$$

Note that this result follows from Theorem 3.1: If $\rho$ preserves the dependencies in $F$, then at the beginning of the first iteration, we will have $F' = F$.

Now, we present an example to show that the condition of Theorem 3.1 is not a necessary condition for a decomposition to be lossless, for $m \geq 4$. Take $R = ABCDEH$, $\rho = \{R_1, R_2, R_3, R_4\}$, where $R_1 = ADEH$, $R_2 = BDE$, $R_3 = ABD$, and $R_4 = BCH$; $F = \{A \rightarrow C; BD \rightarrow C, CE \rightarrow B, BH \rightarrow C\}$. For $1 \leq i \leq 4$, $S_i$ mentioned in the theorem equals $R_i$. But, in the final table resulting from Aho et al.’s algorithm, the first row consists only of $a$ symbols; so $\rho$ is lossless.

As seen from this example, the reason why the condition of Theorem 3.1 is not a necessary condition for losslessness is as follows: In Aho et al.’s algorithm, in a particular column $l$, we could have the two symbols equated in rows $(i,j)$, and then in rows $(j,k)$. Now rows $(i,k)$ agree in column $l$, and so a dependency that has $A_l$ on its left hand side could be used to equate other symbols in rows $(i,k)$. This possibility is not accounted for by the condition in Theorem 3.1, exactly when all three rows $(i,j,k)$ have a $b$ symbol in column $l$. This can not happen for the case $m = 3$, since at least one of $R_1$, $R_2$ and $R_3$ must contain $A_l$. So, for $m = 3$, the condition of Theorem 3.1 is both necessary and sufficient:

**Corollary 3.2**  Let $m = 3$. For $1 \leq i \leq m$, initialize $S_i$ to $R_i$. Then repeatedly do the following: Set $F' = \bigcup_{j=1}^{m} F'_j$, where $F'_j$ is the projection of $F$ onto $S_j$; take the closure of each $S_i$ with respect to $F'$. This is done repeatedly, until no more changes occur to any $S_i$. Then $\rho$ is lossless iff for some $i$, $1 \leq i \leq m$, $S_i$ equals $R$.  

before, this is a subset of the columns that contain an $a$ symbol in row $i$ of the final table produced by Aho et al.’s algorithm.
4. SUFFICIENT/NECESSARY CONDITIONS FOR MULTIVALUED DEPENDENCIES

Aho et al. [1] presented an algorithm for determining if a decomposition \( \rho \) is lossless with respect to a set \( D \) of functional and multivalued dependencies. In this section, using their algorithm, we derive a sufficient condition and a necessary condition for \( \rho \) to be losslessness with respect to \( D \).

Recall that a multivalued dependency \( X \rightarrow Y \) on \( R \) means the following: In any legal relation \( r \) for \( R \), if two tuples \( t_1 \) and \( t_2 \) agree for all the attributes in \( X \), then the two new tuples obtained by swapping the values of \( t_1 \) and \( t_2 \) for all the attributes in \( Y \) must also be in \( r \). Aho et al.’s algorithm is a slight modification of their algorithm for functional dependencies, described in Section 2. When we “consider” a dependency in \( D \), if it is a functional dependency, we do as described in Section 2. When we “consider” a multivalued dependency \( X \rightarrow Y \) in \( D \), we do as follows: If we find two rows \( r_1 \) and \( r_2 \) that agree for all the attributes in \( X \), then the two new rows obtained by swapping the values of \( r_1 \) and \( r_2 \) for all the attributes in \( Y \) are added to the table. If some row consisting only of \( a \) symbols is eventually added to the table, then the decomposition is lossless; otherwise, it is lossy.

Aho et al.’s algorithm for functional dependencies, described in Section 2, can be implemented to run efficiently, in polynomial time. But their algorithm for functional and multivalued dependencies, described above, takes exponential time and space. Also, Fisher and Tsou [6] proved that determining if a decomposition \( \rho \) is lossless with respect to \( D \) is NP–Hard (for a definition of NP–Hardness, see [7]). Hence, it is unlikely that there is a nice condition equivalent to losslessness with respect to \( D \). So, we will be content with proving a sufficient condition, and a weaker necessary condition. To prove our results, we need the following result of Beeri [2].

**Theorem 4.1** [2] Let \( D \) be a set of functional and multivalued dependencies on \( R \). For \( X \subseteq R \), we can partition \( R - X \) into sets of attributes \( Y_1, Y_2, \ldots, Y_b \), such that for \( Z \subseteq R - X \), \( X \rightarrow Z \) iff \( Z \) is the union of some of the \( Y_i \)'s.

The partition \((Y_1, Y_2, \ldots, Y_b)\) referred to in the above theorem is called the dependency basis of \( X \) with respect to \( D \). Now, we have the following.

**Theorem 4.2** Let \( X \subseteq X' \subseteq R \); let \((Y_1, Y_2, \ldots, Y_b)\) be the dependency basis of \( X \) with respect to \( D \). Then the dependency basis of \( X' \) with respect to \( D \) is a refinement of the partition \((Y_1 - X', Y_2 - X', \ldots, Y_b - X')\).

**Proof** Refer to [11, pp. 414–415] for axioms for functional and multivalued dependencies. We need to prove that if \( X \rightarrow Z \) for some
Z \subseteq R - X, then \( X' \rightarrow Z - X' \). Let \( X \rightarrow Z \). Since \( X' \rightarrow X \) (by reflexivity), we have \( X' \rightarrow Z \), by transitivity. Also, since \( X' \rightarrow X' \) (by reflexivity) and \( X' \rightarrow Z \), we have \( X' \rightarrow Z - X' \), by transitivity.

For the special case of \( m = 2 \), there is an explicit condition for losslessness \([5, 1, 11]\): \( \rho = \{R_1, R_2\} \) is lossless with respect to \( D \) iff \( R_12 \rightarrow R_1 \) (by Theorem 4.1, this condition is equivalent to \( R_12 \rightarrow R_2 \)); i.e., the dependency basis of \( R_{12} \) does not contain any part that overlaps both \( R_1 \) and \( R_2 \). We prove that, for general \( m \), one extension of this condition is a sufficient condition and another extension is a necessary condition for \( \rho \) to be lossless with respect to \( D \).

**Theorem 4.3** For \( 2 \leq i \leq m \), let \( R'_i \) denote \( R_i \cap (\bigcup_{j<i} R_j) = \bigcup_{j<i} R_{ij} \); suppose that the dependency basis of \( R'_i \) with respect to \( D \) does not contain any part that overlaps both \( R_i \) and \( \bigcup_{j<i} R_j \). Then \( \rho \) is lossless with respect to \( D \).

**Proof** We will show how to create a row of all \( a \) symbols in the table \( T \) in Aho et al.’s algorithm. This row, row \( m+1 \), is built recursively. It is initialized to a copy of the first row. For \( 2 \leq i \leq m \), the next value of row \( m+1 \) is obtained from its current value, and row \( i \). We will show that the new row will have only \( a \) symbols in the columns for the attributes in \( \bigcup_{j=1}^i R_j \). So, after \( i = m \), the new row will have only \( a \) symbols. By the hypothesis of this theorem, there exists \( Z \), \( R_i - R'_i \subseteq Z \subseteq R - \bigcup_{j<i} R_j \), such that \( R'_i \rightarrow Z \). During the \( i \)th recursive step in the previous paragraph (when we are modifying row \( m+1 \) using row \( i \)), we replace the components of row \( m+1 \) for the attributes in \( Z \), by the corresponding components in row \( i \). Then the new row will have only \( a \) symbols in the columns for the attributes in \( \bigcup_{j=1}^i R_j \).

**Corollary 4.4** Suppose that, for all \( i, j, 1 \leq i \neq j \leq m \), the dependency basis of \( R_{ij} \) with respect to \( D \) does not contain any part that overlaps both \( R_i \) and \( R_j \). Then \( \rho \) is lossless with respect to \( D \).

**Proof** For \( 2 \leq i \leq m \), let \( R'_i \) denote \( R_i \cap (\bigcup_{j<i} R_j) = \bigcup_{j<i} R_{ij} \). By the hypothesis of this corollary, and by Theorem 4.2, the dependency basis of \( R'_i \) does not contain any part that overlaps both \( R_i \) and \( \bigcup_{j<i} R_j \). Then the result follows from Theorem 4.3.

**Theorem 4.5** Suppose that \( \rho \) is lossless with respect to \( D \). Then, for \( 1 \leq i \leq m \), the dependency basis of \( R_i \) with respect to \( D \) does not contain any part that overlaps both \( R_i \) and \( \bigcup_{j \neq i} R_j \); i.e., \( R_i \rightarrow \bigcup_{j \neq i} R_j \).

**Proof** By the definition of lossless join, if \( \rho = \{R_1, R_2, \ldots, R_m\} \) is lossless, then, for \( 1 \leq i \leq m \), \( \{R_i, \bigcup_{j \neq i} R_j\} \) must be a lossless decomposition into two
schemes. Then the theorem follows from the known result for decomposition into two schemes.

References