On the Connections Between Relational Tableaux and Clause Form

Pedro Manoel Silveira

Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica

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.

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.

A	B	C
x	y	

(xy | for some z, xyz ∈ I)

A	B	C
	y	z
x	y	z

(yz | for some x, xyz ∈ I)

A	B	C
x	y	z
x	y	v_{a_1}
v_{a_2}	y	z

(xyz | for some v_{a_1}, v_{a_2}, $xyv_{a_1} \in I$ & $v_{a_2}yz \in I$)

Figure 1. Examples of Relational Tableaux

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 atributes 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.

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_{a_1} and v_{a_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 atributes whose columns in the summary are non-blanks, and is defined as

$$T(I) = \{ \vartheta(w_0) \mid \text{for some valuation } \vartheta,\ \vartheta(w_i) \in I,\ 1 \le i \le n \}$$

A valuation ϑ associates with each symbol of T a constant c. For a row w_i, $\vartheta(w_i)$ is the tuple obtained by substituting

i) $\vartheta(v)$ for every variable v appearing in w_i;

ii) $\psi(c)=c$ for every constant c appearing in w_1.

3. Clause Form

Clause form [63] 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

$$\phi_1 \rightarrow \phi_2 \quad \text{by} \quad \sim\phi_1 \mid \phi_2$$

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

$$\begin{array}{rcl} \sim\sim\phi & \text{by} & \phi \\ \sim(\phi_1 \,\&\, \phi_2) & \text{by} & \sim\phi_1 \mid \sim\phi_2 \\ \sim(\phi_1 \mid \phi_2) & \text{by} & \sim\phi_1 \,\&\, \sim\phi_2 \\ \sim\forall x\phi & \text{by} & \exists x\sim\phi \\ \sim\exists x\phi & \text{by} & \forall x\sim\phi \end{array}$$

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 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

$$(\forall x\ \phi_1) \,\&\, \phi_2 \equiv \forall x(\phi_1 \,\&\, \phi_2)$$

and

$$(\forall x\ \phi_1) \mid \phi_2 \equiv \forall x(\phi_1 \mid \phi_2)$$

if x is not free in ϕ_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

$$\phi_1 \mid (\phi_2 \,\&\, \phi_3) \quad \text{by} \quad (\phi_1 \mid \phi_2) \,\&\, (\phi_1 \mid \phi_3)$$

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

quanfitiers 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

$$t_1,\ldots,t_n \mid v_1 \in D_1, \ldots, v_k \in D_k \mid \phi(v_1, \ldots, v_n)$$

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

$$\forall v_1 \in D_1 \ldots \forall v_k \in D_k \quad \phi(v_1, \ldots, v_n) \rightarrow Q(t_1,\ldots,t_n) \qquad (1)$$

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

$$Q(t_1,\ldots,t_n) \mid \Psi_i$$

where Ψ_i 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

$$Q(summary) \mid \neg I(row_1) \mid \ldots \mid \neg I(row_n)$$

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

$$Q(x,y,z) \mid \neg I(x,y,v_1) \mid \neg I(v_2,y,z)$$

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_a, then $m(w)$ is a row of T_b.

Definition. Let T_a and T_b be tableaux and let m be a mapping from the rows of T_a to the rows of T_b. m is a containment mapping if:

a) if T_a has a distinguished variable in column A of row i then T_b has a distinguished variable or a constant in column A of row $m(row_i)$ of T_b;
b) if T_a has a constant c in column A of row i then T_b has a constant c in row $m(row_i)$;
c) if rows i and j of T_a have the same nondistinguished variable in column A, then rows $m(row_i)$ and $m(row_j)$ of T_b 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, $i=j$, 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_a subsumes another clause C_b if and only if there is a substitution G such that $G.C_a \subseteq C_b$. C_b is said to be subsumed by C_a.

Theorem 1. $T_b \subseteq T_a$ (T_b is contained in T_a) if and only if they define the same target relation and have a containment mapping m from T_a to T_b. 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 containment. Let T_a and T_b be tableaux. We must prove that $T_b \subseteq T_a$ 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 respectivelly. 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 respectivelly, 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, $T_a \leq T_b$ if and

only if C_s subsumes C_m and C_m subsumes C_s.

Lemma. For each homomorphism $m:E_s \rightarrow E_m$ there exists an equivalent substitution G, such that $G.E_s \equiv E_m$.

PROOF. According with the symbols in E_s, 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_s and T_m be tableaux whose corresponding clause forms are C_s and C_m respectively. $T_m \subset T_s$ if and only if there exists G such that C_s subsumes C_m, viz, $G.C_s \subseteq C_m$.

PROOF: If direction.
Suppose G exists. Then $T_m \subset T_s$.
a) if w is a row of T_s and $m(w)$ is in T_m, then if L is a literal in C_s necessarily $G.L$ is a literal in C_m;
b) if T_s and T_m have the same target relation schema, then the literal in Q of C_s maps to the literal in Q of C_m through G.

Only if.
Suppose G does not exist. There can be three reasons for that:
a) the literals in Q in clauses C_s and C_m have different numbers of components, or their corresponding terms have different domains. In this case, T_s cannot contain T_m;
b) the literals in Q have constants c_s and c_m, $c_s \neq c_m$, in corresponding columns. In this case, T_s cannot contain T_m, 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_s, there is a constant c as component i, such that for any literal in I in C_m, $G.$component i is a constant different from c. In this case, T_s cannot contain T_m because if w is the row of T_s corresponding to L, then $m(w)$ is not in T_m.

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

Corollary. Let T be a tableau, w and x rows of T. If in whatever column w and x disagree, and w has a nondistinguished 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 eliminated 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 corresponding to the nondistinguished variable in w that disagrees with some component of x, can be replaced by that component, resulting 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. Semantic 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 -> 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 acceptance 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 expressions 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 characterized 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 difficulty 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 functions for existential quantifiers and the handling of function symbols. In any case, Sagiv & Yannakakis express differences in 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

1. Aho, A., Sagiv, Y. and Ullman, J. Equivalences Among Relational Expressions, SIAM J. Comput. 8,2(May 1979), 218-246.

2. Chang, C. and Lee, R., Symbolic Logic and Mechanical Theorem Proving, Academic Press, 1973.

3. Hammer, M. and Zdonik-Jr, S. B., Knowledge-Base Query Processing, Proc. of the 6th Int. Conf. VLDB, 1980.

4. Jarke, M. and Koch, J., Query Optimization in Database Systems, Computing Surveys 16,2(June 1984).

5. King, J., QUIST: A System for Semantic Query Optimization, Proc. of the 7th Int. Conf. VLDB, 1981.

6. Nilsson, N. J., Problem Solving Methods in Artificial Intelligence, Mc-Graw Hill, 1971.

7. Sagiv, Y. and Yannakakis, M. Equivalences Among Relational Expressions with The Union and Difference Operators, Journal of the ACM 27,4(October 1980), 633-655.

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

9. Ullman, J. D., Principles of Database Systems 2nd ed., Computer Science Press, 1982.