Equivalences Among Relational Expressions with the Union and Difference Operators

YEHOSHUA SAGIV AND MIHALIS YANNAKAKIS

Princeton University, Princeton, New Jersey

ABSTRACT Queries in relational databases can be formulated in terms of relational expressions using the relational operations select, project, join, union, and difference The equivalence problem for these queries is studied with query optimization in mind It is shown that testing equivalence of relational expressions with the operators select, project, join, and union is complete in the class Π_2^P of the polynomial-time hierarchy A nonprocedural representation for queries formulated by these expressions is proposed This method of query representation can be viewed as a generalization of tableaux or conjunctive queries (which are used to represent expressions with only select, project, and join) Furthermore, this method is extended to queries formulated by relational expressions that also contain the difference operator, provided that the project operator is not applied to subexpressions with the difference operator A procedure for testing equivalence of these queries is given It is shown that testing containment of tableaux is a necessary step in testing equivalence of queries with union and difference Three important cases in which containment of tableaux can be tested in polynomial time are described, although the containment problem is shown to be NP-complete even for tableaux that correspond to expressions with only one project and several join operators

KEY WORDS AND PHRASES relational database, relational algebra, query optimization, equivalence of queries, conjunctive query, tableau, NP-complete, polynomial-time hierarchy, Π_2^P-complete

CR CATEGORIES 4 33, 5 25

1. *Introduction*

A number of papers have considered the problem of optimizing queries in a relational database (e.g., [15, 17–19, 21, 23]). These papers develop transformations that reduce the cost of evaluating a given query but do not necessarily produce an equivalent query of least cost. As a first step toward globally optimizing queries, we consider the problem of determining whether two queries are equivalent.

The equivalence problem has been investigated so far only for a restricted subclass of queries. The first results were obtained for the class of conjunctive queries [9]. Every conjunctive query has a unique minimal form, and both the minimization and equivalence problems for conjunctive queries are NP-complete [4, 9]. In [4] tableaux are proposed as a two-dimensional representation of queries. Tableaux are similar to conjunctive queries, and every relational expression over the operators select, project, and join can be represented by a tableau. Polynomial minimization and equivalence algorithms have been developed for an important subclass of tableaux (called simple tableaux) [4, 5], and an optimal relational expression can be obtained in polynomial time from a minimal simple tableau [3].

In this paper we extend the theory of tableaux to all relational expressions over select,

This work was supported in part by the National Science Foundation under Grant MCS-76-15255.

Authors' present addresses Y Sagiv, Department of Computer Science, University of Illinois, Urbana, IL 61801, M Yannakakis, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

project, join, and union. In Section 3 we introduce unions of tableaux and show that every relational expression over select, project, join, and union can be represented as a union of tableaux. In Section 4 we characterize equivalence of unions of tableaux in terms of containment of tableaux, and in Section 5 we prove that every union of tableaux has a unique minimal form. In Sections 6 and 7 this approach is further extended to all relational expressions over select, project, join, union, and difference that do not contain an application of the project operator to a subexpression with the difference operator.

In Sections 8–10 we deal with complexity issues. The results of the previous sections imply that testing containment (rather than equivalence) of tableaux is a necessary step in optimizing queries that contain union and difference. In Section 8 we show that this problem is NP-complete even for tableaux that correspond to expressions with only one project and several join operators. In Section 9 we characterize three special cases for which polynomial containment algorithms for tableaux exist. We feel that most practical queries over select, project, join, union, and difference can be handled by using these algorithms. In Section 10 we show that testing equivalence of relational expressions over select, project, join, and union is complete in the class Π_2^P of the polynomial-time hierarchy of [22]. We conclude the paper with a discussion of some remaining problems.

2. *Basic Definitions*

2.1 The Relational Model. The relational database model [10] assumes that the data are stored in tables called *relations*. The columns of a table correspond to *attributes*, and the rows to *records* or *tuples*. Each attribute has an associated *domain* of values. A tuple is viewed as a mapping from the attributes to their domains, since no canonical ordering of the attributes is needed in this way. If r is a relation with a column corresponding to the attribute A and μ is a tuple in r, then $\mu(A)$ is the value of the A-component of μ. In this paper we usually denote a relation as a set of tuples.

A *relation scheme* is a set of attributes labeling the columns of a table, and it is usually written as a string of attributes. We often use the relation scheme itself as the name of the table. A relation is just the "current value" of a relation scheme. The relation is said to be *defined on* the set of attributes of the relation scheme.

2.2 The Relational Algebra and Relational Expressions. The relational algebra [10, 11] is a set of operators defined on relations. In this paper we consider the operators select, project, join, union, and difference.

Let r be a relation defined on a set of attributes X, A an attribute in X, and c a value from the domain of A. The *selection* $A = c$, written $\sigma_{A=c}(r)$, is the set

$$\sigma_{A=c}(r) = \{\mu \mid \mu \text{ is a tuple in } r \text{ and } \mu(A) = c\}.$$

Let Y be a subset of X. The *projection* of r onto Y, written $\pi_Y(r)$, is

$$\pi_Y(r) = \{\mu \mid \mu \text{ is a mapping on } Y, \text{ and there is a } \lambda \text{ in } r \text{ that agrees with } \mu \text{ on } Y\}.$$

Let r_1 and r_2 be relations defined on the relation schemes R_1 and R_2, respectively. The (natural) *join* of r_1 and r_2, written $r_1 \bowtie r_2$, is

$$r_1 \bowtie r_2 = \{\mu \mid \mu \text{ is a mapping on } R_1 \cup R_2, \text{ and there is a } \lambda_1 \text{ in } r_1 \text{ that agrees with } \mu \text{ on } R_1 \text{ and a } \lambda_2 \text{ in } r_2 \text{ that agrees with } \mu \text{ on } R_2\}.$$

Note that if r_1 and r_2 are defined on the same relation scheme, then the result is the intersection of r_1 and r_2.

Let r_1 and r_2 be two relations defined on the same set of attributes. The *union* of r_1 and r_2, written $r_1 \cup r_2$, is

$$r_1 \cup r_2 = \{\mu \mid \mu \text{ is in } r_1 \text{ or } \mu \text{ is in } r_2\}.$$

Let r_1 and r_2 be two relations defined on the same set of attributes. The *difference* of r_1 and r_2, written $r_1 - r_2$, is

$$r_1 - r_2 = \{\mu \mid \mu \text{ is in } r_1 \text{ but not in } r_2\}.$$

Note that to make our set of operators "complete" (in the sense of Codd [11]), we only need to allow selections involving arithmetic comparisons between two components of a tuple.

Relational expressions are one way of formulating queries in relational databases. A relational expression consists of relation schemes as operands, and select, project, join, union, and difference as operators. A *restricted relational expression* is a relational expression in which the only operators are select, project, and join A *monotonic relational expression* is a relational expression in which the only operators are select, project, join, and union.

2.3 EXPRESSION VALUES AND EQUIVALENCE OF EXPRESSIONS. An underlying assumption in many papers (e.g., [1, 7, 8]) is the existence of a single universal relation at each instant of time. This relation is defined on the set of all the attributes, and it is called a *universal instance* or just an *instance*. If I is an instance and E is a relational expression, then each relation scheme R_i in E is assigned the relation $\pi_{R_i}(I)$. The value of E with respect to I, written $v_I(E)$, is computed by applying operators to operands in the following natural way.

(1) If E is a single relation scheme R_i, then $v_I(E) = \pi_{R_i}(I)$.
(2a) If $E = \sigma_{A=c}(E_1)$, then $v_I(E) = \sigma_{A=c}(v_I(E_1))$.
(2b) If $E = \pi_X(E_1)$, then $v_I(E) = \pi_X(v_I(E_1))$.
(2c) If $E = E_1 \bowtie E_2$, then $v_I(E) = v_I(E_1) \bowtie v_I(E_2)$.
(2d) If $E = E_1 \cup E_2$, then $v_I(E) = v_I(E_1) \cup v_I(E_2)$.
(2e) If $E = E_1 - E_2$, then $v_I(E) = v_I(E_1) - v_I(E_2)$.

We may regard a relational expression as a mapping from instances to relations, i.e., the expression E maps the instance I to the relation $v_I(E)$. Two expressions E_1 and E_2 are *equivalent* if they define the same mapping, that is, if for all instances I, $v_I(E_1) = v_I(E_2)$. In [4] other types of equivalence are discussed, and it is shown that results obtained for the above type of equivalence also apply to the more general case where the relations assigned to the relation schemes do not necessarily come from an instance.

2.4 TABLEAUX. Tableaux are just another way of representing mappings from instances to relations. Unlike relational expressions, tableaux are nonprocedural representations of queries in exactly the sense that relational calculus [10, 11] is nonprocedural. In [4] it is shown that every restricted relational expression has a corresponding tableau that represents the same mapping.

A *tableau* is a matrix consisting of a *summary* and a set of *rows*. The columns of a tableau correspond to the attributes of the universe in a fixed order. The symbols appearing in a tableau are chosen from

(1) distinguished variables, usually denoted by subscripted *a*'s;
(2) nondistinguished variables, usually denoted by subscripted *b*'s;
(3) constants, which are drawn from the domains of the attributes;
(4) blanks.

Each row may contain constants, distinguished variables, and nondistinguished variables. The summary has constants, distinguished variables, and blanks. A variable cannot appear in more than one column, and a distinguished variable may appear in a column only if it appears in the summary of that column. Furthermore, if a constant or a distinguished variable appears in some column A of the summary, then it also appears in column A of some row.

Let T be a tableau with a summary w_0 and rows $w_1, \ldots, w_n$, and let S be the set of all the nonblank symbols in T. A *valuation* ρ for T maps each symbol in S to a constant, such that if c is a constant in S, then $\rho(c) = c$. The valuation ρ is extended to the rows and summary of T by defining $\rho(w_i)$ to be the result of substituting $\rho(v)$ for every variable v that appears in w_i.

The tableau T defines a mapping from instances to relations on its *target relation scheme*, which is the set of all the attributes corresponding to columns that have a nonblank symbol in the summary. Given an instance I, the value of T, written $T(I)$, is

$$T(I) = \{\rho(w_0) \mid \text{for some valuation } \rho \text{ we have } \rho(w_i) \text{ in } I \text{ for } 1 \leq i \leq n\}$$

Conventionally, we also regard $\varnothing$ as a tableau. This tableau represents the function that maps every instance to the empty relation.

Example 1 Consider the following tableaux over the attributes A, B, and C.

$$T_1 = \begin{array}{|ccc|} \multicolumn{1}{c}{A} & \multicolumn{1}{c}{B} & \multicolumn{1}{c}{C} \\ \hline a_1 & & a_3 \\ \hline a_1 & 2 & b_3 \\ b_1 & b_2 & a_3 \\ \hline \end{array}, \qquad T_2 = \begin{array}{|ccc|} \multicolumn{1}{c}{A} & \multicolumn{1}{c}{B} & \multicolumn{1}{c}{C} \\ \hline a_1 & & 1 \\ \hline a_1 & b_3 & 1 \\ \hline \end{array}$$

The summary is shown first, with a line below it, and integers are used as constants Both T_1 and T_2 define relations on the relation scheme AC. Suppose that I is the instance $\{211, 121, 122\}$.

Consider the valuation ρ that assigns 2 to b_2 and 1 to every other variable in T_1 Under this valuation each row of T_1 becomes 121, which is a member of I. Therefore $\rho(a_1a_3) = 11$ is in $T_1(I)$.

If ρ assigns 1 to a_1, b_1, and b_3, and 2 to b_2 and a_3, the first row becomes 121 and the second row becomes 122; both are members of I, and so 12 is in $T_1(I)$.

Since no valuation for T_1 produces a tuple other than 11 or 12, $T_1(I) = \{11, 12\}$ Similarly, $T_2(I) = \{11, 21\}$. □

2.5 TABLEAUX AND CONJUNCTIVE QUERIES. A tableau T on variables $a_1, \ldots, a_k, b_1, \ldots, b_m$, with rows $w_1, \ldots, w_n$ (and without constants) is equivalent to the query

$$(a_1, \ldots, a_k).(\exists b_1)\cdots(\exists b_m)I(w_1) \wedge \cdots \wedge I(w_n)$$

Traditionally, the value of the above query is the set of all tuples $(a_1, \ldots, a_k)$ such that there are values for $b_1, \ldots, b_m$ for which all of $w_1, \ldots, w_n$ are tuples in the relation I. A constant c can be introduced into the above query either by replacing a nondistinguished variable b_j with c, or replacing a distinguished variable a_j with c. In the second case the summary becomes $(a_1, \ldots, a_{j-1}, c, a_{j+1}, \ldots, a_k)$. In [9] a query of this type, i.e., an existentially quantified conjunction of relation schemes, is called a *conjunctive query*. Conjunctive queries are closely related to tableaux. The significant differences between tableaux and conjunctive queries are discussed in [4]. In this paper we generalize the concept of tableaux by considering also the union and difference operators. Analogous results are easily obtained for conjunctive queries.

3. *Monotonic Relational Expressions and Unions of Tableaux*

Every restricted relational expression E has a corresponding tableau T that defines the same mapping as E (however, the converse is not true) [4] In order to obtain a similar result for monotonic relational expressions, we have to introduce unions of tableaux. A *union of tableaux* is an expression $\bigcup_{i=1}^{n} T_i$, where $T_1, T_2, \ldots, T_n$ are tableaux having the same target relation scheme. The common target relation scheme of $T_1, T_2, \ldots, T_n$ is also the target relation scheme of $\bigcup_{i=1}^{n} T_i$. A union of tableaux $\bigcup_{i=1}^{n} T_i$ defines a mapping from instances to relations on the target relation scheme of $\bigcup_{i=1}^{n} T_i$; an instance I is mapped to the relation $(\bigcup_{i=1}^{n} T_i)(I)$ defined by

$$\left(\bigcup_{i=1}^{n} T_i\right)(I) = \bigcup_{i=1}^{n} T_i(I).$$

Example 2. Consider the tableaux T_1 and T_2 in Example 1. Let I be the same instance as in Example 1. The union of tableaux $T_1 \cup T_2$ maps the instance I to the relation {11, 12, 21} defined on AC. □

In this section we prove that every monotonic relational expression can be represented by a union of tableaux. The proof is constructive and it uses the following rules for obtaining a tableau T that represents a restricted relational expression E [4]:

(A) If there are no operators in E, then E is a single relation scheme R. The tableau T for E has one row and a summary such that

(i) If A is an attribute in R, then in the column for A the tableau T has the same distinguished variable in the summary and row.
(ii) If A is not in R, then its column has a blank in the summary and a nondistinguished variable in the row.

(B1) Suppose E is of the form $\sigma_{A=c}(E_1)$, and we have constructed T_1, the tableau for E_1.

(i) If the summary of T_1 has a blank in the column for A, then the expression E has no meaning, and the tableau T for E is undefined.
(ii) If T_1 has a constant $c' \neq c$ in the summary column for A, then for any instance I, $v_I(E)$ has only tuples with c' in the component for A, and $v_I(E)$ is $\emptyset$. Hence, the tableau for E is $\emptyset$.
(iii) If T_1 has c in the summary column for A, then the tableau for E is T_1.
(iv) If T_1 has a distinguished variable a in the summary column for A, the tableau T for E is constructed by replacing a with c wherever it appears in T_1.

(B2) Suppose E is of the form $\pi_X(E_1)$, and T_1 is the tableau for E_1. The tableau T for E is constructed by replacing nonblank symbols with blanks in the summary of T_1 for those columns whose attributes are not in X. Distinguished variables in those columns become nondistinguished.

(B3) Suppose E is of the form $E_1 \bowtie E_2$, and T_1 and T_2 are the tableaux for E_1 and E_2, respectively.

(i) If T_1 and T_2 have some column in which their summaries have distinct constants, then it is easy to show that for all instances I, $v_I(E) = \emptyset$; so $\emptyset$ is the tableau for E.
(ii) If no corresponding positions in the summaries have distinct constants, let S_1 and S_2 be the sets of symbols of T_1 and T_2, respectively. We may take S_1 and S_2 to have disjoint sets of nondistinguished variables, but identical distinguished variables in corresponding columns. Construct T for E to have the union of all the rows of T_1 and T_2. The summary of T has in a given column:

(a) The constant c if one or both of T_1 and T_2 have c in that column's summary. In this case replace any distinguished variable in that column with c.
(b) The distinguished symbol a if (a) does not apply, but one or both of T_1 and T_2 have a in that column's summary.
(c) Blank, otherwise.

These rules can also be used to define the operations select, project, and join on tableaux. The result of applying any one of these operators to tableaux (not necessarily tableaux derived from restricted relational expressions) is defined to be the tableau described in the rule for that operator.

We can now describe the rules for constructing a union of tableaux for a monotonic relational expression E. The union of tableaux for E is constructed bottom up as follows.

(1) If E is a single relation scheme, then there is a single tableau T that corresponds to E (see rule (A) above). The union of tableaux Y for E is T.

(2a) Suppose E is of the form $\sigma_{A=c}(E_1)$, and $Y_1 = \bigcup_{i=1}^{m} T_i$ is the union of tableaux for E_1. The union of tableaux Y for E is obtained by applying the select operator to each tableau in Y_1; i.e., $Y = \bigcup_{i=1}^{m} \sigma_{A=c}(T_i)$.

(2b) Suppose E is of the form $\pi_X(E_1)$, and $Y_1 = \bigcup_{i=1}^{m} T_i$ is the union of tableaux for E_1 The union of tableaux Y for E is obtained by applying the project operator to each tableau in Y_1; i.e., $Y = \bigcup_{i=1}^{m} \pi_X(T_i)$

(2c) Suppose E is of the form $E_1 \bowtie E_2$, and $Y_1 = \bigcup_{i=1}^{m} T_i^1$ and $Y_2 = \bigcup_{j=1}^{n} T_j^2$ are the unions of tableaux for E_1 and E_2, respectively. The union of tableaux Y for E is obtained by joining each tableau in Y_1 with every tableau in Y_2; i.e., $Y = \bigcup_{i=1}^{m} \bigcup_{j=1}^{n} T_i^1 \bowtie T_j^2$.

(2d) Suppose E is of the form $E_1 \cup E_2$, and $Y_1 = \bigcup_{i=1}^{m} T_i$ and $Y_2 = \bigcup_{i=m+1}^{n} T_i$ are the unions of tableaux for E_1 and E_2, respectively. Both Y_1 and Y_2 must have the same target relation scheme; otherwise, E is undefined. The union of tableaux Y for E is the union of all the tableaux in Y_1 and Y_2; i.e., $Y = \bigcup_{i=1}^{n} T_i$

Example 3. Let A, B, C, and D be the attributes, and suppose we have the expression $\pi_{BD}(((\sigma_{B=0}(AB)) \cup (\pi_{AB}(\sigma_{C=1}(AB \bowtie BC)))) \bowtie AD)$. By rule (1), a union of tableaux for AB is T_1, where

$T_1 =$

A	B	C	D
a_1	a_2		
a_1	a_2	b_1	b_2

.

By rule (2a), a union of tableaux for $\sigma_{B=0}(AB)$ is T_2, where

$T_2 =$

A	B	C	D
a_1	0		
a_1	0	b_1	b_2

.

Note that $T_2 = \sigma_{B=0}(T_1)$. By rule (1) a union of tableaux for BC is T_3, where

$T_3 =$

A	B	C	D
	a_2	a_3	
b_3	a_2	a_3	b_4

By rule (2c) a union of tableaux for $AB \bowtie BC$ is T_4, where

$T_4 =$

A	B	C	D
a_1	a_2	a_3	
a_1	a_2	b_1	b_2
b_3	a_2	a_3	b_4

.

Note that $T_4 = T_1 \bowtie T_3$. By rule (2a) a union of tableaux for $\sigma_{C=1}(AB \bowtie BC)$ is T_5, where

$T_5 =$

A	B	C	D
a_1	a_2	1	
a_1	a_2	b_1	b_2
b_3	a_2	1	b_4

.

Note that $T_5 = \sigma_{C=1}(T_4)$. By rule (2b) a union of tableaux for $\pi_{AB}(\sigma_{C=1}(AB \bowtie BC))$ is T_6, where

$$T_6 = \begin{array}{|cccc|} \multicolumn{1}{c}{A} & B & C & \multicolumn{1}{c}{D} \\ \hline a_1 & a_2 & & \\ \hline a_1 & a_2 & b_1 & b_2 \\ b_3 & a_2 & 1 & b_4 \\ \hline \end{array} .$$

Note that $T_6 = \pi_{AB}(T_5)$. By rule (2d) a union of tableaux for $(\sigma_{B=0}(AB)) \cup (\pi_{AB}(\sigma_{C=1}(AB \bowtie BC)))$ is $T_2 \cup T_6$. By rule (1) a union of tableaux for AD is T_7, where

$$T_7 = \begin{array}{|cccc|} \multicolumn{1}{c}{A} & B & C & \multicolumn{1}{c}{D} \\ \hline a_1 & & & a_4 \\ \hline a_1 & b_5 & b_6 & a_4 \\ \hline \end{array} .$$

By rule (2c) a union of tableaux for $((\sigma_{B=0}(AB)) \cup (\pi_{AB}(\sigma_{C=1}(AB \bowtie BC)))) \bowtie AD$ is $T_8 \cup T_9$, where

$$T_8 = \begin{array}{|cccc|} \multicolumn{1}{c}{A} & B & C & \multicolumn{1}{c}{D} \\ \hline a_1 & 0 & & a_4 \\ \hline a_1 & 0 & b_1 & b_2 \\ a_1 & b_5 & b_6 & a_4 \\ \hline \end{array} , \qquad T_9 = \begin{array}{|cccc|} \multicolumn{1}{c}{A} & B & C & \multicolumn{1}{c}{D} \\ \hline a_1 & a_2 & & a_4 \\ \hline a_1 & a_2 & b_1 & b_2 \\ b_3 & a_2 & 1 & b_4 \\ a_1 & b_5 & b_6 & a_4 \\ \hline \end{array}$$

Note that $T_8 = T_2 \bowtie T_7$ and $T_9 = T_6 \bowtie T_7$. Finally, by rule (2b) a union of tableaux for the whole expression is $T_{10} \cup T_{11}$, where

$$T_{10} = \begin{array}{|cccc|} \multicolumn{1}{c}{A} & B & C & \multicolumn{1}{c}{D} \\ \hline & 0 & & a_4 \\ \hline b_7 & 0 & b_1 & b_2 \\ b_7 & b_5 & b_6 & a_4 \\ \hline \end{array} , \qquad T_{11} = \begin{array}{|cccc|} \multicolumn{1}{c}{A} & B & C & \multicolumn{1}{c}{D} \\ \hline & a_2 & & a_4 \\ \hline b_8 & a_2 & b_1 & b_2 \\ b_3 & a_2 & 1 & b_4 \\ b_8 & b_5 & b_6 & a_4 \\ \hline \end{array} .$$

Note that $T_{10} = \pi_{BD}(T_8)$ and $T_{11} = \pi_{BD}(T_9)$. □

THEOREM 1. *The above rules construct for every monotonic relational expression E a union of tableaux $Y = \bigcup_{i=1}^{n} T_i$ such that for all instances I, $v_I(E) = Y(I)$.*

PROOF. The proof is an induction on the number of operators in E.

Basis. Rule (1). If there are no operators in E, then E is a single relation scheme. It follows from the definition of the application of a tableau to an instance that for all instances I, $v_I(E) = Y(I)$.

Induction. Rule (2a). $E = \sigma_{A=c}(Y_1(I))$. We have to show that for all instances I, $Y(I) = \sigma_{A=c}(Y_1(I))$. This is proved as follows.

$$\begin{aligned} Y(I) &= \bigcup_{i=1}^{m} (\sigma_{A=c}(T_i))(I) \\ &= \bigcup_{i=1}^{m} \sigma_{A=c}(T_i(I)) \\ &= \sigma_{A=c}\left(\bigcup_{i=1}^{m} T_i(I)\right) \\ &= \sigma_{A=c}(Y_1(I)). \end{aligned}$$

The second of the four equalities follows from Theorem 1 of [4].

Rule (2*b*). $E = \pi_X(E_1)$. In a similar fashion to rule (2a),

$$\begin{aligned} Y(I) &= \bigcup_{i=1}^{m} (\pi_X(T_i))(I) \\ &= \bigcup_{i=1}^{m} \pi_X(T_i(I)) \\ &= \pi_X\left(\bigcup_{i=1}^{m} T_i(I)\right) \\ &= \pi_X(Y_1(I)). \end{aligned}$$

Again, the second equality follows from Theorem 1 of [4].

Rule (2*c*). $E = E_1 \bowtie E_2$.

$$\begin{aligned} Y(I) &= \bigcup_{i=1}^{m} \bigcup_{j=1}^{n} (T_i^1 \bowtie T_j^2)(I) \\ &= \bigcup_{i=1}^{m} \bigcup_{j=1}^{n} T_i^1(I) \bowtie T_j^2(I) \\ &= \left(\bigcup_{i=1}^{m} T_i^1(I)\right) \bowtie \left(\bigcup_{j=1}^{n} T_j^2(I)\right) \\ &= Y_1(I) \bowtie Y_2(I). \end{aligned}$$

The second equality follows from Theorem 1 of [4].

Rule (2*d*). $E = E_1 \cup E_2$. Both Y_1 and Y_2 must have the same target relation scheme; otherwise, the expression E has no meaning. By definition,

$$Y(I) = (Y_1 \cup Y_2)(I) = Y_1(I) \cup Y_2(I).$$

It is interesting to note that if a union of tableaux Y comes from a monotonic relational expression, then each tableau in Y corresponds to a restricted relational expression. For example, the tableau T_{10} in Example 3 corresponds to the expression

$$\pi_{BD}((\sigma_{B=0}(AB)) \bowtie AD),$$

and the tableau T_{11} corresponds to the expression

$$\pi_{BD}((\pi_{AB}(\sigma_{C=1}(AB \bowtie BC))) \bowtie AD).$$

Thus a union of tableaux obtained from a monotonic relational expression can be viewed as representing an equivalent expression in which the union operator is applied only in the last step.

4. *Testing Equivalence of Unions of Tableaux*

Two unions of tableaux Y_1 and Y_2 are ***equivalent***, written $Y_1 \equiv Y_2$, if for all instances I, $Y_1(I) = Y_2(I)$. We say that Y_1 is ***contained in*** Y_2, written $Y_1 \subseteq_T Y_2$, if for all I, $Y_1(I) \subseteq Y_2(I)$. The main result of this section is that equivalence and containment of unions of tableaux can be characterized in terms of containment of single tableaux.

Let T_1 and T_2 be tableaux with the same target relation scheme, and let S_1 and S_2 be the sets of symbols of T_1 and T_2, respectively. A *homomorphism* is a mapping $\xi: S_1 \rightarrow S_2$ such that

(a) If c is a constant, then $\xi(c) = c$.
(b) If s is the summary of T_1, then $\xi(s)$ is the summary of T_2.
(c) If w is any row of T_1, then $\xi(w)$ is a row of T_2.

By condition (c) a homomorphism ξ corresponds to a mapping θ from the rows of T_1 to the rows of T_2. The mapping θ is called a *containment mapping*, and it satisfies the following conditions.

(1) If row w of T_1 has a constant in some column A, then $\theta(w)$ has the same constant in column A. Furthermore, if that constant appears in column A of the summary of T_1, then it also appears in column A of the summary of T_2.
(2) If row w of T_1 has a distinguished variable in column A, then $\theta(w)$ has a distinguished variable in column A, or a constant if that constant appears in column A of the summary of T_2.
(3) If rows w and v have the same nondistinguished variable in column A, then rows $\theta(w)$ and $\theta(v)$ have the same symbol in column A.

The following lemma is proved in [4, 9].

LEMMA 2. *$T_2 \subseteq_T T_1$ if and only if T_1 and T_2 have the same target relation scheme and there is a containment mapping from the rows of T_1 to the rows of T_2 (or a homomorphism $\xi: S_1 \rightarrow S_2$).*

The proof of Lemma 2 implies the existence of a "magic" instance I and a "magic" tuple μ (both depending only on T_2) such that $T_2 \subseteq_T T_1$ if and only if μ is in $T_1(I)$. Let ρ be a one-to-one valuation for T_2. A magic instance I and a magic tuple μ for T_2 are obtained from ρ as follows. The instance I consists of all tuples $\rho(w)$ for w a row of T_2. The tuple μ is $\rho(s)$, where s is the summary of T_2. If $T_1(I)$ contains μ, then the existence of a containment mapping from T_1 to T_2 follows immediately (because a valuation for T_1 implying that μ is in $T_1(I)$ corresponds to a homomorphism $\xi: S_1 \rightarrow S_2$) Conversely, if $T_1(I)$ does not contain μ, then $T_2 \not\subseteq_T T_1$ (because $T_2(I)$ contains μ). By using this technique, we can prove the following theorem.

THEOREM 3. *Let $Y = \bigcup_{i=1}^{n} T_i$ and $Y' = \bigcup_{j=1}^{m} T'_j$ be unions of tableaux. $Y \subseteq_T Y'$ if and only if Y and Y' have the same target relation scheme and each tableau T_i of Y is contained in some tableau T'_j of Y'.*

PROOF. *If.* Since each tableau T_i of Y is contained in some T'_j of Y', it follows that for all instances I, $\bigcup_{i=1}^{n} T_i(I) \subseteq \bigcup_{j=1}^{m} T'_j(I)$. Thus $Y \subseteq_T Y'$.

Only if. Let T_i be a tableau in Y, and suppose that I is a magic instance and μ is a magic tuple for T_i. Since μ is in $T_i(I)$, there is a T'_j in Y' such that $T'_j(I)$ contains μ. But this implies the existence of a containment mapping from T'_j to T_i, and hence $T_i \subseteq_T T'_j$. □

COROLLARY 4. *Let $Y = \bigcup_{i=1}^{n} T_i$ and $Y' = \bigcup_{j=1}^{m} T'_j$ be two unions of tableaux. $Y \equiv Y'$ if and only if Y and Y' have the same target relation scheme, each tableau T_i of Y is contained in some tableau T'_j of Y', and each tableau T'_j of Y' is contained in some tableau T_i of Y.*

PROOF. The corollary follows from the fact that $Y \equiv Y'$ if and only if $Y \subseteq_T Y'$ and $Y' \subseteq_T Y$. □

Example 4. Let $Y_1 = \bigcup_{i=1}^{3} T_i$ and $Y_2 = \bigcup_{i=4}^{5} T_i$, where

$$T_1 = \begin{array}{|ccc|} \multicolumn{1}{c}{A} & \multicolumn{1}{c}{B} & \multicolumn{1}{c}{C} \\ \hline 3 & 0 & a_3 \\ \hline 3 & 0 & a_3 \\ 0 & b_1 & 2 \\ \hline \end{array}, \quad T_2 = \begin{array}{|ccc|} \multicolumn{1}{c}{A} & \multicolumn{1}{c}{B} & \multicolumn{1}{c}{C} \\ \hline a_1 & a_2 & a_3 \\ \hline a_1 & 0 & a_3 \\ 3 & a_2 & a_3 \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & a_3 \\ a_1 & b_2 & b_3 \\ \hline \end{array}, \quad T_3 = \begin{array}{|ccc|} \multicolumn{1}{c}{A} & \multicolumn{1}{c}{B} & \multicolumn{1}{c}{C} \\ \hline a_1 & 0 & a_3 \\ \hline a_1 & 0 & a_3 \\ 2 & b_4 & a_3 \\ a_1 & b_4 & a_3 \\ \hline \end{array},$$

$$T_4 = \begin{array}{|ccc|} \multicolumn{1}{c}{A} & \multicolumn{1}{c}{B} & \multicolumn{1}{c}{C} \\ \hline a_1 & a_2 & a_3 \\ \hline 3 & a_2 & a_3 \\ a_1 & 0 & a_3 \\ a_1 & a_2 & a_3 \\ \hline \end{array}, \quad T_5 = \begin{array}{|ccc|} \multicolumn{1}{c}{A} & \multicolumn{1}{c}{B} & \multicolumn{1}{c}{C} \\ \hline a_1 & 0 & a_3 \\ \hline a_1 & 0 & a_3 \\ 2 & b_5 & a_3 \\ a_1 & b_5 & a_3 \\ 2 & b_5 & b_6 \\ \hline \end{array}.$$

Conventionally, the row immediately below the summary is called the first row, the next row is the second, etc.

We can map each one of the first three rows of T_2 to an identical row in T_4. The last two rows of T_2 are mapped to the first and third rows of T_4, respectively. Thus $T_4 \subseteq_T T_2$. Similarly, we can map the first three rows of T_3 to the first three rows of T_5, respectively. Therefore $T_5 \subseteq_T T_3$. By Theorem 3, $Y_2 \subseteq_T Y_1$.

The first three rows of T_5 can be mapped to the first three rows of T_3, respectively, and now the last row of T_5 can be mapped to the second row of T_3. Therefore $T_3 \subseteq_T T_5$. Similarly, $T_2 \subseteq_T T_4$. All the rows of T_4 can be mapped to the first row of T_1, and hence $T_1 \subseteq_T T_4$. By Theorem 3, $Y_1 \subseteq_T Y_2$, and now Corollary 4 implies that $Y_1 \equiv Y_2$. □

5. *Minimizing Unions of Tableaux*

A union of tableaux $Y = \bigcup_{i=1}^{n} T_i$ is *nonredundant* if there are no T_i and T_j $(i \neq j)$ such that $T_i \subseteq_T T_j$. We can always transform Y to an equivalent nonredundant union of tableaux by finding T_i and T_j such that $T_i \subseteq_T T_j$, deleting T_i, and repeating this process until no tableau of Y is contained in another tableau of Y.

Example 5. Consider the union of tableaux $Y_1 = \bigcup_{i=1}^{3} T_i$ in Example 4. $T_1 \subseteq_T T_2$, and hence T_1 can be deleted from Y_1. T_2 is not contained in T_3, and T_3 is not contained in T_2 Thus Y_1 is equivalent to the nonredundant union of tableaux $Y_1' = \bigcup_{i=2}^{3} T_i$. Y_2 is already nonredundant. □

The following theorem proves that nonredundant unions of tableaux are unique up to equivalence of tableaux.

THEOREM 5. *Let $Y = \bigcup_{i=1}^{n} T_i$ and $Y' = \bigcup_{j=1}^{m} T_j'$ be nonredundant unions of tableaux. $Y \equiv Y'$ if and only if Y and Y' have the same target relation scheme, and for every T_i in Y there is a unique T_j' in Y' such that $T_i \equiv T_j'$ and vice versa (hence $m = n$).*

PROOF. *If.* This direction is easy by the definition of equivalence.

Only if. Suppose that $Y \equiv Y'$. Let T_i be a tableau of Y. By Corollary 4 there is a T_j' in Y' such that $T_i \subseteq_T T_j'$. If $T_j' \not\subseteq_T T_i$, then there is some T_k $(k \neq i)$ in Y such that $T_j' \subseteq_T T_k$ This implies that $T_i \subseteq_T T_k$ and contradicts the nonredundancy of Y. Therefore $T_i \equiv T_j'$

Since Y' is nonredundant, there can be at most one T'_j in Y' such that $T_i \equiv T'_j$. Similarly, we can show that for every T'_j in Y' there is a unique T_i in Y such that $T'_j \equiv T_i$. □

Example 6. Consider again the unions $Y'_1 = \bigcup_{i=2}^{3} T_i$ and $Y_2 = \bigcup_{i=4}^{5} T_i$ (see Examples 4 and 5). Y'_1 and Y_2 are nonredundant and equivalent, T_2 is equivalent to T_4, and T_3 is equivalent to T_5. □

Obtaining a nonredundant union of tableaux is only the first step. The next step is to minimize each tableau in the union. A tableau T can be minimized by mapping the rows of a tableau to itself and eliminating rows not in the image of the map [5, 9]. This process transforms a tableau T to an equivalent tableau that has a minimum number of rows. Since the number of rows corresponds to the number of joins whenever the tableau comes from an expression, minimizing tableaux is an important step in the optimization of restricted relational expressions [3, 5]. Minimal tableaux are unique up to renaming of variables [9].

Example 7. $Y'_1 = \bigcup_{i=2}^{3} T_i$ and $Y_2 = \bigcup_{i=4}^{5} T_i$ (Examples 4–6) are equivalent and nonredundant. T_3 and T_4 are already minimized. The minimized form of T_2 is T_4, and the minimized form of T_5 is T_3 (there is an efficient algorithm for minimizing tableaux such as these [5]). Thus the minimized form of both Y'_1 and Y_2 is $Y = \bigcup_{i=3}^{4} T_i$ (this is also the minimized form of Y_1). □

6. *Relational Tableau Expressions and Unions of Elementary Differences*

The result of Section 3 can be generalized to all relational expressions with the operators select, project, join, union, and difference, provided the project operator is applied only to subexpressions without the difference operator. In order to do so, we define a new type of expression, called *relational tableau expressions* (r.t.e.), that have tableaux rather than relation schemes as operands.

(1) Every tableau T is an r.t.e. For an instance I, $T(I)$ is defined as in Section 2

(2a) If E is an r.t.e., then so is $\sigma_{A=c}(E)$ (provided A is in the target relation scheme of E). The target relation scheme of $\sigma_{A=c}(E)$ is the same as that of E, and $(\sigma_{A=c}(E))(I) = \sigma_{A=c}(E(I))$, for all instances I.

(2b) If E is an r.t.e., then so is $\pi_X(E)$. The target relation scheme of $\pi_X(E)$ is X (it is assumed that X is contained in the target relation scheme of E). For all instances I, $(\pi_X(E))(I) = \pi_X(E(I))$.

(2c) If E_1 and E_2 are r.t.e.'s, then so is $E_1 \bowtie E_2$, and its target relation scheme is the union of the target relation schemes of E_1 and E_2. For all instances I, $(E_1 \bowtie E_2)(I) = (E_1(I)) \bowtie (E_2(I))$.

(2d) If E_1 and E_2 are r.t.e.'s with the same target relation scheme, then $E_1 \cup E_2$ is an r.t.e. with that target relation scheme. For all instances I, $(E_1 \cup E_2)(I) = (E_1(I)) \cup (E_2(I))$.

(2e) If E_1 and E_2 are r.t.e.'s with the same target relation scheme, then $E_1 - E_2$ is an r.t.e. with that target relation scheme. For all instances I, $(E_1 - E_2)(I) = (E_1(I)) - (E_2(I))$.

Optimization techniques for tableaux are described in [3, 5]. A similar treatment of r.t.e.'s requires that we apply the following equivalence preserving transformations to r.t.e.'s.

(1) $\sigma_{A=c}(E_1 \cup E_2) \equiv \sigma_{A=c}(E_1) \cup \sigma_{A=c}(E_2)$.
(2) $\sigma_{A=c}(E_1 - E_2) \equiv \sigma_{A=c}(E_1) - \sigma_{A=c}(E_2)$.
(3) $(E_1) \bowtie (E_2 \cup E_3) \equiv (E_1 \bowtie E_2) \cup (E_1 \bowtie E_3)$.
(4) $(E_1) \bowtie (E_2 - E_3) \equiv (E_1 \bowtie E_2) - (E_1 \bowtie E_3)$.
(5) $E_1 \bowtie E_2 \equiv E_2 \bowtie E_1$.
(6) $\pi_X(E_1 \cup E_2) \equiv \pi_X(E_1) \cup \pi_X(E_2)$.

The fact that these transformations are equivalence preserving is easily proved. The proof is similar to that of Theorem 1.

It follows from the above transformations that select, join, and project (when project is applied only to subexpressions without difference) can be pushed down to the level of single tableaux and applied to the tableaux. The resulting expressons have only union and difference as operators. All the operands of an r.t.e. with only union and difference have the same target relation scheme. The intersection operator, denoted by $\cap$, is another operator that can be applied only to operands having the same target relation scheme. (Note that intersection is a special case of join when the operands have the same target relation scheme.)

We therefore define a *tableau expression* (t.e.) to be an r.t.e. consisting only of tableaux as operands, and union, intersection, and difference as operators. For every relation scheme R, the set of t.e.'s that have R as their target relation scheme (modulo $\equiv$) with the ordering $\subseteq_T$ is a Boolean algebra. A formal treatment would require that we deal with equivalence classes of $\equiv$. We will use this fact, however, only to prove identities of t.e.'s by applying the axioms of Boolean algebra. Therefore, we only show that t.e.'s satisfy the axioms of Boolean algebra. Indeed $\varnothing$ (the tableau mapping every instance to the empty relation) is 0; and 1 is the tableau T such that for all attributes A in R, T has a row with a distinguished variable in the column for A and distinct nondistinguished variables in the rest of the columns. (Note that 1 is not the mapping that maps an instance I to its projection onto R, but rather the mapping that projects I onto each attribute of R and then joins all these projections.) It is easily shown that T contains every other tableau, and hence it contains every t.e.

The least upper bound of two t.e.'s E_1 and E_2 is $E_1 \cup E_2$. To prove, suppose that $E_1 \subseteq_T E$ and $E_2 \subseteq_T E$. Thus for all instances I, $E_1(I) \subseteq E(I)$ and $E_2(I) \subseteq E(I)$. This implies that $E_1(I) \cup E_2(I) \subseteq E(I)$, or by definition, $(E_1 \cup E_2)(I) \subseteq E(I)$ for all instances I. Hence $E_1 \cup E_2 \subseteq_T E$. Similarly, the greatest lower bound of E_1 and E_2 is $E_1 \cap E_2$. The complement of E, written $\bar{E}$, is $(1 - E)$, and now $E_1 - E_2$ can be written as $E_1 \cap \bar{E}_2$. By transformation (3), intersection is distributive over union, and, similarly, union is distributive over intersection. Thus the set of t.e.'s is indeed a Boolean algebra.

Let E_1 and E_2 be t e 's. The following are well-known identities of Boolean algebra.

(a) $\overline{E_1 \cup E_2} \equiv \bar{E}_1 \cap \bar{E}_2$.
(b) $\overline{E_1 \cap E_2} \equiv \bar{E}_1 \cup \bar{E}_2$.

By using these identities and the fact that intersection is distributive over union, we can transform any t.e. E to an equivalent t.e. in "disjunctive" normal form. Namely, E can be written as

$$D_1 \cup D_2 \cup \cdots \cup D_n,$$

where each D_i is of the form

$$T_1 \cap \cdots \cap T_j \cap \bar{T}_{j+1} \cap \cdots \cap \bar{T}_m.$$

That is, each D_i is the intersection of tableaux and complemented tableaux. Since the intersection operator is a special case of the join operator, $T_1 \cap \cdots \cap T_j$ is equivalent to a single tableau T. By rule (a) above, $\bar{T}_{j+1} \cap \cdots \cap \bar{T}_m$ is equivalent to $(1 - \bigcup_{k=j+1}^{m} T_k)$. Hence D_i is equivalent to

$$T \cap (1 - \bigcup_{k=j+1}^{m} T_k),$$

or equivalently,

$$T - \bigcup_{k=j+1}^{m} T_k.$$

A t.e. of the form $T - \bigcup_{i=1}^{n} T_i$ is called an *elementary difference* and is usually written as

$T - Y$, where Y is understood to be a union of tableaux. It follows that every t.e. can be written as a *union of elementary differences*, i.e., $\bigcup_{i=1}^{n} D_i$, where each D_i is an elementary difference. We assume that in a difference $T - Y$, $Y \subseteq_T T$; otherwise, we replace $T - Y$ with $T - (T \cap Y)$ (by Theorem 1, $T \cap Y$ is also a union of tableaux).

7. *Equivalence of Unions of Elementary Differences*

In this section we give a set of transformations that can be applied to unions of elementary differences to obtain equivalent forms. These transformations are finite Church–Rosser [6], and they can be used to formulate a necessary and sufficient condition for the equivalence of unions of elementary differences. Let $Z = \bigcup_{i=1}^{n} D_i$ be a union of elementary differences, and let $D_k = T_k - Y_k$ $(1 \le k \le n)$. Consider two distinct elementary differences of Z, D_i and D_j. The following transformations may be applied to Z.

(1) *Restriction of the right side*: If Y_i has a tableau T such that $T \subseteq_T T_j$ and $T \subseteq_T Y_j$, then replace T with $T \cap Y_j$.
(2) *Restriction of the left side*: If $T_i \subseteq_T T_j$ and $T_i \not\subseteq_T Y_j$, then replace $T_i - Y_i$ with $(T_i \cap Y_j) - Y_i$. If Y_i is not contained in $T_i \cap Y_j$, replace it with $Y_i' = Y_i \cap (T_i \cap Y_j) \equiv Y_i \cap Y_j$.
(3) *Elimination of null differences*: If $T_i \equiv Y_i$, eliminate $T_i - Y_i$ from Z.

LEMMA 6. *Transformations* (1)–(3) *preserve equivalence.*

PROOF. The lemma is proved by applying the axioms of Boolean algebra. Suppose that D_i is replaced with D_i' by transformation (1). The identity $D_i' \subseteq_T D_i \cup (T - (T \cap Y_j))$ follows from the axioms of Boolean algebra. But $T - (T \cap Y_j) \subseteq_T D_j$ (since $T \subseteq_T T_j$), and therefore $D_i' \subseteq_T Z$. Since $D_i \subseteq_T D_i'$, transformation (1) preserves equivalence.

Suppose now that D_i is replaced with D_i' by transformation (2). Obviously, $D_i' \subseteq_T D_i$. Conversely, $D_i \subseteq_T D_i' \cup D_j$, and hence transformation (2) preserves equivalence. As for transformation (3), it clearly preserves equivalence. □

Suppose that Z is a union of elementary differences with operands $T_1, \ldots, T_n$. If either transformation (1) or transformation (2) is applied to Z, then an operand T_i of Z is replaced with several new operands $T_1', \ldots, T_m'$, such that each T_k' is a join of T_i and one of the operands $T_1, \ldots, T_n$. Since we start with a union of elementary differences Z that has a finite number of operands, there is only a finite number of new operands that can be created by joining some of the original operands of Z. Hence transformations (1)–(3) can be applied to a union of elementary differences only a finite number of times. A union of elementary differences is *irreducible* if no transformation can be applied to Z.

LEMMA 7. *Let $Z = \bigcup_{i=1}^{n} D_i$ be a union of elementary differences, and let $D = T - Y$ be an elementary difference other than $\emptyset$.*

(1) *If $D \subseteq_T Z$, then there is a $D_i = T_i - Y_i$ such that $T \subseteq_T T_i$ and $T \not\subseteq_T Y_i$.*
(2) *If Z is irreducible, $D \subseteq_T Z$, and $T \equiv T_i$ (for some $D_i = T_i - Y_i$ in Z), then $D \subseteq_T D_i$.*

PROOF. (1) We use a magic instance I and a magic tuple μ for the tableau T, as described in Section 4. The tuple μ is in $T(I)$. If μ is also in $Y(I)$, then $T \subseteq_T Y$ and $T - Y$ is $\emptyset$. Since this is impossible, μ cannot be in $Y(I)$, and hence it is in $D(I)$. But $D \subseteq_T Z$ implies that μ is in $Z(I)$. Consequently, there is a $D_i = T_i - Y_i$ such that μ is in $D_i(I)$; i.e., μ is in $T_i(I)$ but not in $Y_i(I)$. It follows that $T \subseteq_T T_i$. Since $\mu \in T(I)$ and $\mu \notin Y_i(I)$, $T \subseteq_T Y_i$ is impossible. Thus $T \subseteq_T T_i$ and $T \not\subseteq_T Y_i$.

(2) Suppose that Z is irreducible and $T \equiv T_i$. If $D \not\subseteq_T D_i$, then there is a tableau T' in Y_i such that $T' - Y \neq \emptyset$ (otherwise it follows from the axioms of Boolean algebra that $D \subseteq_T D_i$). But $T' - Y \subseteq_T D$, and hence $T' - Y \subseteq_T Z$. By part (1) there is a D_j in Z such that $T' \subseteq_T T_j$ and $T' \not\subseteq_T Y_j$. Since $T' \subseteq_T Y_i$, it follows that $i \neq j$. But in this case transformation (1) can be applied to D_i and D_j in contradiction to the irreducibility of Z. Therefore $D \subseteq_T D_i$. □

LEMMA 8. *If Z is irreducible, then it does not contain elementary differences $D_i = T_i - Y_i$ and $D_j = T_j - Y_j$ ($i \neq j$) such that $T_i \equiv T_j$.*

PROOF. If $T_i \equiv T_j$, then $T_i \subseteq_T Y_j$, since transformation (2) cannot be applied. But this implies that $T_j \equiv Y_j$ and D_j is redundant. □

COROLLARY 9. *If Z is irreducible, then $D_i \not\equiv D_j$ if $i \neq j$.*

PROOF. By Lemma 7, $D_i \equiv D_j$ implies that $T_i \equiv T_j$. □

THEOREM 10. *Let $D_1 = T_1 - Y_1$ and $D_2 = T_2 - Y_2$ be elementary differences. $D_2 \subseteq_T D_1$ if and only if they have the same target relation scheme, $T_2 \subseteq_T T_1$, and $T_2 \cap Y_1 \subseteq_T Y_2$.*

PROOF. *If.* This part of the proof follows immediately from the axioms of Boolean algebra.

Only if. By Lemma 7, $T_2 \subseteq T_1$. Now we can use the axioms of Boolean algebra to show that $T_2 \cap Y_1 \subseteq Y_2$. □

COROLLARY 11. *Let $D_1 = (T_1 - Y_1)$ and $D_2 = (T_2 - Y_2)$ be elementary differences. $D_1 \equiv D_2$ if and only if they have the same target relation scheme, $T_1 \equiv T_2$, and $Y_1 \equiv Y_2$.*

PROOF. *If.* This is easy by the definition of equivalence.

Only if. By Theorem 10, $T_2 \subseteq_T T_1$ and $T_1 \subseteq_T T_2$. Thus $T_1 \equiv T_2$. Since $Y_1 \subseteq_T T_1$ and $T_1 \equiv T_2$, it follows that $T_2 \cap Y_1 \equiv Y_1$. Therefore, by Theorem 10, $Y_1 \subseteq_T Y_2$. Similarly, $Y_2 \subseteq_T Y_1$. Thus $Y_1 \equiv Y_2$. □

We now prove that two irreducible unions of elementary differences are equivalent if and only if they are the same (up to equivalence of elementary differences). Therefore transformations (1)–(3) are indeed finite Church–Rosser.

THEOREM 12. *Let $Z_1 = \bigcup_{i=1}^{n} D_i$ and $Z_2 = \bigcup_{j=1}^{m} F_j$ be irreducible unions of elementary differences. $Z_1 \equiv Z_2$ if and only if they have the same target relation scheme, and, for every D_i there is a unique F_j such that $D_i \equiv F_j$, and vice versa (hence $m = n$).*

PROOF. *If.* This direction is easy.

Only if. By Corollary 9, for each D_i there is at most one F_j such that $D_i \equiv F_j$. Let Z_1' be the result of removing from Z_1 every D_i for which there is an equivalent F_j in Z_2. Similarly, Z_2' is the result of removing from Z_2 every F_j for which there is an equivalent D_i in Z_1. It remains to be shown that both Z_1' and Z_2' are empty. In this proof we denote each D_i as $T_i - Y_i$ and each F_j as $U_j - X_j$.

Suppose that one of Z_1' and Z_2', say Z_1', is nonempty. Among all the elementary differences in Z_1', let D_i be an elementary difference such that T_i is maximal. That is, for all D_q ($q \neq i$) in Z_1', T_i is not contained in T_q. (Note that by Lemma 8, T_i exists.) By Lemma 7 there is an F_j in Z_2 such that $T_i \subseteq_T U_j$ and $T_i \not\subseteq X_j$. If F_j is not in Z_2', then there is a D_p in Z_1 such that $D_p \equiv F_j$. By Corollary 11, $T_p \equiv U_j$ and $Y_p \equiv X_j$. Therefore transformation (2) can be applied to Z_1, since $T_i \subseteq_T T_p$, $T_i \not\subseteq Y_p$, and obviously $i \neq p$. This contradiction implies that F_j is in Z_2'. Similarly, there is a D_k in Z_1' such that $U_j \subseteq_T T_k$. Since $T_i \subseteq_T U_j$, it follows that $T_i \subseteq_T T_k$, and the maximality of T_i implies that $i = k$. Therefore $T_i \equiv U_j$. By Lemma 7, $D_i \subseteq_T F_j$. Similarly, $F_j \subseteq_T D_i$, and so $D_i \equiv F_j$. This contradiction completes the proof of the theorem. □

8. *An NP-Completeness Result for Containment of Tableaux*

An efficient algorithm for testing containment of tableaux is necessary for testing equivalence of unions of tableaux or unions of elementary differences. However, efficient containment algorithms for tableaux are unlikely to exist in the general case, because the containment problem is NP-complete (cf. [2, 13]) even for simple tableaux[1] [4]. In this section we strengthen this result by showing that the containment problem is NP-complete even for expressions of the form $\pi_X(\Join_{i=1}^{n} R_i)$, where $R_1, R_2, \ldots, R_n$ are relation schemes and X is a subset of attributes. This result indicates that there is no large class of tableaux for which there is an efficient containment algorithm.

To show that the containment problem for expressions of the form $\pi_X(\Join_{i=1}^{n} R_i)$ is NP-complete, the exact cover problem (shown to be NP-complete in [16]) is reduced to this problem. Let $S_1, S_2, \ldots, S_q$ be the sets and $x_1, x_2, \ldots, x_n$ be the elements of an instance of the exact cover problem. We construct two tableaux as follows. The first tableau, T_1, has n rows that correspond to the elements $x_1, x_2, \ldots, x_n$, and $n + q$ columns as follows. The first n columns correspond to the elements $x_1, x_2, \ldots, x_n$, and the last q columns correspond to the sets $S_1, S_2, \ldots, S_q$. Row i of T_1 represents the element x_i and the sets in which x_i appears. This row contains the distinguished variable a_i in column i, and the nondistinguished variable b_j in column $n + j$ for every j ($1 \le j \le q$) such that x_i is a member of S_j. The rest of the columns have distinct nondistinguished variables. The summary of T_1 has the distinguished variable a_i in column i for $1 \le i \le n$, and blanks in the last q columns.

The second tableau, T_2, has q rows corresponding to the sets $S_1, S_2, \ldots, S_q$, and $n + q$ columns corresponding to the elements and sets exactly as in T_1. Row j represents the set S_j and the elements it contains. This row has the distinguished variable a_i in column i for every i ($1 \le i \le n$) such that x_i is a member of S_j. In the last q columns row j contains the nondistinguished variable d_k in column $n + k$ for $1 \le k \le q$ and $k \ne j$, and the nondistinguished variable e_j in column $n + j$ (note that e_j appears nowhere else). The rest of the columns have distinct nondistinguished variables. The summary of T_2 is identical to the summary of T_1.

Example 8. Let $S_1 = \{x_2, x_4\}$, $S_2 = \{x_2, x_3, x_4\}$, and $S_3 = \{x_1, x_2\}$ be an instance of the exact cover problem. The tableau T_1 for this instance is

	x_1	x_2	x_3	x_4	S_1	S_2	S_3
	a_1	a_2	a_3	a_4			
x_1	a_1	v_1	v_2	v_3	v_4	v_5	b_3
x_2	v_6	a_2	v_7	v_8	b_1	b_2	b_3
x_3	v_9	v_{10}	a_3	v_{11}	v_{12}	b_2	v_{13}
x_4	v_{14}	v_{15}	v_{16}	a_4	b_1	b_2	v_{17}

The tableau T_2 for this instance is

	x_1	x_2	x_3	x_4	S_1	S_2	S_3
	a_1	a_2	a_3	a_4			
S_1	v_{18}	a_2	v_{19}	a_4	e_1	d_2	d_3
S_2	v_{20}	a_2	a_3	a_4	d_1	e_2	d_3
S_3	a_1	a_2	v_{21}	v_{22}	d_1	d_2	e_3

□

LEMMA 13. *$T_2 \subseteq_T T_1$ if and only if the sets $S_1, S_2, \ldots, S_q$ have an exact cover.*

PROOF. *Only if* By Lemma 2 it suffices to show that if there is a containment mapping from the rows of T_1 to the rows of T_2, then the sets $S_1, S_2, \ldots, S_q$ have an exact cover. Under a containment mapping from T_1 to T_2, a row x_i of T_1 must be mapped to a row S_j of T_2 such that $x_i \in S_j$. If the row x_i is mapped to the row S_j, then the variable b_j is mapped to the variable e_j, and hence all the rows of T_1 that correspond to members of S_j have to be mapped to the row S_j of T_2. Suppose that the image of a containment mapping

[1] It is interesting to note that this class has a polynomial equivalence algorithm [4].

from T_1 to T_2 has two rows S_{j_1} and S_{j_2}, such that $S_{j_1} \cap S_{j_2} \neq \emptyset$. Let $x_i \in S_{j_1} \cap S_{j_2}$. Since some row of T_1 that corresponds to a member of S_{j_1} is mapped to row S_{j_1}, row x_i is also mapped to row S_{j_1}. A similar argument shows that row x_i is mapped to row S_{j_2}. This contradiction implies that the image of a containment mapping from T_1 to T_2 consists of rows that correspond to an exact cover of the sets $S_1, S_2, \ldots, S_q$.

If. Let $S_{j_1}, S_{j_2}, \ldots, S_{j_m}$ be an exact cover. We can map each row x_i of T_1 to the unique set S_{j_k} that contains x_i. This is a containment mapping from T_1 to T_2 (in which each b_j is mapped to e_j if S_j is in the exact cover, and to d_j otherwise). □

THEOREM 14. *Let E_1 and E_2 be relational expressions of the form $\pi_X(\bowtie_{i=1}^n R_i)$. Testing whether E_2 is contained in E_1 is NP-complete.*

PROOF. By Lemma 3 in [4] the tableaux T_1 and T_2 correspond to expressions E_1 and E_2, respectively. Furthermore, E_1 and E_2 are of the form $\pi_X(\bowtie_{i=1}^n R_i)$ and can be constructed in polynomial time from T_1 and T_2. Thus the theorem follows from Lemma 13. □

9. *Polynomial-Time Algorithms for Containment of Tableaux*

The result of Section 8 indicates that any wide class of tableaux for which there is an efficient containment algorithm is unlikely to exist. In this section we characterize three cases (albeit very special cases) for which there are polynomial containment algorithms.

9.1 OBTAINING CONTAINMENT ALGORITHMS FROM EQUIVALENCE ALGORITHMS. Suppose that T_1 and T_2 are tableaux with the same target relation scheme. By the axioms of Boolean algebra, $T_2 \subseteq_T T_1$ if and only if $T_1 \bowtie T_2 \equiv T_2$ (because in this case join is the same as intersection). Thus efficient containment algorithms can be obtained in this way from efficient equivalence algorithms. In particular, an $O(n^3)$-time equivalence algorithm for the class of simple tableaux is given in [4]. Therefore, it is possible to test whether $T_2 \subseteq_T T_1$ in $O(n^3)$ time, where n is the size of T_1 and T_2, if $T_1 \bowtie T_2$ (and hence also T_2) is a simple tableau.

9.2 ONE REPEATED NONDISTINGUISHED VARIABLE IN EACH ROW. A *repeated* symbol in a tableau is a symbol that appears in more than one row. Let T_1 and T_2 be tableaux with the same target relation scheme and with rows w and v, respectively. Row w is *covered* by row v if the following conditions are satisfied.

(1) If row w has a constant in some column A, then row v has the same constant in column A. Furthermore, if that constant appears in column A of the summary of T_1, then it also appears in column A of the summary of T_2.
(2) If row w has a distinguished variable in column A, then row v has a distinguished variable in column A, or a constant if that constant appears in column A of the summary of T_2.

Let W and V be sets of rows of T_1 and T_2, respectively. The set W is *covered* by the set V if each row of W is covered by some row of V. Note that if ψ is a mapping from the rows of T_1 to the rows of T_2 such that for all rows w of T_1, w is covered by $\psi(w)$, then ψ satisfies the first two conditions of a containment mapping.

THEOREM 15. *Let T_1 and T_2 be tableaux with the same target relation scheme, and suppose that each row of T_1 has at most one repeated nondistinguished variable. Then $T_2 \subseteq_T T_1$ if and only if*

(1) *for each nondistinguished variable b of T_1, all the rows containing b are covered by a set of rows of T_2 that have the same symbol in the column of b, and*
(2) *each row of T_1 without any repeated nondistinguished variable is covered by some row of T_2.*

PROOF. It is obvious that the conditions of a containment mapping imply the conditions of the theorem. Conversely, conditions (1) and (2) imply that each row of T_1 can be mapped to a row of T_2 that covers it, and moreover, if rows w and v of T_1 have the same nondistinguished variable in some column A, then they are mapped to rows of T_2 that agree in column A. Since each row has at most one repeated nondistinguished variable, this mapping is well defined. Thus $T_2 \subseteq_T T_1$. □

Example 9. Consider the tableaux T_2 and T_4 that appear in Example 4 Theorem 15 can be used to show that $T_4 \subseteq_T T_2$. The only repeated nondistinguished variable appearing in T_2 is b_2. The rows containing b_2 are covered either by the second row of T_4 or by the first and third rows of T_4. Each one of the other rows of T_2 is covered by an identical row of T_4 □

COROLLARY 16. *If T_1 has at most one repeated nondistinguished variable in each row, then it is possible to test whether $T_2 \subseteq_T T_1$ in $O(n^2)$ time, where n is the size of T_1 and T_2.*

PROOF. For each variable of T_1 or T_2 we can create in linear time a linked list that points to all the rows in which the variable appears. By using these lists, conditions (1) and (2) of Theorem 15 can be checked in $O(n^2)$ time. □

9.3 CHOOSING BETWEEN TWO POSSIBILITIES. Suppose that T_1 and T_2 are two tableaux such that every row of T_1 is covered by at most two rows of T_2. In this section we describe an algorithm for deciding whether $T_2 \subseteq_T T_1$. For each row w of T_1, let $t_1(w)$ and $t_2(w)$ be the two rows of T_2 that cover w (If only one row covers w, then $t_1(w) = t_2(w)$; if no row covers w, then $T_2 \not\subseteq_T T_1$.)

Let v be a row of T_1, and define $\theta(v) = t_1(v)$. We can test whether θ can be extended to a containment mapping from T_1 to T_2 as follows. We use a set R that contains all the rows of T_1 for which the value of θ has already been determined. Initially $R = \{v\}$. Let w be a row of R, and let s be another row of T_1 that has the same nondistinguished variable as w in some column A. If only one of $t_1(s)$ and $t_2(s)$, say $t_j(s)$, agrees with $\theta(w)$ in column A, then we say that s is *forced* to $t_j(s)$ by $\theta(w)$. If both $t_1(s)$ and $t_2(s)$ disagree with $\theta(w)$ in column A, then we say that s is *impossible* as a result of $\theta(w)$. Clearly, if s is impossible as a result of $\theta(w)$, then θ cannot be extended to a containment mapping from T_1 to T_2; if s is forced by $\theta(w)$ to some row v, then $\theta(s)$ must be v, and hence we add s to R. Suppose that at some point no row of T_1 for which θ has not been determined yet is forced or impossible as a result of $\theta(w)$ for some w in R. Then the following lemma can be proved.

LEMMA 17. *Let S be the set containing all the rows of T_1 that are not in R, and let ψ be a containment mapping from the rows of S to the rows of T_2. Define a mapping ξ as follows:*

$$\begin{aligned} \xi(w) &= \theta(w) \quad &\text{if} \quad w \in R, \\ \xi(w) &= \psi(w) \quad &\text{if} \quad w \notin R. \end{aligned}$$

Then ξ is a containment mapping from T_1 to T_2.

PROOF. Conditions (1) and (2) of a containment mapping are obviously satisfied by ξ. As for condition (3), let v and w be two rows of T_1 that have the same nondistinguished variable in some column A. If both v and w are in R or both are not in R, then $\xi(v)$ and $\xi(w)$ agree in column A, since ψ is a containment mapping on S and θ is a containment mapping on R. Suppose that one of v and w, say v, is a member of R, but w is not a member of R. Since w is not forced or impossible as a result of v being in R, both $t_1(w)$ and $t_2(w)$ agree with $\theta(v)$ in column A. But $\xi(w)$ is either $t_1(w)$ or $t_2(w)$, since ψ is a containment mapping, and hence $\xi(w)$ and $\xi(v)$ agree in column A. Thus condition (3) is also satisfied. □

We choose a row v of T_1, define $\theta_1(v) = t_1(v)$, $\theta_2(v) = t_2(v)$, and compute θ_1 and θ_2 for the sets R_1 and R_2, respectively, as described above. If the computation of θ_i produces a row which is impossible, then we say that θ_i *fails.* If both θ_1 and θ_2 fail, then there is no containment mapping from T_1 to T_2. However, if one of these computations, say the computation of θ_1, is the first to terminate without failing, then by Lemma 17 there is no need to complete the computation of θ_2 (if it has not failed yet). We can define $\theta(w) = \theta_1(w)$ for all rows w in R_1 and repeat this process for some row for which θ has not yet been defined. By Lemma 17, when the process is repeated, it suffices to consider only rows for which θ has not yet been defined. If θ can be defined in this way for all the rows of T_1, then Lemma 17 implies that θ is a containment mapping T_1 to T_2. However, if both θ_1 and θ_2 fail at some point, then $T_2 \not\subseteq_T T_1$.

Computing θ_1 and θ_2 requires a queue Q_i for each θ_i. When a row v is determined to be forced by some $\theta_i(w)$, row v is put on Q_i. For each row w on Q_i we find all the rows that are forced or impossible as a result of $\theta_i(w)$. In order to obtain an efficient algorithm, θ_1 and θ_2 are computed in parallel.[2] That is, we alternate between removing a row from Q_1 and removing a row from Q_2.

Algorithm

```
(1) Make $\theta$ undefined for all rows $w$ of $T_1$
(2) Make $\theta_1$ and $\theta_2$ undefined for all rows $w$ of $T_1$, make the queues $Q_1$ and $Q_2$ empty, and assign FALSE to
    FAIL(1) and FAIL(2).
(3) If $\theta$ is defined for all the rows of $T_1$, then $T_2 \subseteq_T T_1$, otherwise, arbitrarily choose a row $w$ for which $\theta$ is
    undefined, define $\theta_1(w) = t_1(w)$, $\theta_2(w) = t_2(w)$, and put $w$ on $Q_1$ and $Q_2$.
(4) $i = 1$.
(5) If $Q_i$ is empty, then define $\theta(w) = \theta_i(w)$ for all rows $w$ such that $\theta_i(w)$ is defined and go to (2).
(6) Find and delete $v$, the first element of $Q_i$.
(7) For all rows $s$ of $T_1$ such that $\theta(s)$ is undefined and $s$ and $v$ have the same nondistinguished variable in some
    column $A$ do
    (a) If $\theta_i(s)$ is defined and $\theta_i(s)$ and $\theta_i(v)$ disagree in column $A$, then FAIL($i$) = TRUE and go to (8)
    (b) If $\theta_i(s)$ is undefined and $s$ is forced by $\theta_i(v)$ to some row $u$, then define $\theta_i(s) = u$ and put $s$ at the end of
        $Q_i$; if $\theta_i(s)$ is undefined and $s$ is made impossible by $\theta_i(v)$, then FAIL($i$) = TRUE and go to (8)
(8) If both FAIL(1) and FAIL(2) are TRUE, then $T_2 \not\subseteq_T T_1$.
(9) If $i = 1$ and FAIL(2) = FALSE, then $i = 2$ and go to (5).
(10) If $i = 2$ and FAIL(1) = FALSE, then $i = 1$ and go to (5).
(11) Go to (5).
```

THEOREM 18. *It is possible to test whether $T_2 \subseteq_T T_2$ in $O(n^2)$ time, where n is the size of the input.*

PROOF. Suppose that T_1 and T_2 have m columns, and n_1 and n_2 rows, respectively. We first find the set of rows of T_2 that cover each row of T_1 and check that it contains at most two rows (if it is empty for some row of T_1, then $T_2 \not\subseteq_T T_1$). This can be done in $O(n_1n_2m)$ time. Now consider the time required to compute θ_1 and θ_2 for some arbitrary choice of a row w in step (3) of the algorithm. Executing steps (6)–(7) once (for some row v) requires no more than $O(n_1m)$ time. Let p be the smallest integer such that steps (6)–(7) are repeated no more than p times for each θ_i (for the w chosen in step (3)). Since a row is put on each of Q_1 and Q_2 at most once, p cannot be greater than the number of rows for which θ has not yet been defined (because only rows for which θ has not yet been defined are considered). Also note that if either θ_1 or θ_2 does not fail, then the number of rows for which θ has been defined is increased by p in step (5). It follows that steps (6)–(7) are not repeated more than $2n_1$ times during the execution of the algorithm. Thus the total time required to test whether $T_2 \not\subseteq_T T_1$ is $O(n_1^2m + n_1n_2m)$. □

9.4 ANOTHER NP-COMPLETENESS RESULT FOR CONTAINMENT OF TABLEAUX. If we try to expand the last two cases by considering T_1 and T_2 such that every row of T_1 is covered by no more than three rows of T_2 and contains no more than two repeated nondistinguished variables, then the containment problem becomes NP-complete even when T_1 and T_2 correspond to expressions of the form $\pi_X(\bowtie_{i=1}^{n} R_i)$.

[2] This idea of trying both possibilities in parallel has been used previously to develop linear time algorithms for several combinatorial problems [12, 14].

The proof of this result is a modification of the NP-completeness proof of Section 8. The exact cover problem is NP-complete even if each element appears in no more than three sets [16]. This version of the exact cover problem is reduced to the above containment problem in a way similar to the reduction of Section 8. The tableaux T_1 and T_2 of Section 8 have to be modified as follows. We add to each one of them n columns that correspond to the attributes $y_1, y_2, \ldots, y_n$ and replace the rows of T_1 with up to $3n$ new rows. In T_2 the new columns are filled with distinct nondistinguished variables. In T_1 there are up to three rows for each element x_i. Each one of these rows corresponds to one of the three sets that contain x_i. A row that corresponds to the element x_i and the set S_j containing this element has the distinguished variable a_i in the column corresponding to x_i, the nondistinguished variable b_j in the column corresponding to S_j, and the nondistinguished variable v_i in the column corresponding to y_i. All the other columns have distinct nondistinguished variables. When we consider a containment mapping from T_1 to T_2, the repeated nondistinguished variable v_i forces all the rows of T_1 that correspond to the element x_i to be mapped to the same row of T_2 (because T_2 has distinct nondistinguished variables in the columns that correspond to the y_i's). This implies that Lemma 13 holds also in this case; i.e., a containment mapping from T_1 to T_2 exists if and only if the sets $S_1, S_2, \ldots, S_q$ have an exact cover. Since the tableaux used in this reduction correspond to expressions of the form $\pi_X(\Join_{i=1}^{n} R_i)$, we get the desired result.

10. *Testing Equivalence of Monotonic Relational Expressions*

The results of Section 8 and [4] imply that testing equivalence or containment of monotonic relational expressions is NP-hard. In this section we characterize the complexity of these problems, namely, we show that they are complete in the class Π_2^P of the polynomial-time hierarchy [22]. A language L is in Π_2^P if its complement is in Σ_2^P, the class of languages which can be recognized by nondeterministic polynomial-time algorithms with oracle from NP. The classes Σ_2^P and Π_2^P contain NP and are contained in PSPACE. It is not known if any of these inclusions is proper; however $\Sigma_2^P = \Pi_2^P =$ NP if and only if NP is closed under complementation.

THEOREM 19. *Testing containment for monotonic relational expressions is Π_2^P-complete.*

PROOF. *Part* 1 (*the problem is in Π_2^P*). Let E_1 and E_2 be two monotonic relational expressions on the same target relation scheme. We describe a nondeterministic polynomial algorithm with oracle from NP that answers "yes" if and only if $E_1 \not\subseteq_T E_2$.[3]

Let $Y_1 = \bigcup_{i=1}^{n} S_i$ and $Y_2 = \bigcup_{i=1}^{m} T_i$ be the unions of tableaux that represent the expressions E_1 and E_2, respectively. (Note that n and m may be exponential in the size of E_1 and E_2.) By Theorem 3, $E_1 \not\subseteq_T E_2$ if and only if for some tableau of Y_1, say S_j, $S_j \not\subseteq_T E_2$. A tableau S_j that testifies to $E_1 \not\subseteq_T E_2$ can be guessed bottom up as follows.

(a) If E_1 is a single relation scheme R, then the tableau for E_1 is the tableau corresponding to R.
(b) If $E_1 = \sigma_{A=c}(E')$ or $E_1 = \pi_X(E')$, then we guess a tableau T' for E', and apply the corresponding selection or projection to T'.
(c) If $E_1 = E' \Join E''$, then we guess a tableau for E' and a tableau for E'' and join them.
(d) If $E_1 = E' \cup E''$, then we guess either a tableau for E' or a tableau for E''.

After guessing the tableau S_j, we ask the oracle whether $S_j \subseteq_T E_2$. If the oracle answers "no," then the algorithm answers "yes." It remains to show that $S_j \subseteq_T E_2$ can be decided in NP. By Theorem 3, $S_j \subseteq_T E_2$ if and only if there is a tableaux of Y_2, say T_k, such that $S_j \subseteq_T T_k$. Thus we can guess the appropriate tableau T_k as before, and then check in nondeterministic polynomial time that $S_j \subseteq_T T_k$ [4].

Part 2 (*the problem is Π_2^P-hard*). The following quantified satisfiability problem

[3] The relation $\subseteq_T$ is used here between expressions E_1 and E_2 to mean that for all instances I, $v_I(E_1) \subseteq v_I(E_2)$.

(Q3-SAT) is shown to be Π_2^P-complete in [22, 24]. Given a Boolean formula $F = B_1 \wedge B_2 \wedge \cdots \wedge B_p$ in conjunctive normal form with three literals in each clause (3-CNF) and a partition of the variables of F into two sets $X_1 = \{x_1, \ldots, x_r\}$, $X_2 = \{x_{r+1}, \ldots, x_n\}$, determine if for all assignments of truth values to the variables in X_1, the formula F is satisfiable, i.e., determine if $(\forall X_1)(\exists X_2)F(X_1, X_2) = 1$. (We write X_1 for $x_1, \ldots, x_r$, $\forall X_1$ for $\forall x_1 \cdots \forall x_r$, etc.)

The NP-completeness of the containment problem for expressions over select, project, and join can be shown using the following reduction from the satisfiability problem of Boolean formulas in 3-CNF [4]. Given a formula $F = B_1 \wedge \cdots \wedge B_p$ in 3-CNF with variables $x_1, \ldots, x_n$, two tableaux T_1 and T_2 are constructed, each with $p + n$ columns, as follows. Both T_1 and T_2 have a summary that consists of distinguished variables in the first p columns and blanks in the remaining n columns. Let $x_{i_1}, x_{i_2}, x_{i_3}$ be the variables that appear in B_i (either complemented or not). For each clause B_i, T_1 has a row consisting of the distinguished variable a_i in column i; the nondistinguished variables $x_{i_1}, x_{i_2}, x_{i_3}$ in columns $p + i_1$, $p + i_2$, and $p + i_3$, respectively; and distinct nondistinguished variables in the rest of the columns. Tableau T_2 has seven rows for each clause B_i: one row for each truth assignment for the variables of B_i that satisfies B_i. Thus, if $c_{i_1}, c_{i_2}, c_{i_3}$ is a truth assignment for $x_{i_1}, x_{i_2}, x_{i_3}$ that satisfies B_i, then T_2 has a row consisting of the distinguished variable a_i in column i; the constants $c_{i_1}, c_{i_2}, c_{i_3}$ in columns $p + i_1$, $p + i_2$ and $p + i_3$, respectively; and distinct nondistinguished variables in the rest of the columns. The following two facts are shown in [4].

(a) If ξ is an assignment that satisfies F, then there is a homomorphism ψ from the symbols of T_1 to those of T_2 such that $\psi(a_i) = a_i$ and $\psi(x_i) = \xi(x_i)$. (Hence $T_2 \subseteq_T T_1$.)

(b) If $T_2 \subseteq_T T_1$ and ψ is a homomorphism, then assigning $\psi(x_i)$ to each variable x_i satisfies F.

We are going to use these tableaux and their properties in our reduction of Q3-SAT to the containment problem for monotonic expressions.

Let F be a Boolean formula in 3-CNF, and let $X_1 = \{x_1, \ldots, x_r\}$, $X_2 = \{x_{r+1}, \ldots, x_n\}$ be a partition of the variables of F. We construct a tableau S_1 by adding r rows and r columns to T_1 as follows. Columns $p + n + 1$ through $p + n + r$ of S_1 have the distinguished variables $a'_1, \ldots, a'_r$, respectively, in the summary, and distinct nondistinguished variables in the rows of T_1. For $1 \le i \le r$ we add a row with x_i in column $p + i$, a'_i in column $p + n + i$, and distinct nondistinguished variables in the rest of the columns. Tableau S_1 is depicted in Figure 1. By Lemma 3 of [4], S_1 comes from an expression E_1 that can be constructed in polynomial time.

For $1 \le m \le r$ and $c = 0$ or 1, let H_{mc} be the expression whose tableaux has a single row with the constant c in column $p + m$, the distinguished variable a'_m in column $p + n + m$, and distinct nondistinguished variables in the rest of the columns. The summary of H_{mc} has a'_m in column $p + n + m$ and blanks everywhere else. Let D_2 be an expression whose tableau is identical to T_2 in the first $p + n$ columns and has distinct nondistinguished variables in the rest of the columns. By Lemma 3 of [4], such an expression exists and moreover can be constructed in polynomial time.

Let $E_2 = D_2 \bowtie (H_{11} \cup H_{10}) \bowtie (H_{21} \cup H_{20}) \bowtie \cdots \bowtie (H_{r1} \cup H_{r0})$, and let Y_2 be the corresponding union of tableaux. For every sequence $\alpha = \langle c_1, c_2, \ldots, c_r \rangle$ of length r of 0's and 1's, Y_2 contains a tableaux T_α as in Figure 2. Tableau T_α has the same summary as S_1. The first $7p$ rows of T_α are identical to the rows of T_2 in the first $p + n$ columns and have distinct nondistinguished variables in the last r columns. The last r rows of T_α are obtained from the last r rows of S_1 by replacing each x_i ($1 \le i \le r$) with c_i. It follows that $Y_2 = \cup_\alpha T_\alpha$. We will show that $E_2 \subseteq_T E_1$ if and only if $(\forall X_1)(\exists X_2)F(X_1, X_2) = 1$.

If. Suppose that $(\forall X_1)(\exists X_2)F(X_1, X_2) = 1$. Let T_α be a tableaux of Y_2 that corresponds to the sequence $\alpha = \langle c_1, \ldots, c_r \rangle$. There is a truth assignment ξ for the variables of F that assigns c_i to x_i ($1 \le i \le r$) and satisfies F. Let ψ be the mapping from the symbols of S_1 to those of T_α defined by

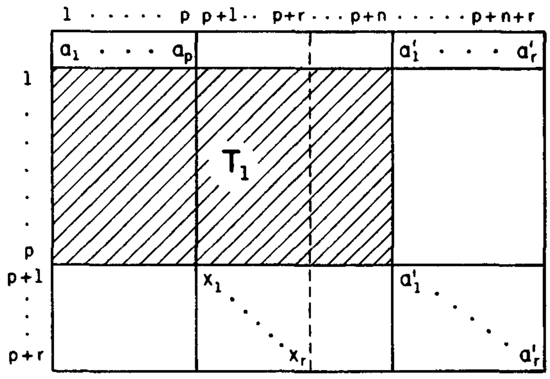

FIG 1 The tableau S_1

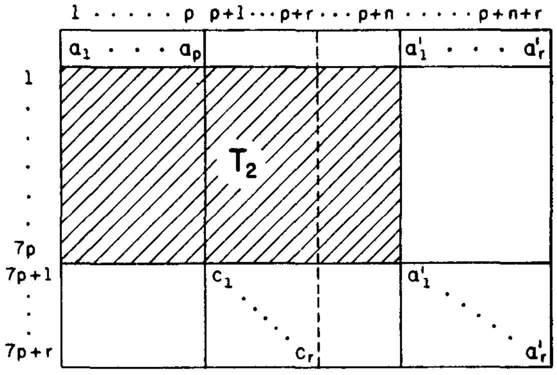

FIG 2 The tableau T_α for the sequence $\langle c_1, \quad , c_r\rangle$

(1) $\psi(a_i) = a_i$,
(2) $\psi(a_i') = a_i'$,
(3) $\psi(x_i) = c_i$ for $1 \leq i \leq r$,
(4) $\psi(x_i) = \xi(x_i)$ for $r + 1 \leq i \leq n$, and
(5) ψ is extended in the obvious way to the distinct nondistinguished variables of S_1.

The mapping ψ maps row $p + j$ of S_1 to row $7p + j$ of T_α. By property (a) of T_1 and T_2, row j (for $j \leq p$) is mapped to one of the first $7p$ rows of T_α. Since this is true for every α, $Y_2 \subseteq_T S_1$, and therefore $E_2 \subseteq_T E_1$.

Only if. Suppose that $E_2 \subseteq_T E_1$. By Theorem 3, $T_\alpha \subseteq_T S_1$ for every sequence α. Let ξ_1 be a truth assignment for X_1, and let $\alpha = \langle c_1, \ldots, c_r\rangle$ be the corresponding sequence, i.e., c_i is the truth value of x_i under ξ_1. Since $T_\alpha \subseteq_T S_1$, there is a homomorphism ψ from the symbols of S_1 to those of T_α. Since $\psi(a_i')$ must be a distinguished variable, row $p + i$ of S_1 must be mapped to row $7p + i$ of T_α, and consequently $\psi(x_i) = c_i$ for $1 \leq i \leq r$. Since each of the first p rows of S_1 has a distinguished variable in one of the first p columns, it has to be mapped to one of the first $7p$ rows of T_α. Therefore ψ gives a homomorphism from the symbols of T_1 to those of T_2, and by property (b) of T_1 and T_2, ψ gives a truth assignment that satisfies F and extends ξ_1. Consequently $(\forall X_1)(\exists X_2)F(X_1, X_2) = 1$. □

COROLLARY 20. *Testing equivalence of monotonic relational expressions is Π_2^P-complete.*

PROOF. The problem is in Π_2^P, since $E_1 \equiv E_2$ if and only if $E_1 \subseteq_T E_2$ and $E_2 \subseteq_T E_1$. The problem is Π_2^P-hard, since $E_1 \subseteq_T E_2$ if and only if $E_1 \bowtie E_2 \equiv E_1$. □

11. *Conclusions*

We have developed an exponential time algorithm for testing equivalence of monotonic relational expressions We have also introduced unions of elementary differences as nonprocedural representations of queries. We feel that many practical queries over select, project, join, union, and difference are naturally represented as unions of elementary differences, and we have isolated important special cases for which testing equivalence requires only polynomial time. We have not discussed the exact time complexity of testing equivalence of unions of elementary differences. However, it follows from the results of [20, 25] that this problem is also Π_2^P-complete.

We have not considered the problem of query optimization. In particular, it is not clear to what extent the unique minimization of unions of tableaux or the transformations of Section 7 are useful tools for query optimization. This problem merits further research. However, note that in many cases the transformations of Section 7 improve queries in ways similar to the optimization techniques of many papers in this area (e.g., [15, 21]). For example, the transformations produce queries in which select is applied as early as possible, and they transform a union of elementary differences to a single tableau whenever it is equivalent to a single tableau. (Thus they have the effect of removing redundant subexpressions.)

ACKNOWLEDGMENTS. The authors are grateful to C. Beeri and J. D. Ullman for valuable suggestions

REFERENCES

1 Aho, A V , Beeri, C , and Ullman, J D The theory of joins in relational databases *ACM Trans Database Syst 4*, 3 (Sept 1979), 297-314
2 Aho, A V , Hopcroft, J.E , and Ullman, J D *The Design and Analysis of Computer Algorithms* Addison-Wesley, Reading, Mass , 1974
3 Aho, A V , Sagiv, Y , Szymanski, T G , and Ullman, J D Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions Proc 16th Ann Allerton Conf on Communication, Control and Computing, Monticello, Ill , 85Oct 1978, pp 54-63
4. Aho, A V , Sagiv, Y , and Ullman, J D Equivalences among relational expressions *SIAM J Comput 8*, 2 (1979), 218-246
5. Aho, A V , Sagiv, Y , and Ullman, J D Efficient optimization of a class of relational expressions *ACM Trans Database Syst 4*, 4 (Dec 1979), 435-454
6 Aho, A V , Sethi, R , and Ullman, J.D Code optimization and finite Church-Rosser systems In *Design and Optimization of Compilers*, R Rustin, Ed , Prentice Hall, Englewood Cliffs, N J , 1972, pp 89-105
7 Armstrong, W W Dependency structures of data base relationship *Proc IFIP 74*, North Holland, New York, 1974, pp 580-583
8 Beeri, C, Fagin, R, and Howard, J H A complete axiomatization for functional and multivalued dependencies. Proc ACM-SIGMOD Intern. Conf on the Management of Data, Toronto, Ontario, Canada, August 1977, pp 47-61.
9 Chandra, A K , and Merlin, P M Optimal implementation of conjunctive queries in relational data bases Proc 9th Ann ACM Symp on Theory of Computing, Boulder, Colo , May 1977, pp 77-90
10 Codd, E F A relational model of data for large shared data banks, *Commun ACM 13*, 6 (June 1970), 377-387
11 Codd, E F Relational completeness of data base sublanguages. In *Data Base Systems*, R Rustin, Ed , Prentice Hall, Englewood Cliffs, N J , 1972, pp 65-98
12 Even, S , Itai, A , and Shamir, A On the complexity of timetable and multicommodity flow problems *SIAM J Comput 5*, 4 (1976), pp 691-703
13 Garey, M R , and Johnson, D S *Computers and Intractability A Guide to the Theory of NP-Completeness* Freeman, San Francisco, 1978
14 Gavril, F Testing for equality between maximum matching and minimum node covering, *Inform Proc Lett 6*, 6 (1977), 199-202
15 Hall, P A V Optimization of a single relational expression in a relational database system *IBM J Res Dev 20*, 3 (1976), 244-257
16 Karp, R M Reducibility among combinatorial problems In *Complexity of Computer Computations*, R E Miller and J W Thatcher, Eds , Plenum Press, New York, 1972, pp 85-103
17 Minker, J Performing inferences over relational databases Proc. ACM-SIGMOD Intern Conf on the Management of Data, San Jose, Calif, May 1975, pp 79-91
18 Palermo, F P A database search problem In *Information Systems COINS IV*, J T Tou, Ed , Plenum Press, New York, 1974

19. PECHERER, R M Efficient evaluation of expressions in a relational algebra Proc. ACM Pacific Conf, San Francisco, Calif, April 1975, pp 44-49
20 SAGIV, Y Optimization of queries in relational databases Ph D Thesis, Dept of Electrical Engineering and Computer Science, Princeton University, Princeton, N J, August 1978
21 SMITH, J M, AND CHANG, P Y -T Optimizing the performance of a relational algebra database interface *Commun ACM 18*, 10 (Oct 1975), 568-579
22 STOCKMEYER, L J The polynomial-time hierarchy *Theor Comput Sci 3*, 1 (1976), 1-22
23 WONG, E, AND YOUSSEFI, K Decomposition—a strategy for query processing *ACM Trans Database Syst 1*, 3 (Sept 1976), 223-241
24 WRATHALL, C Complete sets and the polynomial-time hierarchy *Theor Comp Sci 3*, 1 (1976), 23-33
25 YANNAKAKIS, M Unpublished manuscript

RECEIVED SEPTEMBER 1978, REVISED OCTOBER 1979, ACCEPTED OCTOBER 1979