ON INEQUALITY TABLEAUX

by

Anthony Klug

Computer Sciences Technical Report #403

November 1980

On Inequality Tableaux

Anthony Klug

University of Wisconsin

Abstract

Tableaux are representations of relational algebra expressions suitable for a number of useful computations. The tableau construction is generalized in this paper so that relational algebra expressions having domain/value comparisons with '<', '≤', etc. can be represented. Transformation rules for the tableau chase are given for these "inequality tableaux", and one use of the chase is given in which the validity of view dependencies can be determined. The equivalence and optimization of relational algebra expressions (including '<' comparisons) is shown equivalent to existence of certain row preserving functions on tableaux representing the expressions.

Keywords and Phrases: relational algebra, tableau, inequality comparisons, chase, view constraints, equivalence, optimization
CR Categories: 4.33, 5.21

Author's address: Computer Sciences Dept., University of Wisconsin-Madison, Madison, WI 53706

Contents

1. Introduction

Tableaux have been used by a number of authors to solve a variety of problems in the relational database model. For example, they have been used to optimize relational expressions [AhSU], to test for losslessness of joins [AhBU], to test validity of dependencies [MaMS] and to check correctness of views [KlPr]. The tableaux used in these works could only represent relational expressions which contained equality comparisons among domains and between domains and constants. In the "real world", queries do not consist solely of equality comparisons. Tableaux would be more useful if they could model queries with the domain comparison operators "less-than", "greater-than-or-equal", etc. In this paper we introduce the notion of "inequality tableaux", tabular representations of relational algebra expressions which contain these general comparison operators. We then present a number of useful computations which can be performed on inequality tableaux.

1.1. Overview of Paper

After presenting definitions for relational algebra and inequality tableaux (Sections 2 and 3), we first show that the expressive power of relational algebra is exactly the same as the expressive power of inequality tableaux. Then we consider chase rules for inequality tableaux (Section 4). These rules are basic to the use of tableaux. In Section 5, we show how chases can be used to verify constraints on views defined with expressions having inequality comparisons. In Section 6, we consider equivalence and optimization of inequality tableaux.

2. Relational Definitions

In this section we present our basic set of definitions. we define relations, instances, dependencies and relational algebra expressions.

The formal model we use does not make the universal instance assumption [BeGo2].

A schema consists of a list $<R_1,\ldots,R_N>$ of relation names. With each relation name R_i is associated an integer called its degree and denoted $\deg(R_i)$. Throughout this paper, one fixed schema $<R_1,\ldots,R_N>$ is assumed. In the examples, we assume that R_1 and R_2 are binary and that R_3 and R_4 have three domains.

An instance (or database state) I of schema $<R_1,\ldots,R_N>$ is an N+2-tuple $<D,O,I_1,\ldots,I_N>$, where D is the domain of values, O is a partial, asymmetric, transitive order[1] on D, and for each $i=1,\ldots,N$, $I_i \subseteq D^{\deg(R_i)}$. The relation O specifies which data elements are less than other data elements. Domains of all relations are taken without loss of generality to be the same set D, and D^m is the set of all m-tuples over D. For convenience, we will assume that the set $\mathbb{N}$ of natural numbers is embedded in D and that the less-than relation on natural numbers is consistent with the order on D[2].

[1] For all pairs (a,b) in O, (b,a) is not in O, and if (a,b) and (b,c) are in O, then (a,c) is also in O.
[2] What we really want is the kind of interpretations (instances) which appear in mathematical logic [Ende] in which the formal languages have constant symbols and the interpretations have interpretations of constant symbols. To simplify matters for those not familiar with models in logic, we use the natural numbers both as constant symbols in our language (relational algebra) and as their own interpretations in the instances. Other data types may be considered to be coded as natural numbers.

We usually think of data dependencies as properties of schema relations. However, if we are also studying views and dependencies on views, a more general idea of dependency is convenient. We think of a dependency as a statement about a set of tuples of the appropriate degree:

A <u>functional dependency</u> (<u>FD</u>) <u>for degree k</u> is a pair $\langle Z,A\rangle$, also written $Z \rightarrow A$, where $Z \subseteq \{1,2,...,k\}$, $A \in \{1,2,...,k\}$ and $A \notin Z$.

A <u>join dependency</u> (<u>JD</u>) <u>for degree k</u> is a sequence $\langle X_1,...,X_m\rangle$, also written $[X_1,...,X_m]$, in which each X_i is a sublist of $\{1,...,k\}$ and where $\cup X_i = \{1,2,...,k\}$. A sublist of a set X is a nonrepeating sequence whose entries are taken from X.

A <u>constraint</u> is either an FD or a JD.

A <u>schema FD</u> is a pair $\langle R_i, Z \rightarrow A\rangle$, also written $R_i{:}Z \rightarrow A$, where R_i is a schema relation and $Z \rightarrow A$ is an FD for degree $\deg(R_i)$. Similarly, a <u>schema JD</u> is a pair $\langle R_i,S\rangle$, also written $R_i{:}S$, where S is a JD for degree $\deg(R_i)$.

An FD $Z \rightarrow A$ for degree k is <u>true</u> in a set X of tuples of degree k if for all tuples t_1, t_2 in X, if $t_1[Z] = t_2[Z]$, then $t_1[A] = t_2[A]$. Brackets '[', ']' indicate projection on the listed domains. A JD $[J_1,...,J_m]$ for degree k is <u>true</u> in a set X of tuples of degree k if every element $t \in D^k$ such that $t[J_i]$ is in $X[J_i]$ $(i=1,...,m)$ is in X. A schema constraint $\langle R_i,c\rangle$ is <u>true</u> in an instance $I = \langle D,O,I_1,...,I_N\rangle$ if c is true in the set I_i of tuples. If C is a set of schema constraints, <u>sat(C)</u> is the set of instances in which each constraint of C is true.

The query language we define is a basic relational algebra [Codd]. The set $\mathbb{E}$ of <u>expressions</u> over our fixed schema is defined as follows: Base relations R_i are expressions. If $c \in \mathbb{N}$, then the unary constant relation {c} is an expression. Given expressions e_1, e_2, we get new expressions by taking projections, $e_1[X]$, where X is a sublist of the domains of e_1, cross products, $e_1 \times e_2$, restrictions, $e_1[X\theta Y]$, where $X,Y \in doms(e)$ and θ is '=' or '<', and unions, $e_1 \cup e_2$, where e_1 and e_2 have the same degree.

With these operators we can also define selections, joins and intersections.

For each $e \in \mathbb{E}$ of degree k and for each instance I, the value of e on I, denoted e(I), is a subset of D^k. The formal definition, which is omitted, gives the usual semantics for relational algebra operators [Codd].

Example 1. Consider the schema:

emp(name,dept#)   dept(dept#, mgr#, budget)

The query "List the names of all employees whose departments are headed by manager 15 or which have a budget smaller than \$10000" can be expressed as the algebra statement:

(emp [2=1] (dept[2 = '15' or 3 < '10000']))[1]

(We quote constants to distinguish them from domain numbers.) In terms of the primitive operations, this statement is:

```
( emp ×
   (dept × {15})[2=4][1,2,3] U (dept × {10000})[3<4][1,2,3]
)[2=3] [1]
```

There are two properties of expressions we would like to be able to determine: We would like to know what dependencies hold on sets of tuples derived from expressions, and we would like to know when the tuple sets from one expression are always contained in or are equal to the tuple sets derived from another expression. Knowing these properties would allow use to verify correctness of view constraints, and would help us optimize expressions. Thus we define the following concepts:

An _expression constraint_ is a pair <e,c>, also written e:c, where $e \in \mathbb{E}$ and c is an FD or a JD for degree deg(e). An expression constraint e:c is _true_ in instance I if c is true in e(I).

An expression constraint e:c is valid in a set P of instances if for every $I \in P$, e:c is true in I.

Expression e_1 is _contained in_ expression e_2 _with respect to_ a set P of instances, written $e_1 \subseteq_P e_2$, if $e_1(I) \subseteq e_2(I)$ for all instances I in P. We write $e_1 \subseteq e_2$ if $e_1(I) \subseteq e_2(I)$ for all instances. Expression e_1 is _equivalent_ to expression e_2 on a set P of instances, written $e_1 \equiv_P e_2$, if $e_1(I) = e_2(I)$ for all instances I in P.

3. Inequality Tableaux

Tableaux [AhBU] [AhSU] [ChMe] [SaYa] [KlPr] [MaMS] are shorthand notations for relational expressions which are well suited for computational purposes. Previous definitions of tableaux have modeled only projections, equi-selections and natural joins on universal instances [AhSU] or projections, equi-selection and equi-restriction and cross product on arbitrary instances [ChMe] [KlPr]. Here, we introduce a more general concept of tableau which can represent the relational algebra operators of the previous section in which restrictions may have "less-than" comparisons. We also incorporate a previous type of generalization [KlPr] in which multiple summary rows are allowed. To motivate the definition, we first consider the conjunctive queries of Chandra and Merlin. The derivation of tableaux from conjunctive queries is particularly easy to see.

A <u>conjunctive query</u> is a first-order predicate calculus formula of the form:

$$(x_1,\ldots,x_k).\exists x_{k+1}\ldots x_m.A_1\&\ldots\&A_r,$$

where each A_i is an atomic formula $R_j(t_1,\ldots,t_p)$, where each term t is a variable or a constant. Given an instance I, the <u>result</u> of a conjunctive query is the set of all k-tuples $\langle d_1,\ldots,d_k\rangle$ of values which make the query true when substituted for $x_1,\ldots,x_k$, respectively. It can be shown that every expression in $\mathbb{E}$ with only equality operators can be expressed as a set of conjunctive queries. The tableau for a conjunctive query is obtained by collecting, for each relation R_j, the arguments of each atomic formula for R_j into a table. We could generalize the set of formulas considered conjunctive queries by extending the allowable atomic formulas to include formulas of the form $(t_1 < t_2)$, again where the t's are variables or constants. We can collect the old atomic formulas into tables as before, and the new ones we can collect into a boolean matrix which has a row and a column for each variable and constant appearing in the query. The formal definitions follow.

The <u>transitive closure</u> of a binary relation R (in the mathematical sense), denoted R^*, is defined by the rules:

$$R \subseteq R^*$$
$$R^* \circ R \subseteq R^*$$

Here, 'o' is composition.

The set $\mathbb{V}$ of <u>variables</u> is the set $\{a_1,\ a_2,\ a_3,\ \ldots\ \}$ of subscripted "a"s. The set $\mathbb{Y}$ of <u>symbols</u> is $\mathbb{V} \cup \mathbb{N}$. We associate a <u>natural ordering</u> on $\mathbb{Y}$ as follows: $\mathbb{N}$ has its usual ordering; $\mathbb{V}$ is ordered by index value, and every element of $\mathbb{N}$ is less than every element of $\mathbb{V}$. A tableau T of degree m is an N+2-tuple $\langle S,B,T_1,\ldots,T_N\rangle$ such that $S \subseteq \mathbb{Y}^m$, for each $i=1,\ldots,N$, $T_i \subseteq \mathbb{Y}^{\deg(R_i)}$, every variable in S appears in some T_i, and B is a binary relation on the symbols in $T_1,\ldots T_N$. S is called the summary. The relation B can be considered to be a boolean matrix or a boolean valued function, and we will sometimes write $B(x,y)=1$ when xBy is true (or $(x,y) \in B$). We call B the LT-matrix (less-than matrix). We consider the empty tableau, $\langle\emptyset,\ldots,\emptyset\rangle$, to be a tableau of any degree. Note that summaries may contain many rows. As we will see, this is useful for testing expression constraints.

A tableau set of degree m is a finite set of tableaux of degree m.

If X is a tuple, a tuple set, a tableau or a tableau set, we let $\mathbb{Y}(X)$ denote the set of symbols occurring in X.

A valuation r for tableau T and instance $I = \langle D,O,I_1,\ldots,I_N\rangle$ is a function $\mathbb{Y}(T)\rightarrow D$ which is the identity on $\mathbb{N} \subseteq \mathbb{Y}$. Valuations can be extended to functions on tuples and functions on sets of tuples by component-wise and element-wise extension.

By using valuations, a tableau $T = \langle S,B,T_1,\ldots,T_N\rangle$ can be considered to be a function on instances. Given instance I, the value of T on I is defined by:

$$T(I) = \cup \{r(S) : r \text{ is a valuation for } T,\ r(T_i) \subseteq I_i,\ i=1,\ldots,N,\ r(B) \subseteq O\}$$

Example 2. Consider the tableau:

S	B	R_1	R_2
a_1 a_4	(a_2,a_3)	a_1 a_2	a_3 a_4
a_2 a_3		a_4 $\emptyset$	

The following table gives an instance and three valuations on it which satisfy the above conditions. (They are, in fact, the only valuations which do.) The corresponding valuations of the summary are also given:

R_1		R_2	
4	2	2	1
1	Ø	3	1
3	6	5	4

a_1	a_2	a_3	a_4	S	
4	2	3	1	4	1
				2	3
1	Ø	2	1	1	1
				Ø	2
1	Ø	3	1	1	1
				Ø	3

Given a tableau $T = \langle S,B,T_1,\ldots,T_N\rangle$, if B and the order on $\mathbb{N}$, $<_{\mathbb{N}}$, are union compatible, i.e., if $(B \cup <_{\mathbb{N}})^*$ is asymmetric, then T determines an instance $I = \langle D,O,I_1,\ldots,I_N\rangle$ by taking $D = \mathbb{N} \cup Y(T)$, $O = (B \cup <_{\mathbb{N}})^*$, and $I_i = T_i$, $i=1,\ldots,N$. Historically, I would be called the "Herbrand interpretation" [Herb] for T. We will often simply consider T itself to be the instance. The following lemma states an important but easily checked property of tableaux which relates their use as functions and their use as instances.

Lemma 1. If tableau $T = \langle S,B,T_1,\ldots,T_N\rangle$ can be considered an instance, then $S \subseteq T(T)$.

Proof. Let the identity function on $\mathbb{Y}(T)$ be the valuation. □

A tableau set $Y = \{T_1,\ldots,T_k\}$ may be considered to be a function by defining:

$$Y(I) = T_1(I) \cup \ldots \cup T_k(I).$$

We now define the concepts of "contained in" and "equivalent to" for tableaux.

Tableau T_1 is <u>contained in</u> tableau T_2 <u>with respect to</u> a set P of instances, written $T_1 \subseteq_P T_2$, if $T_1(I) \subseteq T_2(I)$ for all instances I in P. We write $T_1 \subseteq T_2$ if $T_1(I) \subseteq T_2(I)$ for all instances I. Tableau T_1 is <u>equivalent to</u> tableau T_2 on a set P of instances, written $T_1 \equiv_P T_2$, if $T_1(I) = T_2(I)$ for all instances I in P. Analogous definitions are made for tableau sets.

The above constitutes the basic definitions for tableaux. We now give (informally) a transformation from expressions to tableau sets. The tableaux generated contain only one row in their summary. Except for union, the rules are given for single tableaux; if the expression is represented by a tableau set, the rules are applied to each tableau in the set.

(1) The tableau for a schema relation R_i of degree m has a row $\langle a_1,\ldots,a_m\rangle$ in T_i, a row $\langle a_1,\ldots,a_m\rangle$ in the summary, and other components empty.

(2) The tableau for a constant relation {c} has c in the summary and other components empty.

(3) If T represents e, then a projection e[X] is represented by removing from the summary entries not in the columns of X.

(4) If T represents e, and symbols s_X, s_Y are in columns X and Y, respectively, of S, then the tableau T' for e[X=Y] is obtained as follows: If s_X and s_Y are the same symbol, then T' = T. If s_X and s_Y are unequal constants, then T' is the empty tableau. Otherwise assume that the symbol s_X precedes the symbol s_Y in the natural ordering. T' is obtained by replacing all occurrences of s_Y in T by s_X.

For a less-than-restriction, e[X<Y], T' is obtained from

T by adding the pair (s_X,s_Y) to B.

(5) If T_1,T_2 represent e_1,e_2, respectively, then the tableau T' to represent $e_1 \times e_2$ is obtained by making the sets of variables disjoint, making S' the cross product $S_1 \times S_2$, making B' the union $B_1 \cup B_2$ and making each other component T'_i the union $T_{1i} \cup T_{2i}$.

(6) A union $e_1 \cup e_2$ is repesented by the union of the tableau sets representing e_1 and e_2.

__Theorem 1__. Let Y be the tableau set resulting from expression e. Then $Y(I) = e(I)$ for all instances I.

Proof. Left to the reader. $\square$

__Example 3__.

$R_1[2<1]R_2)[1,4]$ -->

S	B	R_1	R_2
$a_1\ a_4$	(a_2,a_3)	$a_1\ a_2$	$a_3\ a_4$

__Example 4__.

$(R_1 \times R_2 \times R_1)[2=3\ \&\ 4=5\ \&\ 6<1\ \&\ 1<'8'][1,3,5]$ -->

S	B	R_1	R_2
$a_1\ a_2\ a_3$	(a_4,a_1)	$a_1\ a_2$	$a_2\ a_3$
	$(a_1,8)$	$a_3\ a_4$	

The existence of the reverse transformation is stated by the next theorem.

__Theorem 2__. For every tableau set Y there is an expression $e \in \mathbb{E}$ such that $e \equiv Y$.

Proof. The proof should generate equi-restrictions from pairs of occurrences of the same variable, less-than-restrictions from pairs in

B, and selections from occurrences of constants. □

Example 5. Consider the tableau:

$$
\begin{array}{ccc}
\multicolumn{3}{c}{S} \\ \hline
a_1 & 2 & a_3 \\
 & &
\end{array}
\qquad
\begin{array}{c}
B \\ \hline
(a_2, a_3) \\
(a_4, 1\ )
\end{array}
\qquad
\begin{array}{cc}
\multicolumn{2}{c}{R_1} \\ \hline
a_1 & a_2 \\
a_4 & a_1
\end{array}
\qquad
\begin{array}{ccc}
\multicolumn{3}{c}{R_3} \\ \hline
a_1 & \emptyset & a_3 \\
a_3 & a_2 & a_5
\end{array}
$$

The corresponding expression is (with domain numbers labeled for clarity):

$$
\begin{array}{ccccccccccl}
(R_1 & \times & R_1 & \times & R_3 & \times & R_3 & \times & \{2\}) & & [\ 2 < 7\ \&\ 3 < \text{'1'}\ \& \\
 & & & & & & & & & & \ \ 1 = 4\ \&\ 1 = 5\ \ \ \& \\
1\ 2 & & 3\ 4 & & 5\ 6\ 7 & & 8\ 9\ 10 & & 11 & & \ \ 4 = 5\ \&\ 2 = 9\ \ \ \& \\
 & & & & & & & & & & \ \ 7 = 8\ \&\ 6 = \text{'}\emptyset\text{'}] \\
 & & & & & & & & & & [1,11,7]
\end{array}
$$

4. Transformation Rules and Chases

Chases for "equality" tableaux were originally used in [AhBU] for testing for lossy joins. Two transformation rules (F-rules and J-rules) are made explicit in [MaMS] where chases are used to determine closures of sets of dependencies on single relations. In [KlPr], T-rules were added in order to test constraints on expressions. In this section, the chase computation for inequality tableaux is defined.

A chase consists of a sequence of transformations on a tableau which preserves equivalence with respect to a given class of instances. Formally, a transformation rule for a set P of instances is a partial function f on tableaux such that $f(T) \equiv_P T$. We define four classes of transformation rules. The first two correspond to schema FDs and JDs, respectively, which are valid in the given class P of instances. The third type of transformation rule adds rows to the summary when this will not change the value of the tableau, and these rules are applicable in any set of instances. The fourth rule forms a closure of the LT-matrix.

The rules for changing the LT-matrix need to infer all "less-than" relationships among symbols of a tableau. Since it is possible to have, say, pairs $(a_1,3)$ and $(4,a_2)$ in the LT-matrix without $(3,4)$ being present, we need to include the order on natural numbers in these rules. Let $\mathbb{N}(T)$ be the constants appearing in tableau $T = \langle S,B,T_1,\ldots,T_N\rangle$, and let $<_{\mathbb{N}(T)}$ be $<_{\mathbb{N}} \sqcap (\mathbb{N}(T)\times\mathbb{N}(T))$. Then the $^{+}$-closure of B, written B^{+}, is the set of order pairs $(B \cup <_{\mathbb{N}(T)})^{*}$. This is a closure of B with the ordering on the constants taken into account.

F-Rules. For each schema FD $R_i{:}Z\rightarrow A$ there is an F-rule which is defined as follows: If $T = \langle S,B,T_1,\ldots,T_N\rangle$ and there are t_1, $t_2 \in T_i$ such that $t_1[Z] = t_2[Z]$ and $t_1[A] \neq t_2[A]$, then

(a) If $t_1[A]$ and $t_2[A]$ are unequal constants, replace T by the empty tableau.

(b) Otherwise, if they are unequal symbols s_1, s_2, and s_1 is less than s_2 in the natural orering, replace all the occurrences in T of s_2 by occurrences of s_1 where B is considered to be a set of ordered pairs. (If we consider B a matrix this means OR-ing the s_2-row into the s_1-row, OR-ing the s_2-column into the s_1-column, and removing the s_2-column and s_2-row.)

J-rules. For each JD $R_i{:}[X_1,\ldots,X_k]$ there is a J-rule which is defined as follows: If there is an element $t \in \mathbb{Y}^{\deg(R_i)}$ such that for each $j=1,\ldots,k$ there is a $t_j \in T_i$ with $t_j[X_j] = t[X_j]$, then add t to T_i if it is not already there.

T-rules. If r is a function $\mathbb{Y}(T)\rightarrow\mathbb{Y}(T)$ such that $r(T_i) \subseteq T_i$ for $i=1,\ldots,N$, and $r(B) \subseteq B$, then add $r(S)$ to S if not already there.

LT-Rules. If $T = \langle S,B,T_1,\ldots,T_N\rangle$ and $B \neq B^+$, replace T by the empty tableau if B^+ has a non-zero diagonal. Otherwise replace T by $\langle S,B^+,T_1,\ldots,T_N\rangle$.

Example 6. In Figure 1 some examples applying these rules are given.

We note in the next lemma that applying any of these rules results in an equivalent tableau.

Lemma 2. Let T be a tableau.

(1) Let T' be the result of applying the F-rule for $R_i:Z\rightarrow A$ to T. Then $T \equiv T'$ wrt $sat(R_i:Z\rightarrow A)$.

(2) Let T' be the result of applying the J-rule for $\langle R_i,S\rangle$ to T. Then $T \equiv T'$ wrt $sat(R_i:S)$.

(3) Let T' be the result of applying the T-rule for the function r to T. Then $T \equiv T'$.

(4) Let T' be the result of applying the LT-rule to T. Then $T \equiv T'$.

Proof. Proofs similar to what is needed for (1), (2) and (3) can be found elsewhere [MaMS] [KlPr]. For (4), note that if $I = \langle D,O,I_1,\ldots,I_N\rangle$ is an instance and r is a valuation for T on I such that $r(B) \subseteq O$, then $r(B^+) \subseteq O$. □

Lemma 3. A given set of F-, J-, T- and LT-rules can be applied to a tableau only a finite number of times.

Proof. As in [MaMS]. The LT-rule can be applied only at the beginning of a chase or after some other rule, and these other rules can only be applied a finite number of times. □

Given a tableau T we may apply the transformation rules to T to obtain a sequence $T_1,T_2,T_3,\ldots$ of tableaux all equivalent to T.

S		
a_1	a_2	a_4

R_3		
a_1	a_2	a_3
a_1	a_4	a_5

B
$(\emptyset,a_4)$
$(a_1,3)$

$$\xrightarrow{\text{F-rule } R_3:1\text{->}2}$$

S		
a_1	a_2	a_2

R_3		
a_1	a_2	a_3
a_1	a_2	a_5

B
$(\emptyset,a_2)$
$(a_1,3)$

S		
a_1	a_2	a_3

R_3		
a_1	a_2	a_3
a_4	a_2	a_5

B
(a_1,a_2)

$$\xrightarrow{\text{J-rule } [\{1,2\},\ \{2,3\}]}$$

S		
a_1	a_2	a_3

R_3		
a_1	a_2	a_3
a_4	a_2	a_5
a_1	a_2	a_5

B
(a_1,a_2)

S		
a_3	a_1	a_2

R_3		
a_1	a_2	a_3
a_1	a_3	a_2

B
(a_2,a_1)
(a_3,a_1)

$$\xrightarrow{\text{T-rule } a_2 \to a_3,\ a_3 \to a_2}$$

S		
a_3	a_1	a_2
a_2	a_1	a_3

R_3		
a_1	a_2	a_3
a_1	a_3	a_2

B
(a_2,a_1)
(a_3,a_1)

S		
a_1	a_2	a_4

R_3		
a_1	a_2	a_3
a_1	a_4	a_5

B
$(5,a_4)$
$(a_1,3)$

$$\xrightarrow{\text{LT-rule } B \text{ --> } B+}$$

S		
a_1	a_2	a_2

R_3		
a_1	a_2	a_3
a_1	a_2	a_5

B
$(5,a_2)$
$(a_1,3)$
(a_1,a_2)
$(3,5)$

Figure 1. Transformation Examples

Eventually there will be no applicable rules, and the sequence will terminate. That the final tableau in such a sequence is unique follows from the next theorem:

<u>Theorem</u> <u>3</u>. The transformations rules above have the Church-Rosser property [Seth]. That is, if T' and T" are the results of applying two transformation rules to T, then there exist sequences $T'_1, T'_1, \ldots, T'_k$ and $T''_1, \ldots, T''_m$ where each tableau in the sequence is obtained from the previous element by a transformation rule, where $T'_1 = T'$, $T''_1 = T''$, and where $T'_k = T''_m$. (See Figure 2.)

Proof. Left to reader. □

Thus the following <u>chase</u> function is well-defined [Seth]: Given a set C of schema constraints and a tableau T, $\text{chase}_C(T)$ is the tableau resulting from applying all possible transformation rules for C to T.

Some important properties of the chase are given in the following theorem.

$$\begin{tikzcd}
& T \arrow[dl] \arrow[dr] & \\
T' \arrow[dr, dashed] & & T'' \arrow[dl, dashed] \\
& T'_k = T''_m &
\end{tikzcd}$$

Figure 2. Church-Rosser Property for the Chase.

Theorem 4. (1) $\text{chase}_C(T)$, considered an instance, satisfies the constraints in C.

(2) $T \equiv \text{chase}_C(T)$ wrt C.

If $T' = \text{chase}_C(T)$, then $S' = T'(T')$, where S' is the summary of T'; the first occurrence of T' is the function on instances, and the second occurrence is T' the instance.

Proof. (1) F- and J-rules can only be applied when the tableau, considered an instance, violates the corresponding constraint. (2) This follows from Theorem 2.

(3) The inclusion "$S' \subseteq T'(T')$" has already been shown. To prove the other inclusion, suppose $x \in T'(T')$. There is a valuation p such that $x = p(s)$ for some $s \in S'$, $p(B) \subseteq B^+$ and $p(T'_i) \subseteq T'_i$ $(i=1,\ldots,N)$. Since T' is at the end of a chase, $B^+ = B$, so the T-rule for p is applicable to T', but since T' is already the end of a chase, we must have $p(S') \subseteq S'$. In particular, $x = p(s) \in S'$. □

The last point we make in this section is that T-rules can be replaced by a possibly simpler algebraic computation:

Theorem 5. Let T be a tableau and e an equivalent expression. Let $T' = \text{chase}_C(T)$ for some constraint set C. Define the function chase^* as the chase function without the T-rules, and let $T^* = \text{chase}^*_C(T)$. Then $S' = e(T^*)$, where T^* is considered an instance.

Proof. Left to the reader. □

5. Testing Expression Constraints

Having introduced inequality tableaux and the chase function for inequality tableaux, we now show how these concepts can be used to determine valid expression constraints given the validity of certain schema constraints. This problems is important because, for example, application programs using views defined by relational algebra expressions may depend on certain constraints holding. Knowing that view constraints follow from schema constraints also means that the views do not need to be materialized in order to check their consistency.

This problem was solved in [KlPr] for expressions containing only equality comparisons. The solution can be generalized to apply to inequality tableaux, and we give in this section some examples of its use.

The motivation for the method we use is Theorem 4. This theorem says that chasing a tableau makes the tableau, as an instance, satisfy the constraints used in the chase. Now we are looking at constraints which might hold in the value set of a tableau, so we want to chase a certain tableau and try to make the summary satisfy the expression constraint. We will need to be able to generate a representative tableau for the given expression which has a summary of a particular form. We arrange that the summary, as a tuple set, will violate the test constraint. If the chase removes this violation in the summary, the constraint must be valid. Otherwise we will have a counterexample state.

Given expression e (without unions) and equivalent tableau T with a one row summary, we replicate the tableau with brand new symbols in some places and old symbols in others. The symbol renaming functions used to replicate the tableau must be compatible. An allowable or compatible set $r_1,\ldots,r_k$ of renamings is a set of one-to-one functions on the variables of T such that for each r_i, r_i may be the identity on some symbols in the summary, but it must otherwise map variables to variables not used by any other r_j. By convention, we also consider the identity renaming to be compatible with any set of compatible renamings.

Example 7. Let T be the tableau:

S	B	R_1	R_2
$a_1\ a_2\ a_3$	(a_4,a_1)	$a_1\ a_2$	$a_2\ a_3$
	$(a_1,8)$	$a_3\ a_4$	

Take r_1 to be the identity, and r_2 and r_3 to be defined by the rules:

r_1	r_2
$a_1 \rightarrow a_1$	$a_1 \rightarrow a_7$
$a_2 \rightarrow a_2$	$a_2 \rightarrow a_8$
$a_3 \rightarrow a_5$	$a_3 \rightarrow a_3$
$a_4 \rightarrow a_6$	$a_4 \rightarrow a_9$

Then $r_1(T)$ U $r_2(T)$ U $r_3(T)$ is the tableau:

S	B	R_1	R_2
$a_1\ a_2\ a_3$	(a_4,a_1)	$a_1\ a_2$	$a_2\ a_3$
$a_1\ a_2\ a_5$	$(a_1,8)$	$a_3\ a_4$	$a_2\ a_5$
$a_7\ a_8\ a_3$	(a_6,a_1)	$a_5\ a_6$	$a_8\ a_3$
	(a_4,a_9)	$a_7\ a_8$	
	$(a_7,8)$	$a_3\ a_9$	

The new tableau is equivalent to the original. This is because no new "connections" have been made among the rows making up the tableau.

We will illustrate with some examples how this replication of tableaux along with the chase procedure of the last section are used to determine valid expression constraints.

Example 8. Consider the expression:

$$((R_3[2='\emptyset'])[3=1]R_2)[1,5]$$

The tableau for this expression is:

S	B	R_3	R_2
a_1 a_2		a_1 $\emptyset$ a_3	a_3 a_2

Given schema FDs R_3:1,2→3 and R_2:1→2, we want to test the FD 1→2 on the expression. We replicate the tableau so that the FD 1→2 is false in the summary. The tableau generated is:

S	B	R_3	R_2
a_1 a_2		a_1 $\emptyset$ a_3	a_3 a_2
a_1 a_4		a_1 $\emptyset$ a_5	a_5 a_4

We can apply the F-rule for R_3:1,2→3 and get:

S	B	R_3	R_2
a_1 a_2		a_1 $\emptyset$ a_3	a_3 a_2
a_1 a_4		a_1 $\emptyset$ a_3	a_3 a_4

The F-rule for R_2:1→2 is now applicable, and the result is:

S	B	R_3	R_2
a_1 a_2		a_1 $\emptyset$ a_3	a_3 a_2
a_1 a_2		a_1 $\emptyset$ a_3	a_3 a_2

The FD 1→2 is true in S, and so it is valid on the expression.

<u>Example 9</u>. Consider the expression:

$$(R_1 \times R_2 \times R_3)\ [2 = 3\ \&\ 4 < 5]\ [1,6]$$

Assume there are FDs R_1:1→2, R_2:1→2 and R_3:1→2. The tableau for this expression is:

S	B	R_1	R_2	R_3
a_1 a_2	(a_4,a_5)	a_1 a_3	a_3 a_4	a_5 a_2 a_6

To test the FD 1→2 on the expression, we generate the tableau:

S	B	R_1	R_2	R_3
$a_1\ a_2$	(a_4,a_5)	$a_1\ a_3$	$a_3\ a_4$	$a_5\ a_2\ a_6$
$a_1\ a_7$	(a_8,a_{10})	$a_1\ a_9$	$a_9\ a_8$	$a_{10}\ a_7\ a_6$

We can apply the rule for the FD R_1:$1 \rightarrow 2$, equating a_9 to a_3, and then the rule for the FD R_2:$1 \rightarrow 2$, equating a_8 to a_4. The result will be:

S	B	R_1	R_2	R_3
$a_1\ a_2$	(a_4,a_5)	$a_1\ a_3$	$a_3\ a_4$	$a_5\ a_2\ a_6$
$a_1\ a_7$	(a_4,a_{10})	$a_1\ a_3$	$a_3\ a_4$	$a_{10}\ a_7\ a_6$

No other rules are applicable. We have not identified a_2 and a_7. The test FD is false in the summary, and this tableau can be taken as a counterexample state: The FDs R_1:$1 \rightarrow 2$, R_2:$1 \rightarrow 2$ and R_3:1_2 are true in this (formal) instance, and the tuples $\langle a_1,a_2\rangle$ and $\langle a_1,a_7\rangle$ are in the value of the expression on this instance.

<u>Example 10</u>. Consider the JD [{1,2},{1,3},{2,3}] on the expression:

$$(R_1\ [2<1]\ R_2)\ [4=1]\ R_3$$

The tableau for this expression is:

S	B	R_1	R_2	R_3
$a_1\ a_2\ a_3$	(a_4,a_2)	$a_1\ a_4$	$a_2\ a_3$	$a_3\ a_1$

A tableau for testing the JD is (we will look for $\langle a_1,a_2,a_3\rangle$ in S):

S	B	R_1	R_2	R_3
$a_1\ a_2\ a_5$	(a_4,a_2)	$a_1\ a_4$	$a_2\ a_5$	$a_5\ a_1$
$a_9\ a_2\ a_3$	(a_6,a_2)	$a_9\ a_6$	$a_2\ a_3$	$a_3\ a_9$
$a_1\ a_7\ a_3$	(a_8,a_7)	$a_1\ a_8$	$a_7\ a_3$	$a_3\ a_1$

We can apply the T-rule with any one of the three functions:

(a) $a_5 \rightarrow a_3$, others unchanged
(b) $a_9 \rightarrow a_1$, $a_6 \rightarrow a_4$, others unchanged
(c) $a_7 \rightarrow a_2$, $a_8 \rightarrow a_4$, others unchanged

In each case, $\langle a_1,a_2,a_3\rangle$ will appear in S. Therefore the JD is valid.

Although we can check the validity of FDs on expressions having '<' comparisons, example 8 had only '=' comparisons. There is a simple reason for this:

Theorem 6. If an expression e which is non-empty on at least one instance contains no unions, then an FD is valid on e if and only if it is valid on the expression obtained from e by deleting all less-than restrictions.

Proof. Let T be a tableau for e which has two summary rows for testing an FD, and let e' be e with all less-than restrictions removed. Note that T-rules are not needed to test FDs on expressions, and the F- and J-rules are applicable regardless of the LT-rule being used. Hence, we may postpone the LT-rule until the last step in the chase. So let $T_1,\ldots,T_n$ be a chase for T in which T_n was derived from T_{n-1} by the only application in the chase of the LT-rule. If we let $T'_1,\ldots,T'_{n-1}$ be derived from $T_1,\ldots,T_{n-1}$ by making all LT-matrices empty, then T'_1 is a tableau for e', and the sequence is a chase for T'_1. Finally, note that the LT-rule will not replace T_{n-1} by the empty tableau and that the summary of T'_{n-1} is the same as the summary of T_n. □

This theorem fails if the expression contains unions:

Example 11. Suppose relation R_1 has an FD R_1:$1\rightarrow 2$. Let e be $R_1 \cup \{\langle 1,5\rangle\}$. Then the FD $1\rightarrow 2$ is not valid on e, but it is valid on e[2 < '4'] (verified below).

The above techniques can be extended to handle FDs (and JDs) on expressions containing unions:

If expression e is a union $e_1 \cup \ldots \cup e_n$, an FD $Z \rightarrow A$ can fail to hold on on e(I) (for some I) if there is a pair of tuples $t_i \in e_i(I)$ and $t_j \in e_j(I)$, i and j not necessarily distinct, which have the same Z-values but different A-values. Hence we test all tableaux generated by concatenating pairs T_i for e_i and T_j for e_j such that the symbols in the Z-columns of the summary are the same.

<u>Example</u> <u>12</u>. Consider the previous expression:

$$(R_1 \cup \{<1,5>\})[2 < '4']$$

We assume R_1 has an FD $1 \rightarrow 2$. To check the FD $1 \rightarrow 2$ on the expression, we need to check three tableaux. We will only show the case for one tuple out of R_1 and the other out of $\{<1,5>\}$. The tableau is: (When the symbols in column of the summary are equated, the variable must be replaced by the constant, not the constant by the variable.)

S		B	R_1	
1	a2	(a2,4)	1	a2
1	5	(5, 4)		

The application of the LT-rule results in an empty tableau since (5,4) from B and (4,5) from $<_{\mathbb{N}}$ yields a diagonal element (5,5). An FD is vacuously true in an empty instance.

Other types of constraints on expressions (views) can also be tested with this method. In [KlPr] a procedure is given for testing "pseudo-keys", sets of domains, one of which must uniquely identify each tuple in a tuple set. For example, a pseudo-key {1},{2} is true in a tuple set S if for all distinct t_1, t_2 in S, if $t_1[1] = t_2[1]$ then $t_1[2] \neq t_2[2]$. To test such a constraint on an expression, we generate a tableau whose summary has two distinct rows with columns 1 and 2 the same. We then see if the chase identifies the two rows.

6. Equivalence and Optimization

It has been shown in [ChMe] and [AhSU] that the "contained in" relation for "equality" tableaux and "equality" tableau sets can be determined by certain row-preserving functions on symbols. These so called "containment mappings" can also be used to optimize expressions. In this section consider the proper "row-preserving" functions which can determine "equivalence" and "contained in" for inequality tableaux[3].

A containment mapping f from tableau T_1 to tableau T_2 is a function $\mathbb{V}(T_1) \to \mathbb{V}(T_2)$ (considered the identity on constants), which is one-to-one from the summary of T_1 onto the summary of T_2, and which has the properties that $f(B_1) \subseteq (B_2 \cup <_{\mathbb{N}})^{*}$[4], and $f(T_{1i}) \subseteq T_{2i}$ for $i=1,\dots,N$.

Example 13. We should not replace the condition:

$$f(B_1) \subseteq (B_2 \cup <_{\mathbb{N}})^{*}$$

by either $f(B_1) \subseteq B_2$ or even $f(B_1) \subseteq B_2^{+}$. Consider the following two tableaux:

T1: S	B	R_1	T2: S	B	R_1
a_2	a_1 3	a_1 a_2	a_4	a_3 4	a_3 a_4

Clearly, $T_1 \subseteq T_2$, but the function sending a_1 to a_3 and a_2 to a_4 does not send B_1 to B_2 or to B_2^{+}.

[3] We only consider tableaux with single row summaries. A tableau with n rows in its summary is equivalent to a set of n tableaux with single row summaries.

[4] Although the set $(B_2 \cup <_{\mathbb{N}})^{*}$ is infinite, the relation $f(B_1) \subseteq (B_2 \cup <_{\mathbb{N}})^{*}$ is easily decidable.

Containment mappings are similar to algebraic homomorphisms, and the composition of two containment mappings is also a containment mapping:

<u>Theorem 7</u>. Let $f_1:T_1 \rightarrow T_2$ and $f_2:T_2 \rightarrow T_3$ be containment mappings. Then the functional composition $f_2 \circ f_1$ is a containment mapping $T_1 \rightarrow T_3$.

Proof. Clearly, $f_2(f_1(T_{1i})) \subseteq f_2(T_{2i}) \subseteq T_{3i}$. For the LT-matrix, we have $f_2(f_1(B_1)) \subseteq f_2((B_2 \cup <_{\mathbb{N}})^*)$. We only need to show that $f_2((B_2 \cup <_{\mathbb{N}})^*) \subseteq (B_3 \cup <_{\mathbb{N}})^*$. We use induction on the transitive closure. If x is in B_2, then $f_2(x)$ is in $B_3 \cup (<_{\mathbb{N}})^*$, because f_2 is a containment mapping. If x is in $<_{\mathbb{N}}$, then $f(x) = x$, which is in $B_3 \cup <_{\mathbb{N}}$. If $<a,b>$ and $<b,c>$ are in $(B_2 \cup <_{\mathbb{N}})^*$, then $f(<a,b>) = <f(a),f(b)>$ and $f(<b,c>) = <f(b),f(c)>$ are in $(B_3 \cup <_{\mathbb{N}})^*$. Hence, $f(<a,c>) = <f(a),f(c)>$ is in $(B_3 \cup <_{\mathbb{N}})^*$. □

Note that there are only a finite number of possible containment mappings from one tableau to another. The first theorem of this section relates the "contained in" property, which involves an infinite number of instances, to the finite set of possible containment mappings from one tableau to the other.

<u>Theorem 8</u>. Let T_1, T_2 be tableaux.

(1) $T_1 \subseteq T_2$ if and only if T_1 is equivalent to the empty tableau, or there is a containment mapping $f:T_2 \rightarrow T_1$.

(2) If T_1 is in sat(P) (considered as an instance), then $T_1 \subseteq_P T_2$ if and only if there is a containment mapping $f:T_2 \rightarrow T_1$.

Proof. Similar to corresponding proofs in [ChMe] and [AhSU]. □

<u>Corollary</u>. For any tableaux T_1, T_2, and any set P of constraints,

$T_1 \subseteq_P T_2$ if and only if $\text{chase}_P(T_1)$ is empty, or there is a containment mapping $f{:}T_2 \rightarrow \text{chase}_P(T_1)$.

Corollary. If T is a tableau and T' is a tableau obtained from T by removing rows from some T_i, then $T \equiv T'$ if and only if there is a containment mapping $f{:}T \rightarrow T'$.

Proof. This follows from Theorem 8 and the fact that we will always have $T \subseteq T'$ for such T and T'. □

These results are easily extended to tableau sets:

Theorem 9. Let Y_1 and Y_2 be tableau sets.

(1) $Y_1 \subseteq Y_2$ if and only if there is a containment mapping from each element of Y_2 to some element of Y_1.

(2) $Y_1 \subseteq_P Y_2$ if and only if the chases of the tableaux in Y_1 are all empty, or there is a containment mapping from each element of Y_2 to the chase wrt P of some element of Y_1.

Proof. Similar to Theorem 2 of [SaYa]. □

The join operation, whether an equi-join or a less-than-join, is the most expensive operation in evaluating a query. Since the number of joins in an expression (without union) is one less than the number of rows in the corresponding tableau, optimization of a query corresponds to removing as many rows as possible from a tableau [ChMe] [AhSU]. We can optimize an expression, that is, remove rows from a tableau by the following process: Given tableau T find a containment mapping $f{:}T \rightarrow T$ where for some i, $f(T_i)$ is a proper subset of T_i. Now find another proper containment mapping $f'{:}f(T) \rightarrow f(T)$. Since tableaux are finite, this process can be repeated only a finite number of times. If T' is the final tableau, the only containment mappings

T'→T' will be one-to-one and onto. This means will not be able to remove any more rows from T' and still maintain equivalence.

Example 14. Consider the expression:

$$(((R_1[2<'3']\ [2=1]\ R_3[3='Ø'])\ [1=1]\ (R_1[2<'4']\ [1=2]\ R_3))$$
$$[6>1]\ (R_1\ [1<2]\ R_3))\ [15=5]\ (R_1\ [1<2]\ R_3)$$

The tableau for this expression is:

S	B	R_1	R_3
a_1 a_3	$(a_2, 3)$	a_1 a_2	a_2 a_3 Ø
	$(a_4, 4)$	a_1 a_4	a_4 a_5 a_6
	(a_8, a_9)	a_7 a_8	a_9 a_{10} a_{11}
	(a_{13}, a_{14})	a_{12} a_{13}	a_{14} a_{15} a_{11}
	(a_7, a_1)		

The following symbol mapping is a containment mapping:

a_1	→	a_1	a_8	→	a_8
a_2	→	a_2	a_9	→	a_9
a_3	→	a_3	a_{10}	→	a_{10}
a_4	→	a_2	a_{11}	→	a_{11}
a_5	→	a_3	a_{12}	→	a_7
a_6	→	Ø	a_{13}	→	a_8
a_7	→	a_7	a_{14}	→	a_9
			a_{15}	→	a_{10}

It produces the tableau:

S	B	R_1	R_3
a_1 a_3	$(a_2, 3)$	a_1 a_2	a_2 a_3 Ø
	(a_8, a_9)	a_7 a_8	a_9 a_{10} a_{11}
	(a_7, a_1)		

This tableau is minimal and it corresponds to the expression:

$$(R_1[2<'3']\ [2=1]\ R_3[3='Ø'])\ [1>1]\ (R_1\ [2<1]\ R_3)$$

7. Summary and Conclusions

In this paper it was shown how general relational algebra expressions which may include less-than comparison between domains and between a domain and a constant can be represented tabular forms called inequality tableaux. The full relational algebra as described, for example by Codd [Codd], can be represented. (Set difference is not represented, but including this operator in the algebra would make the problems considered in this paper undecidable.) We used this representation to check view constraints, to check equivalence of expressions, and to optimize expressions.

Tableaux have been shown by a number of authors to be quite useful devices. In addition to the applications of them given in this paper, tableaux have also been used to check the equivalence of schemas [BMSU], to evaluate joins in polynomial time [Hone], to investigate semi-joins [BeGo1], to check for necessary schema constraints in schema design [KlPr] and to represent locks in a generalization of predicate locks [EGLT] [Klug]. By widening the class of relational algebra expressions which tableaux can represent, all of the previous uses of them become more valuable.

8. References

[AhBU] Aho A.V.,Beeri C. and Ullman J.D. "The Theory of Joins in Relational Databases" ACM-TODS 4, 297-314 (1979)

[AhSU78] Aho A.V., Sagiv Y. and Ullman J.D. "Efficient Optimization of a Class of Relational Expressions", ACM TODS, 4, 435-454 (1979)

[AhSu79] Aho A.V. Sagiv Y. and Ullman J.D. "Equivalence of Relational Expressions" SIAM J. Comptng. 8, 2, 218-246 (May 1979)

[BeGo1] Bernstein P.A. and Goodman N. "The Theory of Semi-Joins", Tech. Rep. CCA-79-27, 1979, Computer Corp. of America

[BeGo2] Bernstein P.A. and Goodman N. "What does Boyce-Codd normal form do?" Proc. 6-th Int. Conf. on Very Large Data Bases, Montreal 198Ø

[BMSU] Beeri C., Mendelzon A.O., Sagiv Y. and J.D. Ullman "Equivalence of relational database schemes" Proc. 11-th Ann. ACM Symp. on the Theory of Computing, 1979

[ChMe] Chandra A.K. and Merlin P.M. "Optimal Implementation of Conjunctive Queries in Relational Databases", Proc. 9-th Annual Symp. on Theory of Computing, May, 1976, 77-90

[Codd] Codd E.F. "Relational Completeness of Data Base Sublanguages" Data Base Systems, R. Rustin (ed.), Prentice Hall, 1972

[EGLT] Eswaran K.P., Gray J.N., Lorie R.A. and Traiger I.L. "The Notions of Consistency and Predicate Locks in a Database System" CACM 19, 11, pp.624-633

[Ende] Enderton H.B. "A Mathematical Introduction to Logic", Academic Press 1972

[Herb] Herbrand J. "Recherches sur la theorie de la demonstration" Trav. Soc. Sci. Lett. Varsovie, cl. III, 33

[Hone] Honeyman P. "Extension Joins" Proc. 6-th Int. Conf. on Very Large Data Bases, 1980

[KlPr] Klug A. and Price R. "Generalized Tableaux for Chasing Expression Constraints" to appear

[Klug] Klug A. "Locking expressions for increased database concurrency" to appear

[Korf] Korfhage R.R. Discrete Computational Structures, Academic Press, 1974

[MaMS] Maier D., Mendelzon A.O. and Sagiv Y. "Testing Implications of Data Dependencies" ACM-TODS 4, 4, 455-469 (1979)

[SaYa] Sagiv Y. and Yannakakis M. "Equivalence Among Relational Expressions with the Union and Difference Operator" Fourth International Conference on Very Large Data Bases, West-Berlin, 1978

[Schm] Schmidt J.W. "Some High-Level Constructs for Data of Type Relation" ACM TODS 2, 3, pp.247-261

[Seth] Sethi R. "Testing for the Church-Rosser property" JACM, 21, pp. 671-679 (1974)