ON THE CONNECTIONS BETWEEN RELATIONAL TABLEAUX AND CLAUSE FORM

Pedro Manoel Silveira

NCE 0487

Abril, 1987

Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica
Caixa Postal 2324
20001 - Rio de Janeiro, RJ
BRASIL
On the Connections Between
Relational Tableaux and Clause Form

Pedro Manoel Silveira

ABSTRACT

The theory of tableaux for relational databases is applied to the test of containment among relational expressions. In this paper we analyse the connections between tableaux and clause form representation. We show that results from the theory of tableaux are also obtainable with clause form.

SUMÁRIO

A teoria dos tableaux para bancos de dados relacionais aplica-se ao teste de equivalência entre expressões da Álgebra Relacional. Neste artigo nós analisamos as conexões entre tableaux e a representação por cláusulas. Nós mostramos que resultados da teoria dos tableaux podem também ser obtidos com cláusulas.
1. Introduction

The test of equivalence of queries is a basic step towards the obtention of methods for database query optimization. The main results in this topic seem to be the ones published in [1, 7, 9], where the authors introduce and further extend the concept of relational tableaux. These provide a tabular set description of queries and methods have been developed to carry out containment tests among tableaux instances. Despite their obvious utility tableaux, as they stand, do not support the complete range of relational expressions. The main problems remain around the use of the difference operator and some other minor points [4].

This paper deals with the connections between relational tableaux and some techniques used in the manipulation of logical formulae. The relevance of this work is the exploration of new approaches for the complete solution of the problem of query containment for unrestricted relational expressions.

This paper is organized as follows. Section 2 and 3 contain brief reviews of relational tableaux and clause form representation. Section 4 presents concepts and definitions covering the subject and formally introduces the central arguments of our work. Section 5 discusses these results and hints further research on the subject.

2. Relational Tableaux

Tableaux were introduced by Aho, Sagiv and Ullman [1] and provide a tabular set description. A tableau is a matrix, where the first row is called summary and the remaining rows are called rows. The summary represents the target list of a set expression and the rows represent the tuples required to be in the universe I, supposing there is a relation R on the set of all attributes whose instance is I. The columns of a tableau correspond to the attributes of R in a fixed order. Figure 1 shows examples of tableaux, with their corresponding set expressions.
Figure 1. Examples of Relational Tableaux

The first two examples correspond to relation schemata $AB$ and $BC$ respectively, and the third to the natural join of $AB$ and $BC$. In the example, $x, y, z$ are said to be distinguished variables, because they appear in the target list of the corresponding set expressions, while $v_1$ and $v_2$ are said to be nondistinguished variables, because they appear only on the right side of the bar. Constants and blank symbols can also appear in a tableau.

$T(I)$ is a relation defined on the attributes whose columns in the summary are non-blanks, and is defined as:

$$T(I) = \{ \rho(w_i) \mid \text{for some valuation } \rho, \rho(w_i) \in I, 1 \leq i \leq n \}$$
A valuation \( \rho \) associates with each symbol of \( \mathbf{T} \) a constant \( c \). For a row \( w_i \), \( \rho(w_i) \) is the tuple obtained by substituting

i) \( \rho(\psi) \) for every variable \( \psi \) appearing in \( w_i \);

ii) \( \rho(c) = c \) for every constant \( c \) appearing in \( w_i \).

3. Clause Form

Clause form [6] is a normalized representation of logical formulae. In general, any Predicate Calculus formula can be represented by a conjunction of clauses, where each clause contains only the disjunction of literals. The following transformations are necessary to translate a formula to its clause form representation.

a) the elimination of implication signs, replacing

\[ \psi_1 \rightarrow \psi_2 \text{ by } \neg \psi_1 \lor \psi_2 \]

b) the reduction of the scopes of the negation symbols, replacing

\[
\begin{align*}
\neg \neg \phi & \text{ by } \phi \\
\neg (\psi_1 \land \psi_2) & \text{ by } \neg \psi_1 \lor \neg \psi_2 \\
\neg (\psi_1 \lor \psi_2) & \text{ by } \neg \psi_1 \land \neg \psi_2 \\
\neg \exists \phi & \text{ by } \forall \neg \phi \\
\neg \forall \phi & \text{ by } \exists \neg \phi 
\end{align*}
\]

c) the renaming of variables, such that no two quantifiers apply to variables with identical names.

d) the elimination of existential quantifiers, through the introduction of Skolem functions. The idea is to replace the occurrence of existentially quantified variables by functions that represent elements in the domain of the original variables. These functions take as arguments the universally quantified variables whose scopes reach the existential quantifiers being removed.

e) the conversion to prenex normal form. A formula \( \phi \) is in prenex
normal form if and only if it has the form of a list of quantifiers followed by a quantifier free formula (called the matrix). At this stage, there can be only universal quantifiers and they can be simply moved to the front of the formula, following

\[(\forall x \varphi_1) \& \varphi_2 = \varphi(x, \varphi_1 \& \varphi_2)\]

and

\[(\forall x \varphi_1) \varphi_2 = \varphi(x, \varphi_1, \varphi_2)\]

\(f \, x \) is not free in \( \varphi_2 \).

f) the conversion of the matrix to conjunctive normal form. A formula is in conjunctive normal form when it is formed by a conjunction of clauses. This can be done by replacing

\[\varphi_1 (\varphi_2 \& \varphi_3) \text{ by } (\varphi_1 \varphi_2) \& (\varphi_1 \varphi_3)\]

g) the elimination of universal quantifiers. All variables remaining at this stage are universally quantified. Assuming that the variables in the matrix remain so, we can drop the universal quantifiers altogether.

h) the elimination of the logical AND connectives. This way we are left with a finite set of clauses, which are disjunctions in the matrix resulting from step g.

A method for translating a query to clause form is now shown. Let \( Q \) be a query of the form

\[t_1, \ldots, t_n \quad v_1 \in D_1, \ldots, v_k \in D_k \quad \varphi(v_1, \ldots, v_n)\]

Taking \( Q \) as a hypothetical relation containing all and only the tuples forming the answer to a query \( Q \) in the format above, we can say that

\[\forall v_1 \in D_1 \ldots \forall v_k \in D_k \\
\varphi(v_1, \ldots, v_n) \rightarrow Q(t_1, \ldots, t_n)\]  \( (1)\)

Translating formula 1 into clause form, as indicated above, we
have a set of clauses of the form

\[ Q(t_1, \ldots, t_n) \land \psi \]

where \( \psi \) is a possibly empty disjunction of literals, \( t_i \) is a term and all variables in each clause are universally quantified over their respective domains.

4. Clause Form and Relational Tableaux

We now show the connection between clause form representation and the theory of tableaux for relational databases.

To represent tableaux we use a very simple method. Taking \( Q \) as the virtual relation containing all and only the tuples in the answer for a query \( Q \), we can write a clause

\[ Q(\text{summary}) \land \neg I(\text{row}_1) \land \neg I(\text{row}_n) \]

for a tableau with summary and \( n \) rows. This can be verifiable from the definitions for tableaux. Notice that each term in the literal in \( Q \) must have a specified domain. In a tableau, the elements of the summary have their domains positionally determined. In the translation to clause form, it is necessary to indicate the attribute in \( R \) corresponding to each term in the summary. So, for the third tableau in Figure 1 we would have the clause

\[ Q(x, y, z) \land \neg I(x, y, v_x) \land \neg I(v_x, y, z) \]

Before proceeding, we must introduce some concepts for tableaux and clause manipulation, borrowing material from [1, 2, 6].

Definition. Let \( T_1 \) and \( T_2 \) be tableaux, whose sets of symbols are \( E_1 \), \( E_2 \) respectively. A homomorphism is a mapping \( m: E_1 \rightarrow E_2 \), such that

a) if \( c \) is a constant, \( m(c) = c \);

b) if \( v \) is a distinguished variable, then \( m(v) \) is
either a distinguished variable or a constant; 
c) if \( w \) is any row of \( T_1 \), then \( m(w) \) is a row of \( T_2 \).

**Definition.** Let \( T_1 \) and \( T_2 \) be tableaux and let \( m \) be a mapping from the rows of \( T_1 \) to the rows of \( T_2 \). \( m \) is a containment mapping if:

a) if \( T_1 \) has a distinguished variable in column \( A \) of row \( i \) then \( T_2 \) has a distinguished variable or a constant in column \( A \) of row \( m(\text{row}_i) \) of \( T_2 \);

b) if \( T_1 \) has a constant \( c \) in column \( A \) of row \( i \) then \( T_2 \) has a constant \( c \) in row \( m(\text{row}_i) \);

c) if rows \( i \) and \( j \) of \( T_1 \) have the same nondistinguished variable in column \( A \), then rows \( m(\text{row}_i) \) and \( m(\text{row}_j) \) of \( T_2 \) have the same symbol on that column.

**Definition.** A substitution is a possibly empty set of the form \((t_1/v_1, \ldots, t_n/v_n)\), where \( v_i \) is a variable and \( t_i \) is a term, different from \( v_i \), such that no two variables \( v_i, v_j \), if \( j \), are the same.

**Definition.** Let \( G \) be a substitution and let \( E \) be an expression. The instantiation of \( E \) (or instance of \( E \)) by \( G \) is denoted \( G.E \) and is the expression \( E \) where each occurrence of the symbol \( v_i \) has been replaced by its corresponding \( t_i \) in \( G \).

**Definition.** A clause \( C_1 \) subsumes another clause \( C_2 \) if and only if there is a substitution \( \theta \) such that \( \theta.C_1 \subset C_2 \). \( C_2 \) is said to be subsumed by \( C_1 \).

**Theorem 1.** \( T_2 \subset T_1 \) (\( T_2 \) is contained in \( T_1 \)) if and only if they define the same target relation and have a containment mapping \( m \) from \( T_1 \) to \( T_2 \). Proved in [11].

The definitions above introduce some necessary concepts we use ahead. The central idea here is to find a containment mapping between the two tableaux in order to verify their contain
ment. Let $T_a$ and $T_b$ be tableaux. We must prove that $T_a \subseteq T_b$ with $m$ if and only if there exists a substitution $G$ such that $G.C_a \subseteq C_b$, where $C_a$ and $C_b$ are the clauses corresponding to $T_a$ and $T_b$ respectively. Recall that if $G.C_a \subseteq C_b$, then we say that $C_a$ subsumes $C_b$. In other words, if the clauses $C_a$ and $C_b$ obtained from tableaux $T_a$, $T_b$ respectively, admit a substitution $G$ such that $G.C_a \subseteq C_b$ or $G.C_b \subseteq C_a$, then we can infer the existence of a containment mapping between the corresponding tableaux. That is, we can test the containment of tableaux using their corresponding clause form representation. In doing this, we are in fact testing the containment of two relational expressions. The equivalence of tableaux is decided from their containment. $T_a \equiv T_b$ if and only if $T_a \subseteq T_b$ and $T_b \subseteq T_a$. Similarly, $T_a = T_b$ if and only if $C_a$ subsumes $C_b$ and $C_b$ subsumes $C_a$.

**Lemma.** For each homomorphism $m: E_a \to E_b$ there exists an equivalent substitution $G$, such that $G.E_a \equiv E_b$.

**PROOF.** According with the symbols in $E_a$, we can always build $G$ as follows:

a) if $c$ is a constant, then $m(c) = c$, and no substitution for $c$ appears in $G$;
b) if $v_i$ is a distinguished variable and $m(v_i) = v_j$, the pair $v_j/v_i$ is in $G$;
c) if $v_i$ is a distinguished variable and $m(v_i) = t$, where $t$ can be distinguished, nondistinguished or a constant, the pair $t/v_i$ is in $G$.

**Theorem 2.** Let $T_a$ and $T_b$ be tableaux whose corresponding clause forms are $C_a$ and $C_b$ respectively. $T_a \subseteq T_b$ if and only if there exists $G$ such that $C_a$ subsumes $C_b$, viz. $G.C_a \subseteq C_b$.

**PROOF:** If direction.

Suppose $G$ exists. Then $T_a \subseteq T_b$.

a) if $w$ is a row of $T_a$ and $m(w)$ is in $T_b$, then if $L$ is a literal in $C_a$ necessarily $G.L$ is a literal in $C_b$;
b) if $T_a$ and $T_b$ have the same target relation schema, then
responding to the nondistinguished variable in \( w \) that disagrees
with some component of \( x \), can be replaced by that component, re-
sulting in a clause \( C' \), such that \( C' \) subsumes \( C \).

5. Conclusions

In the sections above we showed how some results from the
theory of tableaux can be obtained in a framework of clause form
representation for database queries. This section discusses some
of the implications of such and directions for further research.

The use of clause form is a useful representation due to the
possibility of using other methods for query optimization. Se-
monic query optimization \([3,5]\), for example, is well suited in
this framework and combines logical inference with the treatment
of clauses.

An issue not mentioned in the text is the use of functional
dependencies. Basically, if \( X \rightarrow Y \) is a functional dependency
and if rows \( i \) and \( j \) of tableau \( T \) have identical symbols in the
columns corresponding to the attributes in \( X \), then the symbols
corresponding to columns \( Y \) should be made identical for both
rows. This possibility is covered in clause form by the accep-
tance of function symbols in a better structured way. That is,
functions can be directly represented as terms and substitution
can take them into account.

Sagiv & Yannakakis \([7]\) extended the theory of tableaux for
the union operation and to a restricted family of relational ex-
pressions containing the set difference operator. For monotonic
relational expressions, viz, the ones where only the operators
select, projection, join and union are used, the authors show
that the equivalence and containment of tableaux can be charac-
terized in terms of containment of single tableaux. That permits
us to infer that the proofs presented above enable the assumption
that monotonic expressions are covered by clause form. The dif-
ficulty of tableaux in dealing with difference expressions can be
traced to its inability to handle function symbols. In clause
form however, this is solved by the introduction of Skolem func-
tions for existential quantifiers and the handling of function
symbols. In any case, Sagiv & Yannakakis express differences in
the literal in $G$ of $C_1$ maps to the literal in $G$ of $C_2$
through $G$.

Only if.
Suppose $G$ does not exist. There can be three reasons for that:
a) the literals in $G$ in clauses $C_1$ and $C_2$ have different
numbers of components, or their corresponding terms have
different domains. In this case, $T_1$ cannot contain $T_2$;
b) the literals in $G$ have constants $c_1$ and $c_2$, $c_1 \neq c_2$, in
corresponding columns. In this case, $T_1$ cannot contain $T_2$,
either because they contain different summaries or because their
summaries contain distinguished variables that have been mapped
to different terms;
c) for some literal $L$ in $C_1$, there is a constant $c$ as comp-
ponent 1, such that for any literal in $I$ in $C_2$, $G$ com-
ponent $i$ is a constant different from $c$. In this case,$
T_1$ cannot contain $T_2$ because if $w$ is the row of $T_1$ cor-
responding to $L$, then $m(w)$ is not in $T_2$.

Another major benefit from the theory of tableaux is the row
elimination rule, i.e., the simplification of queries. This co-
mes as a corollary to Theorem 1, and is proved in [13].

**Corollary.** Let $T$ be a tableau, $w$ and $x$ rows of $T$. If in what-
ver column $w$ and $x$ disagree, and $w$ has a non-distin-
guished variable that appears nowhere else in $T$, then
row $w$ can be eliminated, and the resulting tableau is
equivalent to $T$.

We state without proof its equivalent for clause form

**Theorem 3.** Let $T$ be a tableau, $w$ and $x$ rows of $T$. $w$ can be eli-
minated by the rule above if and only if clause $C$, 
corresponding to $T$ can be simplified by the removal
of its literal corresponding to row $w$.

The proof is based on the fact that the variable in $C$ cor-
terms of tableaux expressions, consisting of tableaux as operands and union, intersection and difference as operators.

In [8], we investigate the generalization of clause form representation to the complete set of relational operations, in order to analyse the connections between clause form representation and the work cited above.

6. References


8. Silveira, P. M. Equivalences Among Relational Expressions with Clause Manipulation, to appear as Technical Report NCE/UFRJ.