Efficient Optimization of Simple Chase Join Expressions

PAOLO ATZENI
IASI-CNR
and
EDWARD P. F. CHAN
University of Toronto

Simple chase join expressions are relational algebra expressions, involving only projection and join operators, defined on the basis of the functional dependencies associated with the database scheme. They are meaningful in the weak instance model, because for certain classes of schemes, including independent schemes, the total projections of the repesentative instance can be computed by means of unions of simple chase join expressions. We show how unions of simple chase join expressions can be optimized efficiently, without constructing and chasing the corresponding tableaux. We also present efficient algorithms for testing containment and equivalence, and for optimizing individual simple chase join expressions.

Categories and Subject Descriptors: H.2.3 [**Database Management**]: Languages—*relational algebra*; H.2.4 [**Database Management**]: Systems—*query processing*

General Terms: Algorithms, Languages, Theory

Additional Key Words and Phrases: Chase, functional dependency, query equivalence, query optimization, relational database, tableau

1. INTRODUCTION

The *weak instance model* is an approach to relational databases that allows us to consider in a single framework databases composed of more than one relation. The motivation for its study lies in the importance of the notion of *decomposition*; that is, the replacement of a relation with two or more new relations, during the design or restructuring of a database.

The weak instance model was originally introduced in order to define the notion of global satisfaction of a set of dependencies [14], then used as a basis

The work by P. Atzeni was supported by Consiglio Nazionale delle Richerche, Italy while visiting the Department of Computer Science at the University of Toronto. The work of E. P. F. Chan was supported by the National Sciences and Engineering Research Council of Canada.
Authors' current addresses: P. Atzeni, IASI-CNR, Viale Manzoni 30, 00185 Rome, Italy; E. P. F. Chan, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1.

for query answering [22, 23, 29, 30], and to study equivalence [21] and desirable properties of database schemes [7, 8, 9].

With respect to query answering, it allows the formulation of queries on the original relation, providing the possibility of producing answers from data in the decomposed database. This idea has been carried further by some authors, who conceived a user interface, called *universal relation interface*, which presents the database as if it were composed of a single relation, thus relieving the user from specifying access paths and connections among the actual relations (see, for example, [19, 28] for surveys, and [4, 16, 27] for discussions).

Queries are posed on a relation defined on all the attributes in the various relations, but not actually stored in the database. The connection between the original (or universal) relation and the actual relations is provided by the *representative instance*, a relation over the universe U of the attributes, defined, for each database state $\mathbf{r}$, as follows. First, a relation over U (called the *state tableau* for $\mathbf{r}$, denoted by $T_{\mathbf{r}}$) is formed by taking the union of all the relations in $\mathbf{r}$ extended to U by means of distinct variables (or nulls). Then, the chase procedure [18] is applied to $T_{\mathbf{r}}$ to equate variables and generate new tuples. The chase process essentially performs inferences on data using the given constraints. If a contradiction is found during the chase process, then the representative instance is assumed to be empty. Any given query Q involves a set of attributes X that is a subset of the universe U; at the same time, each tuple in the representative instance contains constants only on a given subset of the universe U. Therefore, it is meaningful to consider, as the answer to such a query Q, the set of tuples in the representative instance that have only constants as values for the attributes in X. This set of tuples is called the X-*total projection* of the representative instance. It was shown that the X-total projection corresponds to the set of sentences that is logically implied by the database state and the constraints [20]. In this sense, the answer generated by this method is correct.

Example 1. Consider the database scheme

$$\mathbf{R} = \{R_1(\mathit{Course, Tutor, Instructor, Department}),\ R_2(\mathit{Course, Tutor, Room})\}$$
$$F = \{\mathit{Course} \rightarrow \mathit{Instructor},\ \mathit{Course} \rightarrow \mathit{Department}\}$$

and the database state $\mathbf{r}$:

r_1

Course	Tutor	Instructor	Department
CSC101	White	Smith	CS
ELE301	Red	Jones	EE

r_2

Course	Tutor	Room
CSC101	Green	A227
CSC201	Black	A325

The state tableau $T_{\mathbf{r}}$ for state $\mathbf{r}$ is the following:

Course	Tutor	Instructor	Department	Room
CSC101	White	Smith	CS	v_1
ELE301	Red	Jones	EE	v_2
CSC101	Green	v_3	v_4	A227
CSC201	Black	v_5	v_6	A325

The representative instance for **r** is obtained by chasing $T_{\mathbf{r}}$: since the first and third rows in $T_{\mathbf{r}}$ agree on the *Course*-component, and the functional dependency *Course* $\rightarrow$ *Instructor* is defined, the *Instructor*-component of the third row is set to "Smith"; similarly, its *Department*-component is set to "CS", because of the dependency *Course* $\rightarrow$ *Department*. Since no contradiction is found and no other inferences can be made from the state and the constraints, the corresponding representative instance is as follows:

Course	Tutor	Instructor	Department	Room
CSC101	White	Smith	CS	v_1
ELE301	Red	Jones	EE	v_2
CSC101	Green	Smith	CS	A227
CSC201	Black	v_5	v_6	A325

Suppose we want to know who are the instructor and the tutors of a course: the total projection on *Course Instructor Tutor* is required. It is obtained by projecting the representative instance on the attributes *Course Instructor Tutor*, and then removing the rows that contain variables:

Course	Tutor	Instructor
CSC101	White	Smith
ELE301	Red	Jones
CSC101	Green	Smith

The most straightforward way of finding an X-total projection is to follow the definitions, as we have just done in the example: first build the state tableau, then chase it to obtain the representative instance, and finally perform the total projection on the representative instance. Unfortunately, in this way, the whole database is involved in the answer to any query; in the easy cases (functional dependencies together with a full acyclic join dependency), time and space proportional to the size of the database are needed; in general, the upper bounds are exponential. Therefore more efficient strategies have been looked for.

Sagiv [22, 23] showed that, for a restricted class of database schemes, for any set of attributes $X \subseteq U$, there is a relational algebra expression E that can be used to compute the X-total projection; E is independent of the database state, and can be generated in time polynomial in the size of the database scheme. Clearly this strategy guarantees an enormous improvement over the previous one, since the size of the database scheme is much smaller than the size of the database state.

More recently, various authors [3, 15, 24] have extended this result to a larger class: the schemes that are *independent* with respect to functional dependencies [12]. Moreover, it was shown [6, 19] that, for this class, the expression E is always the union of *simple chase join expressions* (*scje*'s),[1] which are relational algebra expressions of a restricted form, involving only projections and joins, and defined on the basis of the functional dependencies associated with the database scheme. For instance, in Example 1, the total projection on *Course Instructor*

[1] Maier et al. [19] use the term *fd-join*.

Tutor can be computed by means of the following expression, which is a union of simple chase join expressions:

$$\pi_{Course,Instructor,Tutor}(R_1) \cup \pi_{Course,Instructor,Tutor}(R_2 \bowtie \pi_{Course,Instructor}(R_1)).$$

The subject of this paper is the optimization of expressions of this class; we say that a union of projection-join expressions E is *optimal* [26] if there is no other union of projection-join expressions that is equivalent to E and involves a smaller number of joins. It is known [26] that a union of projection-join expressions is optimal if and only if (i) there is no redundant term in the union, and (ii) each term is optimal. So, since scje's are projection-join expressions, our problem can be reduced to the problems of testing containment and optimizing individual scje's. Efficient solutions for these problems for more general projection-join operations already exist in the literature [1, 2, 25]; however, they require the construction and chase of the tableaux corresponding to the expressions, an operation that takes time $O(S^2 \log S)$, where S is the size of the input, that is, the size of the tableau plus the space needed to represent the fd's [10]. We show that, for the restricted class of scje's, it is not needed to consider the tableaux explicitly, since their structure can be predicted on the basis of the dependencies; more efficient algorithms are therefore devised.

The paper is organized as follows. Section 2 reviews the relevant definitions. In Section 3, we study containment and equivalence of scje's (which allow the elimination of redundant terms in the unions). In Section 4, we study the optimization of scje's. In Section 5, we discuss the applications of our results.

2. DEFINITIONS AND NOTATION

2.1 Basics

The *universe* $U = A_1A_2 \cdots A_m$ is a finite set of *attributes*. *A relation scheme* R is a subset of U.[2] A *database scheme* $\mathbf{R} = \{R_1, \ldots, R_n\}$ is a collection of relation schemes, such that the union of the R_i's is U.

Associated with each attribute $A \in U$ there is a set of *constants*, called the *domain* of A and indicated with $dom(A)$. A *tuple* on a set of attributes X is a function t that maps each attribute $A \in X$ to a constant in $dom(A)$; the notation $t[A]$ is used for the value of t on an attribute $A \in X$. If t is a tuple on X, and Y is a subset of X, then $t[Y]$ denotes the restriction of the mapping t to Y, and is therefore a tuple on Y. A *relation* on R is a set of tuples on R. A *state* (or a *database*) of a database scheme $\mathbf{R}$ is a function $\mathbf{r}$ that maps each relation scheme $R_i \in \mathbf{R}$ to a relation on R_i; with a slight abuse of notation, given $\mathbf{R} = \{R_1, \ldots, R_n\}$, we write $\mathbf{r} = \{r_1, \ldots, r_n\}$.

We shall consider relational expressions in which the only operators are *project* (π), (*natural*) *join* ($\bowtie$), and *union* ($\cup$). If only the operators project and join are involved, the expressions are called *PJ-expressions*. In the subsequent discussion, the operands of a relational expression are relation schemes in a database scheme.

[2] In the weak instance model it is usually required that no pair of relations are defined on the same set of attributes. As a consequence, it is possible to omit *relation names*, and identify the various relations by means of the involved sets of attributes.

2.2 Tableaux

A *tableau* consists of a *body* and a possibly empty *summary row*. The body of a tableau is composed of *rows*, which are defined as tuples on the universe augmented with a special attribute *TAG* (called the *tag* of the row), but on different domains. The definition of row requires a number of preliminary concepts. We fix a finite set $V_D = \{a_1, a_2, \ldots, a_n\}$ of *distinguished variables* {*dv*'s}, in one-to-one correspondence with the universe, and a countable set $V_N = \{v_1, v_2, \ldots\}$ of *nondistinguished variables* (*ndv*'s). The two sets V_D and V_N are disjoint from one another and disjoint from all the domains of the attributes; for each $1 \le i \le n$, the dv a_i, associated with A_i, is called the *distinguished variable for* A_i. Then, for each attribute $A_i \in U$ we define the *tableau domain* of A_i (indicated with $tdom(A_i)$) as the disjoint union of three sets: (i) $dom(A_i)$, (ii) the singleton set $\{a_i\}$, and (iii) V_N. The domain of the additional attribute *TAG* is the set of relation schemes. Finally, a *row* is a function that maps each element A of $U' = U \cup \{TAG\}$ to a value in $tdom(A)$; the summary row of a tableau is a function from a subset X of U, the *target relation scheme*, that maps each $A_i \in X$ to the dv $\{a_i\}$; an attribute A_i belongs to the target relation scheme if and only if the dv a_i appears in at least one of the rows of the body. When considering a given tableau, an ndv is called *unique* if it appears only once in the tableau, *repeated* otherwise. A symbol is called *significant* if it is not a unique ndv.

Tableaux are used in various ways in relational theory; we need two of them in this paper:

(1) Given a database state $\mathbf{r} = \{r_1, \ldots, r_k\}$, the *state tableau for* $\mathbf{r}$ is a tableau T_r, with an empty summary row, containing, for each relation $r_i \in \mathbf{r}$, and for each tuple $t \in r_i$, a row u with the same constants as t. Row u is defined as follows:

  —$u[TAG] = R_i$,
  —$u[A] = t[A]$, if $A \in R_i$,
  —$u[A]$ is a unique ndv, if $A \in U - R_i$.

  An example of state tableau was shown in the Introduction.

(2) Tableaux can be associated with expressions of a subset of relational algebra [1, 2], and used to study their properties, such as containment, equivalence, or optimization. The method for the construction of a tableau for a generic expression of this class is out of the scope of this paper; we will show in Section 2.5 the tableaux for the restricted class of expressions of interest in this paper, the scje's.

It is useful to define a partial order $\le$ on the elements of the tableau domains, as follows:

—Tags are pairwise incomparable and incomparable with all other elements. Constants are pairwise incomparable. Dv's are pairwise incomparable.
—If c is a constant and v a variable (either dv or ndv), then $c \le v$.

—If a_j is a dv and v a ndv, then $a_j \leq v$.
—The countable set V_N is totally ordered according to $\leq$.[3]

Let T be a tableau and t a row of T and X a subset of U. We say $t[X]$ is *total* if $t[A]$ is a constant, for all $A \in X$. The *total projection* (indicated with $\pi^{\downarrow}$) is an operator on tableaux that produces relations: $\pi^{\downarrow}{}_X(T) = \{t[X] \mid t \in T \text{ and } t[X] \text{ is total}\}$.

2.3 The Chase Procedure and the Representative Instance

The kinds of constraints considered in this paper are *functional dependencies* (*fd*'s). Given a set of fd's, there are additional dependencies implied by this set. The set of dependencies that are logically implied by F is the *closure of* F, denoted by F^+. Given a set of attributes X, the *closure* of X with respect to F, denoted by X^+, is the set of attributes $\{A \mid X \rightarrow A \in F^+\}$. An fd $X \rightarrow Y$ is *embedded* in a relation scheme R if $XY \subseteq R$.

The chase [18] is a procedure that, given a tableau T and a set of dependencies (in our case fd's), F, transforms, if possible, T into another tableau, denoted by $CHASE_F(T)$, whose body satisfies F. It involves the exhaustive application of *transformations*. A transformation can be applied to a tableau if there are an fd $X \rightarrow Y$ in F and two rows, l_s, l_t, in the body of the tableau, such that $l_s[X] = l_t[X]$ and $l_s[Y] \neq l_t[Y]$: for every attribute $A \in Y$ such that $l_s[A] \neq l_t[A]$, if $l_s[A]$ and $l_t[A]$ are comparable (with respect to the partial order $\leq$), then all the occurrences of the higher entry are replaced with the other entry, otherwise the tableau is replaced with the empty tableau, and it is said to *contradict* the set F of dependencies.

Given a set of dependencies F, the *representative instance* for a database state $\mathbf{r}$ is the tableau $CHASE_F(T_{\mathbf{r}})$ obtained by chasing, with respect to F, the state tableau for $T_{\mathbf{r}}$. The representative instance is important in this framework for two reasons:

—The notion of consistency of states is based on the representative instance: a state $\mathbf{r}$ is said to be (*globally*) *consistent* [14] if its representative instance is produced without encountering contradictions.
—The answer to a query involving a set of attributes $X \subseteq U$ is defined as the X-total projection of the representative instance [22, 23].

We have shown in the Introduction the representative instance for a simple database state, produced by means of the chase procedure, and the answer to a query, obtained by executing a total projection on it.

2.4 Containment Mappings and Containment and Optimality of Expressions

A *valuation function* is a function ν from $D' = \cup_{A \in U'} tdom(A)$ to itself, with the condition that $\nu(c) \leq c$, for each $c \in D'$. (Therefore, ν is the identity mapping on constants and tags.) A valuation function ν can be extended to rows and tableaux: if t is a row in the body of a tableau, then $\nu(t)$ is a row such that, for every $A \in$

[3] We will assume that the order coincides with the natural order of subscripts: $v_i \leq v_j$ if and only if $i \leq j$.

U', $\nu(t)[A] = \nu(t[A])$; similarly, if s is a summary row with target relation scheme X, then $\nu(s)$ is defined on X, with $\nu(s)[A] = \nu(s[A])$, for every $A \in X$; finally, given a tableau T, with summary row s and body $\{t_1, t_2, \ldots, t_n\}$, then $\nu(T)$ is a tableau with summary $\nu(s)$, and body $\{\nu(t_1), \nu(t_2), \ldots, \nu(t_n)\}$.

A tableau T_1 is said to *contain* another tableau T_2 (written $T_1 \supseteq T_2$) if there is a valuation function ν (called *containment mapping* from T_1 to T_2) such that the summaries of $\nu(T_1)$ and T_2 are the same and the body of $\nu(T_1)$ is contained in the body of T_2. T_1 is *equivalent* to T_2 ($T_1 \equiv T_2$), if $T_1 \supseteq T_2$ and $T_2 \supseteq T_1$.

Let E be a relational expression with operands in $\mathbf{R} = \{R_1, \ldots, R_k\}$. Then $E(\mathbf{r})$ denotes the value returned by E if a database state $\mathbf{r} = \{r_1, \ldots, r_k\}$ on $\mathbf{R}$ is substituted into the corresponding relation variables in E and is evaluated according to the usual definitions of the various operators. Let E_1 and E_2 be two relational expressions with operands defined on $\mathbf{R}$, with an associated set of dependencies F. E_1 is said to *contain* E_2, denoted by $E_1 \supseteq E_2$, if, for every state $\mathbf{r}$ of $\mathbf{R}$ consistent with respect to F, it is the case that $E_1(\mathbf{r}) \supseteq E_2(\mathbf{r})$. E_1 is said to be *equivalent* to E_2, denoted by $E_1 \equiv E_2$, if $E_1 \supseteq E_2$ and $E_2 \supseteq E_1$. A union of PJ-expressions E is *optimal* (or *minimal*) if there does not exist a union of PJ-expressions equivalent to E, with a fewer number of join operations.

2.5 Derivation Sequences and Simple Chase Join Expressions

Given a set of fd's F, a *derivation sequence* (*ds*) of some relation scheme R_{i_0} is a finite sequence of fd's $\tau = \langle f_1\colon Y_1 \to Z_1, \ldots, f_m\colon Y_m \to Z_m \rangle$ from F^+ such that, for all $1 \leq j \leq m$:

(1) f_j is embedded in some relation scheme R_{i_j}, with $i_j \neq i_0$, and
(2) $Y_j \subseteq R_{i_0}Z_1 \cdots Z_{j-1}$ and $Z_j \cap R_{i_0}Z_1 \cdots Z_{j-1} \neq \varnothing$.

We say that τ *covers* a set of attributes X if $R_{i_0}Z_1 \cdots Z_j \supseteq X$. Essentially, a ds of R_{i_0} is a sequence of embedded fd's used in computing (part of) the closure of R_{i_0}.

Let R_1 and R_2 be a pair of relation schemes for which there exists $Y \subseteq R_2 - R_1$ such that $R_1 \cap R_2 \to Y \in F^+$. The join of r_1 and $\pi_{(R_1 \cap R_2) \cup Y}(r_2)$ is called an *extension join* [13]. A variant of extension joins, called *simple chase join expressions* (*scje's*) can be defined on the basis of ds's. Given the ds τ above, assuming it covers X, the scje for τ *with target* X is the PJ-expression:

$$\pi_X(R_{i_0} \bowtie \pi_{Y_1Z_1}(R_{i_1}) \bowtie \cdots \bowtie \pi_{Y_mZ_m}(R_{i_m})).$$

Example 2. Consider the following database scheme:

$\mathbf{R} = \{R_1(AB), R_2(ABCDEG)\}$
$F = \{AB \to D, BC \to E, B \to C, D \to G, E \to G\}$.

Then the following are ds's.

(1) $\langle B \to C, BC \to E, E \to G \rangle$ is a ds of R_1 covering $ABCG$.
(2) $\langle AB \to D, D \to G, B \to C \rangle$ is a ds of R_1 covering $ABCG$.
(3) $\langle\ \rangle$ (the empty sequence) is a ds of R_2 covering $ABCG$.

The following are scje's for the above three ds's, respectively.

(1) $E_1 = \pi_{ABCG}(R_1 \bowtie \pi_{BC}(R_2) \bowtie \pi_{BCE}(R_2) \bowtie \pi_{EG}(R_2))$,

(2) $E_2 = \pi_{ABCG}(R_1 \bowtie \pi_{ABD}(R_2) \bowtie \pi_{DG}(R_2) \bowtie \pi_{BC}(R_2))$,

(3) $E_3 = \pi_{ABCG}(R_2)$. $\square$

Now, as anticipated in Section 2.2, we show how to construct a tableau for a scje, as a special case of the general construction method [1, 2]. Given a scje

$$E = \pi_X(R_{i_0} \bowtie \pi_{Y_1Z_1}(R_{i_1}) \bowtie \cdots \bowtie \pi_{Y_mZ_m}(R_{i_m}))$$

corresponding to a ds $\tau = \langle f_1\colon Y_1 \rightarrow Z_1, \ \ldots, f_m\colon Y_m \rightarrow Z_m \rangle$, the tableau T_E for E has a summary containing distinguished variables on the attributes in X, and undefined on the other attributes, and a body composed of $m + 1$ rows, l_0, $l_1, \ldots, l_m$, corresponding to the $m + 1$ factors in the join, defined as follows:

—for every $0 \leq j \leq m$, row l_j has tag R_{i_j}

—for every $A \in U$, $l_0[A]$ is the distinguished variable a if $A \in X$, and an ndv otherwise.

—for every $j \geq 1$, l_j is defined on the basis of rows $l_0, l_1, \ldots, l_{j-1}$:

—if $A \in (Y_j \cap X)$, then $l_j[A]$ is the dv a;

—if $A \in (Y_j - X)$, then by definition of ds, there is an integer k, such that $1 \leq k \leq j - 1$ and $A \in Z_k$: then $l_j[A]$ is set equal to $l_k[A]$, and is therefore a repeated ndv;

—if $A \in (U - Y_j)$, the $l_j[A]$ is a ndv not appearing in rows $l_0, l_1, \ldots, l_{j-1}$.

Example 3. The tableau corresponding to the scje E_1 in Example 2 is the following:

A	B	C	D	E	G	TAG
a	b	c			g	
a	b	v_1	v_2	v_3	v_4	R_1
v_5	b	c	v_6	v_7	v_8	R_2
v_9	b	c	v_{10}	v_{11}	v_{12}	R_2
v_{13}	v_{14}	v_{15}	v_{16}	v_{11}	g	R_2

If the chase procedure is applied to T_E, the following tableau $CHASE_F(T_E)$ is obtained:

A	B	C	D	E	G	TAG
a	b	c			g	
a	b	c	v_2	v_3	g	R_1
v_5	b	c	v_6	v_3	g	R_2
v_9	b	c	v_{10}	v_3	g	R_2
v_{13}	v_{14}	v_{15}	v_{16}	v_3	g	R_2

3. CONTAINMENT AND EQUIVALENCE OF SCJE'S

The goal of this paper is the efficient optimization of unions of scje's. Scje's are PJ-expressions, and Sagiv and Yannakakis [26, pp. 642–643] have shown that a union of PJ-expressions has a minimum number of joins if and only if (i) there is no redundant expression in the union, and (ii) each expression in the union has a minimum number of joins. Therefore, we study containment and equivalence of scje's (in order to be able to eliminate redundant expressions in the unions) in this section, and minimization of scje's in the following. In both cases we find efficient algorithms, which take into account the restricted nature of scje's, and do not require the construction and the chase of the tableaux corresponding to the expressions.

The plan for this section is the following. We first recall existing results that relate containment and equivalence of expressions to containment and equivalence of the corresponding chased tableaux. Then we study the structure of the chased tableaux for scje's, and, finally, we characterize containment (and equivalence) of scje's on the basis of the associated ds's.

Graham and Mendelzon [11] studied the equivalence of SPJ-expressions with respect to a given set of dependencies. Chan [6] extended their results to containment of expressions, proving the following fact.

FACT 3.1. *Given a database scheme* $\mathbf{R}$ *with an associated set of fd's F and two SPJ-expressions* E_1, E_2, *then* $E_1 \supseteq E_2$ *if and only if* $CHASE_F(T_{E_1}) \supseteq CHASE_F(T_{E_2})$.

On the basis of Fact 3.1, we show some properties of scje's, which are a special case of SPJ-expressions. Let $\tau = \langle f_1 : Y_1 \rightarrow Z_1, \ldots, f_m : Y_m \rightarrow Z_m \rangle$ be a ds for a relation scheme R_{i_0} covering a set of attributes $X \subseteq U$, let $E = \pi_X(R_{i_0} \bowtie \pi_{X_1Y_1}(R_{i_1}) \bowtie \cdots \bowtie \pi_{Y_kZ_k}(R_{i_k}))$ be the corresponding scje, and let T_E be the tableau corresponding to E. We use $l_0, \ldots, l_k$ to denote the rows in the body of T_E corresponding to the subexpressions $R_{i_0}, \ldots, \pi_{X_kY_k}(R_{i_k})$ respectively. The following is a way to obtain $CHASE_F(T_E)$. Also, without loss of generality, we assume that the ndv's in l_0 follow, in the total order $\leq$ defined on the set V_N, all the other ndv's appearing in T_E. (In this way, whenever row l_0 is involved in a transformation during the chase, the values in l_0 will always be modified, and equated to the values in the other row involved.)

Algorithm 3.1. *A method for chasing the tableau for an scje.*
INPUT: *The tableau* T_E *for an scje, as above, and a set of fd's F.*
OUTPUT: $CHASE_F(T_E)$.
METHOD:

(1) *For* $j = 1$ *to* m
  apply to T_E *the transformation involving rows* l_0 *and* l_j,
  and the j*th fd in* τ, $Y_j \rightarrow Z_j$.
 Let the resulting tableau be T'_E.

(2) *Apply all possible transformations to* T'_E *to obtain* $CHASE_F(T_E)$.

LEMMA 3.1. *Algorithm* 3.1 *correctly produces the chased tableau* $CHASE(T_E)$, *and, for each* $1 \leq j \leq m$, *the* j*th transformation applied during step* (1) *equates* $l_0[Z_j]$ *to* $l_j[Z_j]$.

PROOF. We first prove that, for each $1 \le j \le m$, the jth transformation applied during step (1) of Algorithm 3.1 is valid and equates $l_0[Z_j]$ to $l_j[Z_j]$. We proceed by induction on j. The basis ($j = 1$) is trivial. Then, let $j > 1$, and assume that, for all $1 \le i \le j - 1$, the ith transformation is valid, and equates $l_0[Z_i]$ and $l_i[Z_i]$. Let us consider $Y_j \rightarrow Z_j$, the jth fd in the ds τ. By definition of ds, $Y_j \subseteq R_{i_0} Z_1 \cdots Z_{j-1}$. Let $Y_j = Y'Y''$, where $Y' \subseteq R_{i_0}$ and $Y'' \subseteq Z_1 \cdots Z_{j-1}$. By definition of T_E, for each $A \in Y'$, $l_0[A] = l_j[A]$ in T_E. Similarly for each $B \in Y''$, $l_j[B] = l_i[B]$ in T_E, where $B \in Z_i$, for some $1 \le i \le j - 1$. By the induction hypothesis, $l_0[Z_i] = l_i[Z_i]$, for all $1 \le i \le j - 1$. Hence $l_0[Y''] = l_j[Y'']$, and so $l_0[Y_j] = l_0[Y'Y''] = l_j[Y'Y''] = l_j[Y_j]$. Therefore, the jth transformation is valid. Since $Z_j \cap R_{i_0} Z_1 \cdots Z_{j-1} = \varnothing$, all the values in $l_0[Z_j]$ are unique ndv's before the application of the transformation, and therefore no contradiction arises; as a consequence of the assumption made just before presenting the algorithm, the values in $l_0[Z_j]$ are modified, and equated attribute-wise to those in $l_j[Z_j]$. This completes the induction.

We have therefore proved the second part of the claim, together with the validity of the transformations applied in step (1) of the algorithm. To complete the proof, we have to show that the algorithm correctly computes the chase of the tableau T_E. Now all transformations are valid, and, in step (2), transformations are applied as long as possible. Therefore, the algorithm specifies a possible sequence of valid transformations that produces a tableau that is no longer modifiable. Since the chase is a Church-Rosser process (that is, its result is independent of the order of application of the individual transformations [17, pp. 168–174]), the algorithm produces $\mathit{CHASE}_F(T_E)$. □

The following lemmas summarize some fundamental properties about the chased tableau $\mathit{CHASE}_F(T_E)$, which will enable us to know its content without actually building and chasing T_E.

LEMMA 3.2. *Let l_p be a row in T_E, with $p \ge 1$. Then the following are equivalent.*

(1) $A \in Y_p^+$.

(2) $l_p[A] = l_0[A]$ *in* $\mathit{CHASE}_F(T_E)$.

(3) $l_p[A]$ *is a significant symbol in* $\mathit{CHASE}_F(T_E)$.

PROOF. (1) ⇒ (2). After step (1) of Algorithm 3.1, $l_p[Y_p] = l_0[Y_p]$, for all $1 \le p \le k$. Since chasing T_E cannot encounter contradiction, if $A \in Y_p^+$, then $l_p[A] = l_0[A]$ in $\mathit{CHASE}_F(T_E)$.

(2) ⇒ (3). Follows directly from the definition of significant symbols.

(3) ⇒ (1). We show that, if $A \notin Y_p^+$, then $l_p[A]$ is a unique ndv in $\mathit{CHASE}_F(T_E)$. We proceed by induction on the number j of transformations applied to T'_E in step (2).

Basis: $j = 0$. Transformations in step (1) only change entries in l_0, and so, if $p \ne 0$, then every entry in l_p is unchanged after step (1). Therefore if $A \notin Y_p^+$, the value $l_p[A]$ is a unique ndv. Hence the basis is established.

Induction: $j > 0$. Suppose that, after the application of $(j - 1)$ transformations to T'_E, for all $p \ne 0$, for all $A \notin Y_p^+$, $l_p[A]$ is a unique ndv. Let the jth transformation involve the fd $X \rightarrow A$ and rows l_s and l_t. Now the claim is violated only if $s \neq 0$, $A \notin Y_s^+$, and $l_s[A]$ is a unique ndv (before the application of the transformation), or $t \neq 0$, $A \notin Y_t^+$, and $l_t[A]$ is a unique ndv.

If $s \neq 0$ and $l_s[X] = l_t[X]$, then the symbols in l_s are not unique ndv's, and so, by the induction hypothesis, $X \subseteq Y_s^+$; therefore, by the property of transitivity for fd's, $A \in Y_s^+$. The same argument can be carried out for l_t, thus completing the induction. □

A tableau for a PJ-expression E is *simple* [25] if, for each attribute $A \in U$, the corresponding column in $\mathit{CHASE}_F(T_E)$, contains at most one significant symbol.

LEMMA 3.3. *The tableau $\mathit{CHASE}_F(T_E)$ is simple.*

PROOF. Suppose there exist two distinct significant symbols $l_p[A]$ and $l_q[A]$ in column A. By Lemma 3.2, $l_p[A] = l_0[A]$ and $l_q[A] = l_0[A]$. Hence $l_0[A] = l_p[A] = l_q[A]$, a contradiction. Hence there is at most one significant symbol in column A. □

Let T_E and T_F be tableaux for scje's E and F, respectively, and l_s and l_t be any two rows in T_E and T_F, respectively. The row l_s is said to *subsume* l (written $l_s \geq l_t$) if, for all $A \in U$ whenever $l_t[A]$ is a significant symbol, then $l_s[A]$ is also a significant symbol.

LEMMA 3.4. *$l_0[A]$ is a significant symbol in $\mathit{CHASE}_F(T_E)$ if and only if $A \in XY_1^+Y_2^+ \cdots Y_k^+$.*

PROOF. (If) If A belongs to $Y_p^+ - R_{i_0}$, for some $1 \leq p \leq k$, then, by Lemma 3.2, we know that $l_0[A] = l_p[A]$ in $\mathit{CHASE}_F(T_E)$, and so $l_0[A]$ is a significant symbol. If A belongs to $X - Y_1^+Y_2^+ \cdots Y_k^+$, then, by definition of ds, we have that $A \in X \cap R_{i_0}$, and so $l_0[A]$ is a dv and hence a significant symbol in T_E.

(Only if) Suppose, by way of contradiction, that there exists an attribute A not belonging to $XY_1^+Y_2^+ \cdots Y_k^+$, such that $l_0[A]$ is a significant symbol. Since $A \notin X$, there is no dv in the A-column, and so $l_0[A]$ is a repeated ndv in $\mathit{CHASE}_F(T_E)$. Therefore, there exists a row $l_p \in \mathit{CHASE}_F(T_E)$, with $p > 0$ such that $l_0[A] = l_p[A]$. Then, by Lemma 3.2, A belongs to Y_p^+—a contradiction. □

LEMMA 3.5. *Let $l_p[A]$ be a repeated symbol in $\mathit{CHASE}_F(T_E)$, for some $0 \leq p \leq m$. Then there exists another row l_q in $\mathit{CHASE}_F(T_E)$ such that l_p and l_q have different tags and $l_p[A] = l_q[A]$.*

PROOF. Let us distinguish two cases.

(1) $p = 0$. Since $l_0[A]$ is a repeated symbol, there exists l_q such that $l_0[A] = l_q[A]$. But l_0 has a tag R_{i_0} that appears nowhere else, and so l_0 and l_q have different tags.

(2) $p \neq 0$. Since $l_p[A]$ is a significant symbol, by Lemma 3.2, $l_p[A] = l_0[A]$. Again, we know that l_p and l_0 have different tags. □

On the basis of the previous results, we can characterize containment of scje's, in terms of chased tableaux.

THEOREM 3.1. *Let E_1 and E_2 be two scje's with the same target X. $E_1 \supseteq E_2$ if and only if for each row s_i in $CHASE_F(T_{E_1})$ there exists a row t_j in $CHASE_F(T_{E_2})$ such that $s_i[TAG] = t_j[TAG]$ and $t_j \geq s_i$.*

PROOF. By Fact 3.1, $E_1 \supseteq E_2$ if and only if $CHASE_F(T_{E_1}) \supseteq CHASE_F(T_{E_2})$. Therefore it suffices to show that $CHASE_F(T_{E_1}) \supseteq CHASE_F(T_{E_2})$ if and only if for each row s_i in $CHASE_F(T_{E_1})$ there exists a row t_j in $CHASE_F(T_{E_2})$ such that $s_i[TAG] = t_j[TAG]$ and $t_j \geq s_i$.

(If) Let it be the case that, for each row s_i in $CHASE_F(T_{E_1})$, there exists t_j in $CHASE_F(T_{E_2})$, with the same tag as s_i and subsuming it. On the basis of this correspondence, we consider the mapping ν, from the symbols T_{E_1} to the symbols in T_{E_2}, that, for every $s_i \in CHASE_F(T_{E_1})$, for every $A \in U'$, maps $s_i[A]$ to $t_j[A]$, and prove that it is a containment mapping. Following the definition of containment mapping, we show that ν is a function and that it is the identity on dv's (since there are no constants, no symbol is lower than dv's with respect to the partial order $\leq$) and tags.

Since, for every s_i, the corresponding t_j has the same tag ($s_i[TAG] = t_j[TAG]$), the mapping ν is the identity on tags.

Let $s_i[A]$ be a dv; by definition of ν, $\nu(s_i[A]) = t_j[A]$; we show that $t_j[A] = s_i[A]$, and therefore $\nu(s_i[A]) = s_i[A]$. Since $s_i[A]$ is a dv (the dv for attribute A), then $A \in X$ and therefore the dv for A appears in column A in T_{E_2}. Then, by definition of subsumption, since $t_j \geq s_i$ and $s_i[A]$ is a dv, we have that $t_j[A]$ is also a significant symbol. By Lemma 3.3, the dv is the only significant symbol in column A in $CHASE_F(T_{E_2})$, and therefore $t_j[A] = s_i[A]$.

We now show that ν is in fact a function, that is, if $s_i[A] = s_h[A]$, then $\nu(s_i[A]) = \nu(s_h[A])$. Let $s_i[A] = s_h[A] = v$. If v is a dv, we already know that $\nu(v) = v$ and therefore $\nu(s_i[A]) = \nu(s_h[A])$. Otherwise, v is a repeated ndv, and so $A \notin X$. Therefore, there is no dv in column A in T_{E_2}. Hence, since $t_j \geq s_i$, $t_j[A] = \nu(s_i[A])$ is a repeated ndv. Similarly, we could prove that $\nu(s_h[A])$ is a repeated ndv. By Lemma 3.3, there is only one repeated ndv in column A in $CHASE_F(T_{E_2})$, and so $\nu(s_i[A]) = \nu(s_h[A])$.

Summarizing, the two tableaux have identical target relation schemes (because E_1 and E_2 are scje's with the same target) and ν is a containment mapping; therefore, $CHASE_F(T_{E_1}) \supseteq CHASE_F(T_{E_2})$.

(Only if) Assume that $CHASE_F(T_{E_1}) \supseteq CHASE_F(T_{E_2})$; therefore there exists a containment mapping from $CHASE_F(T_{E_1})$ to $CHASE_F(T_{E_2})$. Let s_i be a row in $CHASE_F(T_{E_1})$ and let t_j be the row to which s_i is mapped; we show that $t_j \geq s_i$ and $t_j[TAG] = s_i[TAG]$. This last claim is immediate, since containment mappings are the identity on tags. To complete the proof, we show that, for every attribute $A \in U$, if $s_i[A]$ is significant, then $t_j[A]$ is also significant. Let us distinguish two cases:

(1) $s_i[A]$ is a dv. Since (when no constants are involved) a containment mapping is the identity on dv's, $t_j[A] = s_i[A]$, and so $t_j[A]$ is a dv, a significant symbol.
(2) $s_i[A]$ is a repeated ndv. By Lemma 3.3, there is no dv in column A in $CHASE_F(T_{E_1})$, and hence $A \notin X$, and so there is no dv in column A in $CHASE_F(T_{E_2})$ either. We claim that $t_j[A]$ is also a repeated ndv. Since $s_i[A]$

is a repeated ndv, by Lemma 3.5, there exists another row s_p in $CHASE_F(T_{E_1})$ such that $s_p[A] = s_i[A]$ and $s_p[TAG] \neq s_i[TAG]$. If $t_j[A]$ were a unique ndv, then s_p would be mapped to t_j, and the valuation function would not be the identity on tags, against the definition. Therefore $t_j[A]$ is not a unique ndv, and it is therefore significant. □

The above theorem characterizes containment of scje's in terms of their chased tableaux. However, it is not necessary to generate the chased tableaux physically, since, by Lemmas 3.2 and 3.4, we know exactly which values in each row of a chased tableau are significant symbols. This is confirmed by the following corollary, which characterizes containment of scje's in terms of the associated ds's. It refers to the scje's $E_\tau = \pi_X(R_{i_0} \bowtie \pi_{Y_1Z_1}(R_{i_1}) \bowtie \cdots \bowtie \pi_{Y_mZ_m}(R_{i_m}))$ and $E_\chi = \pi_X(R_{p_0} \bowtie \pi_{V_1W_1}(R_{p_1}) \bowtie \cdots \bowtie \pi_{V_nW_n}(R_{p_n}))$, corresponding to the ds's $\tau = \langle Y_1 \to Z_1, \ldots, Y_m \to Z_m \rangle$, and $\chi = \langle V_1 \to W_1, \ldots, V_n \to W_n \rangle$, respectively.

COROLLARY 3.1. *$E_\tau \supseteq E_\chi$ if and only if:*

(1) *If $R_{i_0} \neq R_{p_0}$, then there is an fd $V_k \to W_k$ in χ such that $R_{i_0} = R_{p_k}$ and $V_k^+ \supseteq XY_1^+ \cdots Y_m^+$, and*

(2) *for each $Y_j \to Z_j$ in τ, if $R_{i_j} = R_{p_0}$, then $XV_1^+ \cdots V_n^+ \supseteq Y_j^+$; otherwise, there exists a $V_k \to W_k$ in χ such that $V_k^+ \supseteq Y_j^+$ and $R_{p_k} = R_{i_j}$.*

PROOF. Let the sets of rows in T_{E_τ} and T_{E_χ} be $\{u_0, \ldots, u_m\}$ and $\{v_0, \ldots, v_n\}$, respectively. By Theorem 3.1, $E_\tau \supseteq E_\chi$ if and only if for each row u_j in $CHASE_F(T_{E_\tau})$, there exists a row v_k in $CHASE_F(T_{E_\chi})$ such that u_j and v_k have the same tag and $v_k \geq u_j$. Therefore we show that it is the case that, for each row u_j in $CHASE_F(T_{E_\tau})$, there exists a row v_k in $CHASE_F(T_{E_\chi})$ such that u_j and v_k have the same tag and $v_k \geq u_j$ if and only if conditions (1) and (2) in this corollary hold.

(If) Assume that conditions (1) and (2) hold. Consider row u_0. If $R_{i_0} = R_{p_0}$, then v_0 is the only row in $CHASE_F(T_{E_\chi})$ with the same tag as u_0; by Lemma 3.4, u_0 and v_0 have significant symbols on the attributes of $XY_1^+ \cdots Y_m^+$ and $XV_1^+ \cdots V_n^+$, respectively; then, it follows from (2) that $XV_1^+ \cdots V_n^+ \supseteq XY_1^+ \cdots Y_m^+$, and therefore, by definition of subsumption, $v_0 \geq u_0$. If $R_{i_0} \neq R_{p_0}$, by (1), there is an fd $V_k \to W_k$ in χ such that $V_k^+ \supseteq XY_1^+ \cdots Y_m^+$ and $R_{i_0} = R_{p_k}$; then u_0 and v_k have the same tag, and, by Lemmas 3.4 and 3.2, they have significant symbols on $XY_1^+ \cdots Y_m^+$ and V_k^+, respectively, and so $v_k \geq u_0$.

Now, consider a generic row u_j in $CHASE_F(T_{E_\tau})$, with $j \geq 1$, and the associated fd $Y_j \to Z_j$ in τ; by (2), if $R_{i_j} = R_{p_0}$, then $XV_1^+ \cdots V_n^+ \supseteq Y_j^+$; otherwise, there exists a $V_k \to W_k$ in χ such that $V_k^+ \supseteq Y_j^+$ and $R_{p_k} = R_{i_j}$. In the first case, arguing as above about significant symbols, we can conclude that $v_0 \geq u_j$; in the second case, that $v_k \geq u_j$. Therefore, in either case, there is a row v_h in $CHASE_F(T_{E_\chi})$ with the same tag as u_j such that $v_h \geq u_j$.

(Only if) Assume that, for each row u_j in $CHASE_F(T_{E_\tau})$, there exists a row v_k in $CHASE_F(T_{E_\chi})$ such that u_j and v_k have the same tag and $v_k \geq u_j$. We show that conditions (1) and (2) hold.

(1) Let $R_{i_0} \neq R_{p_0}$; therefore, since u_0 and v_0 have different tags, there is a row v_j in $CHASE_F(T_{E_\chi})$, with the tag R_{i_0} such that $v_j \geq u_0$. Then, by Lemmas 3.2 and 3.4, it follows that $V_k^+ \supseteq XY_1^+ \cdots Y_m^+$.

(2) Let $Y_j \to Z_j$ be an fd in τ. Let us consider two subcases.
   (a) $R_{i_j} = R_{p_0}$. Then, v_0 is the only row in $CHASE_F(T_{E_\chi})$ with the same tag as u_j; therefore, it is the case that $v_0 \geq u_j$, and so, by Lemmas 3.2 and 3.4, $XV_1^+ \cdots V_n^+ \supseteq Y_j^+$.
   (b) $R_{i_j} \neq R_{p_0}$. Then there is a row v_k in $CHASE_F(T_{E_\chi})$, with $k \geq 1$, with the same tag as u_j and subsuming it. Then, by Lemma 3.2, $V_k^+ \supseteq Y_j^+$, and so, there is an fd $V_k \to W_k$ in χ such that $V_k^+ \supseteq Y_j^+$. □

Given E_τ and E_χ as above, in order to test whether $E_\tau \supseteq E_\chi$, we can proceed as follows:

(1) Compute the closures of the Y_i's and V_j's.
(2) If $R_{i_0} \neq R_{j_0}$, test the existence of the fd in χ, required by condition (1) of Corollary 3.1.
(3) For each fd in τ, check for the satisfaction of condition (2) of Corollary 3.1.

Step (1) can be executed in time $O((m + n) \times \| F \|)$, where m, n are the lengths of the two derivation sequences, and $\| F \|$ is the space required to write the set of fd's F, by applying the widely known closure algorithm in [5] to each set of attributes in the ds's. Steps (2) and (3) essentially require pairwise comparisons of sets of attributes, and can therefore be executed in time $O(m \times n \times | U |)$, where m, n are as above, and $| U |$ is the number of attributes in the universe. Therefore, containment of scje's can be tested in time $O((m + n) \times \| F \| + m \times n \times | U |)$.

The following corollary is the special case of Corollary 3.1, if E_τ and E_χ are scje's for the same relation scheme.

COROLLARY 3.2. *Let τ, χ, E_τ, E_χ be as above, with the further condition that $R_{i_0} = R_{p_0}$. Then, $E_\tau \supseteq E_\chi$ if and only if for each $Y_j \to Z_j$ in τ there exists a $V_k \to W_k$ in χ such that $V_k^+ \supseteq Y_j^+$ and $R_{p_k} = R_{i_j}$.*

Example 4. Consider again the database scheme and the scje's in Example 2:

$\mathbf{R} = \{R_1(AB), R_2(ABCDEF)\}$.
$F = \{AB \to D, BC \to E, B \to C, D \to F, E \to F\}$.
$E_1 = \pi_{ABCF}(R_1 \bowtie \pi_{BC}(R_2) \bowtie \pi_{BCE}(R_2) \bowtie \pi_{EF}(R_2))$.
$E_2 = \pi_{ABCF}(R_1 \bowtie \pi_{ABD}(R_2) \bowtie \pi_{DF}(R_2) \bowtie \pi_{BC}(R_2))$.
$E_3 = \pi_{ABCF}(R_2)$.

Let us try to eliminate redundant subexpressions from the union of the three scje's. Let us consider E_1 and E_2: since $ABD^+ = ABCDEF = U$, by Corollary 3.2, $E_1 \supseteq E_2$. Then, E_1 cannot contain E_3, by Theorem 3.1, since T_{E_1} has a row with tag R_1 and T_{E_3} does not. Finally, $E_3 \supseteq E_1$ if and only if one of the closures of BC, BCE, or EF contains $ABCF$. Since $BC^+ = BCEF$, $BCE^+ = BCEF$ and $EF^+ = EF$, E_3 does not contain E_1. Hence $E_1 \cup E_3$ is an equivalent expression to $E_1 \cup E_2 \cup E_3$, and it does not contain redundant subexpressions. In the next section, we will see how to minimize the individual scje's, in order to obtain the optimal expression.

4. OPTIMIZATION OF SCJE'S

In this section we study the problem of optimization of scje's. As usual for PJ-expressions, optimization means minimization of the number of joins in the expression, which corresponds to minimization of the corresponding tableau. The general problem of minimizing a tableau is NP-complete, whereas for simple tableaux the problem can be solved in polynomial time [1, 2, 25]. By Lemma 3.3, the chased tableaux for scje's are simple, and therefore can be minimized efficiently; however, the above methods require us to construct and chase the tableaux. In this section, we show that this is not necessary. We first characterize when a chased tableau for an scje is minimal. Then we show how, on the basis of this result, it is possible to obtain a minimal equivalent expression for an scje, without generating the chased tableau.

It is known [2, p. 446] that, for every tableau T, there is an equivalent minimal tableau whose body is a subset of the body of T. The following lemma characterizes this minimal tableau if T is the tableau for an scje.

LEMMA 4.1. *Let E be an scje, T_E the corresponding tableau, and T_E^* the minimal tableau contained in $\text{CHASE}_F(T_E)$. Then a row l_k in the body of $\text{CHASE}_F(T_E)$ is in T_E^* if and only if there is no other row l_k in the body of $\text{CHASE}_F(T_E)$, with the same tag, such that $l_k \geq l_j$.*

PROOF. (If) Let l_j, l_k be distinct rows in $\text{CHASE}_F(T_E)$; assume, by way of contradiction, that they are both in T_E^*, have the same tag, and $l_k \geq l_j$. Now, let ν be the mapping that (i) maps the unique symbols in l_j to the respective symbols in l_k, and (ii) is the identity on the other symbols. Clearly, ν is a containment mapping from T_E^* to$T_E^* - \{l_j\}$, and so, since there is always a containment mapping from $T_E^* - \{l_j\}$ to T_E^*, T_E^* is not minimal, a contradiction.

(Only if) Let l_j be a row in the body of $\text{CHASE}_F(T_E)$ such that there is no other row l_k in the body of T_E^* such that $l_k \geq l_j$ and $l_k[TAG] = l_j[TAG]$. Then, since T_E^* is equivalent to $\text{CHASE}_F(T_E)$, there is a containment mapping ν, from $\text{CHASE}_F(T_E)$ to T_E^*; therefore, $\nu(l_j)$ is in the body of T_E^*. By definition of containment mapping and of subsumption, $\nu(l_j) \geq l_j$ and $\nu(l_j)[TAG] = l_j[TAG]$, and so, if there is no row l_k in the body of T_E^*, distinct from l_j, such that $l_k \geq l_j$ and $l_k[TAG] = l_j[TAG]$, it must be the case that $\nu(l_j) = l_j$, and so l_j is in the body of T_E^*. □

By Lemma 4.1, the chased tableau for an scje can be minimized by repeatedly eliminating rows subsumed by other tuples. We have therefore proved the correctness of the following algorithm.

Algorithm 4.1. *Minimizing the chased tableau for an scje.*

INPUT: *A tableau $\text{CHASE}_F(T_E)$, where E is an scje*:

$$E = \pi_X(R_{i_0} \bowtie \pi_{Y_1Z_1}(R_{i_1}) \bowtie \cdots \bowtie \pi_{Y_mZ_m}(R_{i_m})).$$

OUTPUT: *A minimal tableau T_E^* equivalent to $\text{CHASE}_F(T_E)$.*

METHOD:

(1) *Let $T = \text{CHASE}_F(T_E)$*

(2) *For $j = 1$ to m do*:

If there is a row l_k in T, such that $k \neq j$, $l_j[TAG] = l_k[TAG]$ and $l_j \geq l_k$, then delete l_k from T.

(3) *Let $T_E^* = T$.*

Algorithm 4.1 can be executed in quadratic time in the number of rows of the tableau, after having chased it. Let l_p be a row in $\mathit{CHASE}_F(T_E)$. Row l_p is called a *dominant* row if l_p is in T_E^*. Otherwise it is called a *deleted* row.

By Theorems 3.2 and 3.4, we know exactly which symbol in $\mathit{CHASE}_F(T_E)$ is significant. Therefore we can generate a minimal tableau T_E^* for $\mathit{CHASE}_F(T_E)$ (and therefore an optimal expression for the scje E) without constructing the tableau T_E and chasing it. The following algorithm performs this task.

Algorithm 4.2. *Optimizing an scje.*
INPUT: *An scje* $E = \pi_X(R_{i_0} \bowtie \pi_{Y_1Z_1}(R_{i_1}) \bowtie \cdots \bowtie \pi_{Y_mZ_m}(R_{i_m}))$.
OUTPUT: *An optimal expression* E' *equivalent to* E.
METHOD:

(1) *Initialize three arrays of length m as follows.*
For $j := 1$ *to* m *do*:
$TAG[j] := R_{i_j}$
$SUBEXP[j] := Y_jZ_j$
$CLOSURE[j] := Y_j^+$

(2) *For* $j := 1$ *to* m *do*:
If there exists k, $k \neq j$, $1 \le k \le m$, *such that* $TAG[k] = TAG[j]$ *and* $CLOSURE[k] \supseteq CLOSURE[j]$ *then*
$SUBEXP[k] := SUBEXP[k] \cup SUBEXP[j]$
$TAG[j] := \varnothing$.

(3) *Construct an expression* J *as follows.*
$J := R_{i_0}$
For $j = 1$ *to* m *do*:
If $TAG[j] \neq \varnothing$ *then* $J := J \bowtie \pi_{SUBEXP[j]}(TAG[j])$.

(4) *Return the final expression.*
$E' := \pi_X(J)$.

Let us call a subexpression $\pi_{Y_jZ_j}(R_{i_j})$ in E a *dominant* subexpression if $\mathrm{TAG}[j] \neq \varnothing$ after step (2) of Algorithm 4.2. Otherwise, it is called a *deleted* subexpression. By Lemma 3.2, $l_p \ge l_q$ exactly when $Y_p^+ \supseteq Y_q^+$. Hence step (2) of Algorithm 4.2 is essentially the same as step (2) of Algorithm 4.1. That is, the jth row of $\mathit{CHASE}_F(T_E)$ is in T_E^* if and only if $\mathrm{TAG}[j] \neq \varnothing$ after step (2) of Algorithm 4.2. Hence there is a one-to-one correspondence between dominant rows in the tableau $\mathit{CHASE}_F(T_E)$ and dominant subexpressions in E. Hence the number of join operations in E' is the same as in any optimal expression equivalent to E. To complete the proof of correctness of Algorithm 4.2 we show that the expression E' is in fact equivalent to E. We prove this by showing that $\mathit{CHASE}_F(T_{E'}) \equiv T_E^*$, where T_E^* is the tableau produced by Algorithm 4.1.

LEMMA 4.2. $\mathit{CHASE}_F(T_{E'}) \equiv T_E^*$.

PROOF. From the argument presented above, there is a one-to-one correspondence between the rows in T_E^* and the subexpressions in E'. Let $\{s_1, \ldots, s_q\}$ be the set $\{j \mid TAG[j] \neq \varnothing\}$; therefore, $E' = R_{i_0} \bowtie \pi_{SUBEXP[s_1]}(R_{i_{s_1}}) \bowtie \cdots \bowtie \pi_{SUBEXP[s_q]}(R_{i_{s_q}})$. Consider the tableau $T_{E'}$ for E', and let $u_0, \ldots, u_q$ be the rows in $\mathit{CHASE}_F(T_{E'})$. Also, let $w_0, \ldots, w_q$ be the rows in T_E^*, respectively equal to the rows $v_0, v_{s_1}, \ldots, v_{s_q}$ of $\mathit{CHASE}_F(T_E)$.

Clearly, $w_j[TAG] = u_j[TAG]$, for $1 \le j \le q$. To complete the proof, we show that $w_j \ge u_j$ and $u_j \ge w_j$; by Theorem 3.1, this will imply $T_E^* \supseteq CHASE_F(T_{E'})$ and $CHASE_F(T_{E'}) \supseteq T_E^*$ and so $CHASE_F(T_{E'}) \equiv T_E^*$. By Lemma 3.2, if $j \ge 1$, $u_j[A]$ is significant if and only if $A \in SUBEXP[s_j]^+$ and $w_j[A]$ is significant if and only if $A \in Y_{s_j}^+$; by step (2) of Algorithm 4.2, $SUBEXP[s_j]^+ = Y_{s_j}^+$, and therefore $u_i[A]$ is significant if and only if $w_j[A]$ is significant; so $w_j \ge u_j$ and $u_j \ge w_j$.
Similarly, by Lemma 3.4, $w_0[A]$ is significant if and only if $A \in \cup_{j=1}^{q}(SUBEXP[s_j])^+ \cup R_{i_0}$ and $u_0[A]$ is significant if and only if $A \in R_{i_0} Y_{s_1}^+ \cdots Y_{s_q}^+$; again, by step (2) of Algorithm 4.2, we have $\cup_{j=1}^{q}(SUBEXP[s_j])^+ \cup R_{i_0} = R_{i_0} Y_{s_1}^+ \cdots Y_{s_q}^+$, and so $u_0[A]$ is significant if and only if $w_0[A]$ is significant, that is, $w_0 \ge u_0$ and $u_0 \ge w_0$. □

The most significant factor in step (1) of Algorithm 4.2 is computing the closures. Computing the closure for Y_j requires $O(\| F \|)$ steps. Therefore step (1) requires $O(m \times \| F \|)$. In step (2) the most significant factor is the comparison of two closures. There are at most m^2 comparisons in this step, each of which requires $O(| U |)$ operations. Hence step (2) has complexity $O(m^2 \times | U |)$ steps. Steps (3) and (4) are negligible with respect to the previous ones. Therefore, the time complexity of Algorithm 4.2 is $O(m \times \| F \| + m^2 \times | U |)$. So we proved the following theorem.

THEOREM 4.1. *The expression E' produced by Algorithm 4.2 is equivalent to E and is minimal in the number of join operations. The time complexity of Algorithm 4.2 is $O(m \times \| F \| + m^2 \times | U |)$.*

The combined use of the results in Section 3 and Section 4 yields an efficient method for optimizing unions of scje's: given an expression of this class, redundant subexpressions (i.e., scje's) can be discovered by means of Corollary 3.1, and therefore eliminated from the union; then, each of the remaining scje's is minimized by means of Algorithm 4.2. Since each stage can be done efficiently, optimization of unions of scje's can be done efficiently.

Example 5. Let us consider again the database scheme in Examples 2 and 4.

$$\mathbf{R} = \{R_1(AB), R_2(ABCDEG)\}.$$

$$F = \{AB \to D, BC \to E, B \to C, D \to G, E \to G\}.$$

Let us consider the expression $E_1 \cup E_3$ in Example 3, where

$$E_1 \cup E_3 = \pi_{ABCG}(R_1 \bowtie \pi_{BC}(R_2) \bowtie \pi_{BCG}(R_2) \bowtie \pi_{EF}(R_2)) \cup \pi_{ABCG}(R_2).$$

Since E_3 is already minimal, let us consider E_1. $BC^+ = BCEG$, $BCE^+ = BCEG$ and $EG^+ = EG$. Hence in step (2) of Algorithm 4.2, TAG[1] and TAG[3] are set to ∅ and SUBEXP[2] = $BCEG$. Hence $\pi_{ABCG}(R_1 \bowtie \pi_{BCEG}(R_2))$ is returned in step (3) of the Algorithm 4.2. Therefore the optimal expression equivalent to $E_1 \cup E_2 \cup E_3$ is $\pi_{ABCG}(R_1 \bowtie \pi_{BCEG}(R_2)) \cup \pi_{ABCG}(R_2)$.

5. CONCLUSIONS

The common feature of the results of this paper is that it is possible to study the major properties of expressions of a restricted class, scje's, without explicitly building and chasing the tableaux corresponding to the expressions. With the results existing in the literature, the optimization of a union of PJ-expressions, $E = E_1 \cup \cdots \cup E_k$, where, for each $1 \le j \le k$, E_j contains at most m joins, would require time $O(k \times (m \times |U| + \|F\|)^2 \times \log(m \times |U| + \|F\|))$ just to build and chase the k tableaux corresponding to the subexpressions; then, every comparison, or minimization, of tableau would require an amount of time that is quadratic in the size of the tableaux, for a restricted subclass, and bigger in general: in the favorable case [25] $O(k^2 \times m^2 \times |U|^2)$ time would therefore be required to execute pairwise comparisons of the k subexpressions. Our results show that, for the restricted class of scje's, the whole process can be carried out in time $O(k \times m \times \|F\| + k^2 \times m^2 \times |U|)$, by means of an implementation of the process suggested by Corollary 3.1, where the closures of the sets of attributes are never computed twice. Therefore we have shown that it is possible to take advantage of the restricted nature of scje's, in order to obtain more efficient algorithms.

Scje's are a meaningful subclass of PJ-expressions, since they can be used to compute the total projections of the representative instance [6, 19]: if the database scheme is independent then, for every X, the X-total projection of the representative instance can be computed by means of a union of scje's. Therefore, our results show how to optimize the expressions that compute the total projections. As a matter of fact, various algorithms have been proposed that, given X, generate the expression that computes the X-total projection [3, 15, 24]; the algorithm proposed by Ito et al. [15] produces an expression that is already optimal, and therefore need not be optimized. However it is possible to implement this algorithm following a strategy proposed by Atzeni and Chan [3, p. 188]: essentially, a number of precomputations (including the closures of all the sets of attributes involved in fd's, and some subexpressions associated with the attributes) are performed in the beginning, when the scheme is defined, and never repeated; then, each time a total projection is needed, the union of scje's is generated more efficiently, and optimized taking advantage of the precomputed closures. If this method is adopted, the use of our optimization algorithm is definitely convenient.

ACKNOWLEDGMENTS

We would like to thank Fred Lochovsky, Alberto Mendelzon, and the referees for their helpful comments on earlier drafts of this paper.

REFERENCES

1. AHO, A. V., SAGIV, Y., AND ULLMAN, J. D. Equivalence of relational expressions. *SIAM J. Comput. 8*, 2 (May 1979), 218–246.
2. 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.
3. ATZENI, P., AND CHAN, E. P. F. Efficient query answering in the representative instance approach. In *Fourth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems* (Portland, Ore., Mar. 25–27, 1985). ACM, New York, 1985, pp. 181–188.
4. ATZENI, P., AND PARKER, D. S., JR. Assumptions in relational database theory. In *ACM SIGACT-SIGMOD Symposium on Principles of Database Systems* (Los Angeles, Calif., Mar. 29–31, 1982). ACM, New York, 1982, pp. 1–9.
5. BEERI, C., AND BERNSTEIN, P. A. Computational problems related to the design of normal form relational schemas. *ACM Trans. Database Syst. 4*, 1 (Mar. 1979), 30–59.

6. CHAN, E. P. F. Query Answering and Schema Analysis under the Weak Instance Model. Ph.D. dissertation, University of Toronto, 1984.
7. CHAN, E. P. F., AND ATZENI, P. On the properties and characterization of connection-trap-free schemes. In *Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems* (Cambridge, Mass., Mar. 24–26, 1986). ACM, New York, 1986, pp. 140–147.
8. CHAN, E. P. F., AND MENDELZON, A. O. Answering queries on embedded-complete database schemes. *J. ACM 34*, 2 (Apr. 1987), 349–375.
9. CHAN, E. P. F., AND MENDELZON, A. O. Independent and separable database schemes. *SIAM J. Comput. 16*, 5 (Oct. 1987), 841–851.
10. DOWNEY, P. J., SETHI, R., AND TARJAN, R. E. Variations on the common subexpression problem. *J. ACM 27*, 4 (Oct. 1980), 758–771.
11. GRAHAM, M. H., AND MENDELZON, A. O. Strong equivalence of relational expressions under dependencies. *Inf. Process. Lett. 14*, 2 (Apr. 1982), 57–62.
12. GRAHAM, M. H., AND YANNAKAKIS, M. Independent database schemas. *J. Comput. Syst. Sci. 28*, 1 (1984), 121–141.
13. HONEYMAN, P. Extension joins. In *Sixth International Conference on Very Large Data Bases* (Montreal, Oct. 1–3, 1980). IEEE, New York, 1980, pp. 239–244.
14. HONEYMAN, P. Testing satisfaction of functional dependencies. *J. ACM 29*, 3 (July 1982), 668–677.
15. ITO, M., IWASAKI, M., AND KASAMI, T. Some results on the representative instance in relational databases. *SIAM J. Comput. 14*, 2 (May 1985), 334–354.
16. KENT, W. Consequences of assuming a universal relation. *ACM Trans. Database Syst. 6*, 4 (Dec. 1981), 539–556.
17. MAIER, D. *The Theory of Relational Databases.* Computer Science Press, Potomac, Md., 1983.
18. MAIER, D., MENDELZON, A. O., AND SAGIV, Y. Testing implications of data dependencies. *ACM Trans. Database Syst. 4*, 4 (Dec. 1979), 455–468.
19. MAIER, D., ROZENSHTEIN, D., AND WARREN, D. S. Windows functions. In *Advances in Computing Research, Vol. 3.* P. C. Kanellakis and F. Preparata, Eds. JAI Press, Greenwich, Conn., 1986, pp. 213–246.
20. MAIER, D., ULLMAN, J. D., AND VARDI, M. On the foundations of the universal relation model. *ACM Trans. Database Syst. 9*, 2 (June 1984), 283–308.
21. MENDELZON, A. O. Database states and their tableaux. *ACM Trans. Database Syst. 9*, 2 (June 1984), 264–282.
22. SAGIV, Y. Can we use the universal instance assumption without using nulls? In *ACM SIGMOD International Conference on Management of Data* (Ann Arbor, Mich., Apr. 29–May 1, 1981), ACM, New York, 1981, pp. 108–120.
23. SAGIV, Y. A characterization of globally consistent databases and their correct access paths. *ACM Trans. Database Syst. 8*, 2 (June 1983), 266–286.
24. SAGIV, Y. Evaluation of queries in independent database schemes. 1984. Unpublished manuscript.
25. SAGIV, Y. Quadratic algorithms for minimizing joins in restricted relational expressions. *SIAM J. Comput. 12*, 2 (May 1983), 316–328.
26. SAGIV, Y., AND YANNAKAKIS, M. Equivalence among relational expressions with the union and difference operators. *J. ACM 27*, 4 (Oct. 1980), 633–655.
27. ULLMAN, J. D. The U.R. strikes back. In *ACM SIGACT-SIGMOD Symposium on Principles of Database Systems* (Los Angeles, Calif., Mar. 25–27, 1982), ACM, New York, 1982, pp. 10–22.
28. ULLMAN, J. D. Universal relation interfaces for database systems. In *IFIP Congress* (Paris, Sept. 19–23, 1983). North-Holland, Amsterdam, 1983, pp. 243–252.
29. YANNAKAKIS, M. Algorithms for acyclic database schemes. In *Seventh International Conference on Very Large Data Bases* (Cannes, Sept. 9–11, 1981). IEEE, New York, 1981, pp. 82–94.
30. YANNAKAKIS, M. Querying weak instances. In *Advances in Computing Research, Vol. 3.* P. C. Kanellakis and F. Preparata, Eds. JAI Press, Greenwich, Conn., 1986, pp. 185–211.

Received December 1984 revised November 1987; accepted April 1988