A THEORY OF DATA DEPENDENCIES OVER RELATIONAL EXPRESSIONS

Marco A. Casanova
Pontifícia Universidade Católica do R.J.
22453, Rio de Janeiro, RJ
Brasil

ABSTRACT

A formal system is developed for reasoning about a class of dependencies that includes all classes considered in the literature. The usefulness of the system is illustrated by applying it to various database design problems. The system is shown to be sound and complete by adapting the analytic tableaux method of first-order predicate calculus to the class of dependencies adopted. Finally, the method is shown to be a decision procedure for the inference problem of a subclass of the dependencies considered.

1. Introduction

We describe in this paper a formal system for reasoning about implicational dependencies [Fa2] defined over relational expressions (IDEXs). Examples of IDEXs are:

(1) (e(a,b,c), e(a,b',c') → b=b')

(2) (e(a,b,c), e(a,b',c') → e(a,b',c))

where e is a ternary relational expression. That is, e can be a complicated expression, such as (EMP[NAME,SAL] X MGR[MNAME,MSAL])[SAL>MSAL], and not just a base relation. The first IDEX asserts that the functional dependency A→B holds in the relation denoted by e and is abbreviated as e: A→B. The second IDEX asserts that the MVD A→→B|C holds in the relation denoted by e, and is abbreviated as e: A→→B.

IDEXs were chosen because they contain as special cases all data dependencies defined in the literature (as far as we know), such as functional dependencies (FDs) [Co1], multivalued and embedded multivalued dependencies (MVDs, EMVDs) [Fa1, ZM], join dependencies (JDs)[ABU,MMS], subset dependencies (SDs) [SW], template dependencies (TDs) [SU], algebraic dependencies [PY], generalized dependencies [GJ], extended embedded implicational dependencies [Fa2] and inclusion dependencies (INDs) [Fa3] (also called subset constraints in [Kℓ1, WM]).

The proof procedure behind the formal system is an adaptation of the analytic tableaux method [Sm], which has already proved quite attractive when applied to other logics, such as Process Logic [Pr] and Temporal Logic [RU]. The method can be viewed, in a sense, as a generalization of the chase procedure developed to reason about FDs and JDs [MMS] and later extended to FDs and TDs [SU]. The consistency and completeness of S are also obtained along the lines of [Sm].

2. Implicational Dependency Languages

This section defines a family of formal languages that we call implicational dependency languages (ID Languages). The formulas of an ID language are essentially boolean combinations of IDEXs and formulas of the form e($\underline{c}$) or $\underline{c}$ = $\underline{d}$, where e is a relational expression and $\underline{c}$, $\underline{d}$ are tuples of constants. An ID language L contains the following symbols:

(1) relation names: for each n>0, a non-empty set of n-ary relation names
(2) constant symbols: a non-empty set of symbols, distinct from the above
(3) the usual connectives and special symbols: ¬,∧ ∨, = ,(,),[,],→,=
(4) the usual relational operators: ×,∪,−

A relational expression of L, or simply an expression, and the arity of an expression are defined inductively as follows (an expression e of arity n is called an n-ary expression). Let ATTR(n) denote the set of sequences of distinct integers from the interval [1,n]:

(1) an n-ary relation name is an n-ary (atomic) expression;
(2) if e is an n-ary expression, T,U,V,X ∈ ATTR(n) and $\underline{a}$ is a tuple of constants such that |U| = |V| and |X| = |$\underline{a}$|, then the projection e[T] is a |T|-ary expression and the restriction e[U=V] and selection e[X=$\underline{a}$] are n-ary expressions;
(3) if e and f are m-ary and n-ary expressions, respectively, then the product (e×f) is an (n+m)-ary expression and, if n=m, the union (e ∪ f) and difference (e − f) are n-ary expressions.

We also introduce the join (e[X=Y]f) as an abbreviation for (e×f)[X=Y'], where Y' is obtained by adding n to each element of Y, if e is n-ary expression [K12]. Likewise, the intersection $(e \cap f)$ abbreviates e-(e-f).

Note that, following [K12], we do not give names to relation columns as usual in the relational model. This greatly simplifies the treatment of relational expressions. However, to enhance readability, we may occasionally reverse this position and name columns via relation schemas of the form $r[A_1,\ldots,A_n]$.

An atomic formula of L is either an equality $\bar{a}=\bar{b}$ or a relational formula $e(\bar{a})$, where $\bar{a}$, $\bar{b}$ are n-ary tuples of constants and e is an n-ary expression. Each constant in $\bar{a}$ or $\bar{b}$ is said to occur visible in $\bar{a}=\bar{b}$; each constant in $\bar{a}$ is said to occur visible in $e(\bar{a})$; and each constant in e is said to occur hidden in $e(\bar{a})$. If P is atomic, we use $P[\bar{a}/\bar{b}]$ to denote the atomic formula obtained by replacing each visible occurrence of b_i by a_i, where $\bar{a} = (a_1,\ldots,a_n)$ and $\bar{b} = (b_1,\ldots,b_n)$. A wff of L is either an atomic formula or of the form $\neg P$, $(P \wedge Q)$, $(P \vee Q)$, $(P \Rightarrow Q)$, where P,Q are wffs, or an implicational dependency over relational expressions of the form $(A_1,\ldots,A_n \rightarrow B)$ where A_i, $1 \leq i \leq n$, are relational formulas and B is either a relational formula or an equality such that each constant occurring visible in B also occurs visible in some A_i. By convention, we assume that no constant occurs visible in A_i and hidden in A_j, $1 \leq i,\ j \leq n$.

A structure I with domain D_I for L is a function assigning to each n-ary relation name r of L an n-ary relation $I(r) \in D_I^n$, n>0, and to each constant a, $I(a) \in D_I$ (if $\bar{a}=(a_1,\ldots,a_k)$, then $I(\bar{a})$ abbreviates $(I(a_1),\ldots,I(a_k))$). I is extended to the expressions of L as usual and to the wffs of L as follows:

(1) $I(\bar{a}=\bar{b})$ = true iff $I(\bar{a}) = I(\bar{b})$, otherwise $I(\bar{a}=\bar{b})$ = false
(2) $I(e(\bar{a}))$=true iff $I(\bar{a}) \in I(e)$, otherwise $I(e(\bar{a}))$= false
(3) $I((A_1,\ldots,A_n \rightarrow B))$ = true iff $J(A_1 \wedge \ldots \wedge A_n \Rightarrow B)$= true for all structures J of L which are identical to I, except on values of the constants occurring visible in $A_1,\ldots,A_n$; otherwise $I((A_1,\ldots,A_n \rightarrow B))$ = false
(4) I is extended to the other non-atomic wffs using the rules of Propositional Calculus.

The meaning of the wffs of L should be clear, except for IDEXs. Loosely speaking an IDEX could be defined as

(5) $(A_1,\ldots,A_n \rightarrow B) \equiv \forall x_1 \ldots \forall x_k\ (A_1' \wedge \ldots \wedge A_n' \Rightarrow B')$ where $A_1', \ldots, A_n', B'$ are obtained from $A_1,\ldots,A_n,B$, respectively, by replacing each visible occurrence of w_i $(1 \leq i \leq k)$ by x_i, where $w_1,\ldots,w_k$ are the constants occurring visible in $A_1,\ldots,A_k$. Note that, without our convention about the use of constants in $A_1,\ldots,A_n$, the definitions in (3) and (5) would not agree.

We now introduce in L by definition some of the familiar dependencies, all defined over relational expressions.

functional dependencies (FDs) :

(1) $e\colon X \rightarrow Y \equiv (e(\bar{a}),\ e(\bar{b}) \rightarrow \bar{a}_Y = \bar{b}_Y)$

where e is an n-ary expression, X, Y $\in$ ATTR(n), and $\bar{a}$, $\bar{b}$ are tuples of constants which are equal on the X-entries

inclusion dependencies (INDs) :

(2) $e \subseteq f \equiv (e(\bar{a}) \rightarrow f(\bar{a}))$

where e, f are n-ary expressions

multivalued dependencies (MVDs) :

(3) $e\colon X \twoheadrightarrow Y \equiv (e(\bar{a}),\ e(\bar{b}) \rightarrow e(\bar{c}))$

where e, X,Y, $\bar{a}$, $\bar{b}$ are as for FDs and $\bar{c}$ is a tuple of constants which agrees with $\bar{a}$ on all entries, except those in Y, and which agrees with $\bar{b}$ on the Y-entries.

join dependencies (JDs) :

(4) $e\colon\{X_1,\ldots,X_k\} \equiv (e(\bar{a}_1),\ldots,\ e(\bar{a}_k) \rightarrow e(\bar{b}))$

where e is an n-ary expression, $X_1,\ldots,X_k \in$ ATTR(n) are such that every $m \in [1,n]$ occurs in some X_j, $1 \leq j \leq k$, and $\bar{a}_i$ agrees with $\bar{b}$ on the X_i entries.

We conclude our list of basic definitions by saying that a set $\mathcal{P}$ of wffs is satisfiable iff all wffs in $\mathcal{P}$ are true in some structure of L. A set $\mathcal{P}$ of wffs logically implies a wff P iff P is true in every structure of L where all wffs in $\mathcal{P}$ are true (written $\mathcal{P} \models P$). P is a tautology iff P is true in all structures of L (written $\models P$, since, P is a tautology iff P is logically implied by the empty set of wffs). Given a class Σ of data dependencies the inference problem for Σ is the problem of determining, for any set $\mathcal{P}$ of dependencies in Σ and any dependency P in Σ, if $\mathcal{P} \models P$.

3. Examples

We start this section with examples illustrating the use of an ID language. Then, we discuss several problems that can be formulated within the framework we develop.

Examples of formulas involving define dependencies and their translations are (r is ternary and s is binary):

(1a) $r\colon 1 \rightarrow 2 \vee r\colon 1 \rightarrow 3 \Rightarrow r\colon 1 \twoheadrightarrow 2$

(1b) $((r(a,b,c),\ r(a,b',c') \rightarrow b = b')$

$(r(a,b,c),\ r(a,b',c') \rightarrow c = c')) \Rightarrow$

$(r(a,b,c),\ r(a,b',c') \rightarrow r(a,b',c))$

This formula is a well-known tautology [Ri].

(2a) $r\colon 1 \twoheadrightarrow 2 \equiv r\colon \{12,13\}$

(2b) $(r(a,b,c),\ r(a,b',c') \rightarrow r(a,b',c)) \equiv$

$(r(a,b,c),\ r(a,b',c') \rightarrow r(a,b,c'))$

Formula (2a) is also a well-known tautology [Fa1].

(3a) $(r[12] \subseteq s \wedge r[13] \subseteq s \wedge s\colon 1 \rightarrow 2) \Rightarrow (r(a,b,c) \rightarrow b=c)$

(3b) $((r[12](a,b) \rightarrow s(a,b)) \wedge (r[13](a,c) \rightarrow s(a,c)) \wedge$

$(s(a,b),\ s(a,c) \rightarrow b=c)) \Rightarrow$

$(r(a,b,c) \rightarrow b=c)$

This formula is a tautology and illustrates how the interplay between FDs and INDs leads to interesting new facts about r that can neither be expressed by an FD nor by an IND (the IDEX $(r(a,b,c) \rightarrow b=c)$ indicates that the second and third entries of each tuple in r are equal).

(4a) r: $1 \to 2 \wedge r[1] \subseteq r[2] \Rightarrow r[2] \subseteq r[1]$

(4b) $(r(a,b,c), r(a,b',c') \to b=b') \wedge (r[1](a) \to r[2](a)) \Rightarrow (r[2](a) \to r[1](a))$

This formula is not a tautology, but it is true in every structure I such that I(r) is finite. Hence , finite and infinite logical implication are not the same for IDEXs.

We now briefly discuss several problems that can be formulated within the framework we develop. Although each of these problems arise quite naturally when databases are designed, they had not received the appropriate attention in the literature or were attacked under unrealistic assumptions (such as the universal relation assumption or variations thereof).

Consider first the problem of determining if a constraint of a subschema σ' is valid in any state of σ' constructed from a consistent state of the base schema σ. The importance of this problem, in the context of database design, is discussed in [CCF]. To fix ideas, suppose that σ' has relation names $r_1,\ldots,r_n$ defined by expressions $e_1,\ldots,e_n$ (involving only relation names of σ). Let P be an IDEX $(A_1,\ldots,A_m \to B)$ and F be the constraints of σ. Then, P is valid in any state of σ' constructed from a consistent state of σ via $e_1,\ldots,e_n$ iff $F \models Q$ holds, where Q is obtained by replacing each occurrence of r_i in P by e_i $(1 \le i \le n)$. For example, if P is the FD r_i:X→Y, Q will be e_i:X→Y; note that, although P is an FD over a relation, Q is an FD over a relational expression. The subschema constraint problem was addressed in [Kℓ2], but only for FDs over expressions without set difference. As we show in Section 6, the decision procedure we develop extends the results in [Kℓ2] to a much more general class of dependencies over less restricted expressions.

Dependencies over expressions also arise quite naturally in another situation. For example, assume that we have two relation schemas r[ABC] and s[ABD] and that B must be functionally dependent on A in r and s taken together. That is, r(a,b) and s(a,b') would also imply b = b'. This constraint can be expressed simply as r[AB] ∪ s[AB]: A→B. We note that the same constraint is sometimes expressed by forcing r and s to be projections of a universal relation U with attributes ABCD and then assuming that U: A→B holds [BBG,BMSU]. (A better, but very similar approach can be found in [Me]). But the universal relation assumption is hard to justify [Ke] and may lead to undesirable consequences [BG].

The framework we develop is also useful to study lossless decompositions of relations. For example, let r[ABCD] be a relation scheme (to improve readability, we name the columns of r; we also use '*' to denote the natural join, defined in the usual way). Consider the horizontal fragmentation of r into r[A=1] and (r-r[A=1]) followed by a vertical fragmentation into s = (r[A=1])[AB], t = (r-r[A=1])[AC], u = (r[A=1])[ACD] and v = (r-r[A=1])[ABD]. Then, in the presence of s: A→B and t: A→C, we can reconstruct r as (s*u) ∪ (t*v). To prove this, it suffices to show that

(1) $s: A\to B,\ t: A \to C \models (s*u) \cup (t*v) \subseteq r$

Concrete illustrations of horizontal and vertical fragmentations can be found in [SS]. We show in Section 6 that our method can be used to establish (1). However, all currently available methods developed to cope with lossless decompositions are inappropriate to establish (1), including the chase procedure of [MMS, SU] (we will discuss the chase procedure further in Section 4).

To conclude this section, we discuss the problem of proving that an update preserves consistency of the database. For example, suppose that we want to prove that the deletion <u>if</u> $\neg r[X](\bar{a})$ <u>then</u> s:= s-s[Y=$\bar{a}$] preserves the consistency criterion $r[X] \subseteq s[Y]$. Then, using the familiar rules for assignments and if-then-else's [CB] this problem reduces to proving that

(2) $r[X] \subseteq s[Y] \models \neg r[X](\bar{a}) \Rightarrow r[X] \subseteq (s-s[Y=\bar{a}])[Y]$

which can be proved using the inference rules of Section 4. Note that (2) offers yet another natural example of a dependency defined over a relational expression.

4. A Formal System for Reasoning about IDEXs

Let L be an ID language. We introduce in this section a formal system S, whose language is L, and a proof procedure for S such that a wff P of L is logically implied by a set $\mathcal{P}$ of wffs of L iff P is a theorem of $\mathcal{P}$ in S. This result is proved in Section 5. Since the description of the rules of S dependes on the proof procedure, we discuss it first. From the point of view of classic Mathematics, our proof procedure formalizes the following familiar strategy to prove that $\mathcal{P} \models$ P. Start with $\mathcal{P}$ and $\neg$P and work out all possible cases. More precisely, organize the proof as a tree whose root contains $\mathcal{P}$ and $\neg$P and is such that the sons of a node correspond to branching cases. A proof organized this way is called an <u>analytic tableau</u>. Terminate the proof when each branch either contains a contradiction (i.e, <u>closes</u>) or cannot be extended further without repetition (i.e.<u>completes</u>). If all branchs close, $\mathcal{P} \cup \{\neg P\}$ is unsatisfiable and, hence, $\mathcal{P} \models$ P. If some branch completes without closing, $\mathcal{P} \cup \{\neg P\}$ is satisfiable (this is the main lemma of Section 5) and, hence, $\mathcal{P} \models$ P does not hold.

Reasoning by cases is captured by using rules of the form

$$\mathcal{R}_i \cdot \frac{\mathcal{P}_i}{\mathcal{Q}_{i1} | \ldots | \mathcal{Q}_{in_i}} \quad \text{where } \mathcal{P}_i \text{ and } \mathcal{Q}_{ij}\ (1 \le j \le n_i) \text{ are}$$

finite sets of wffs. Intuitively, $\mathcal{R}_i$ means that from $\mathcal{P}_i$ we can derive all wffs in $\mathcal{Q}_{ij}$ for some $j \in [1,n_i]$. We call $\mathcal{P}_i$ the <u>antecedent</u> of $\mathcal{R}_i$ and $\mathcal{Q}_{i1},\ldots,\mathcal{Q}_{in_i}$, the <u>consequents of $\mathcal{R}_i$</u>. A proof by case analysis can be formalized as follows:

Definition 4.1

(a) The set of <u>analytic tableaux</u> for a set $\mathcal{P}$ of wffs consists of trees whose nodes are sets of wffs. It is defined inductively as follows

(i) The tree whose only node is $\mathcal{P}$ is analytic tableau for $\mathcal{P}$;

(ii) Suppose that τ is an analytic tableau for $\mathcal{P}$ and let λ be a leaf of τ. Then, the tree obtained by extending τ by the following operation is also an analytic tableau for

P : if there is a rule R_i with antecedent P_i and consequents $Q_{i1},\ldots,Q_{in_i}$ such that all wffs in P_i occur in the branch ending in λ, then n_i distinct sons $\lambda_1,\ldots,\lambda_{n_i}$ may simultaneously be adjoined to λ , where $\lambda_j \subseteq Q_{ij}$ $(1 \le j \le n_i)$.

(b) A set H of wffs is a Hintikka set with respect to a set U of constants iff

(i) no wff and its negation are in H;

(ii) if there is a rule R_i with antecedent P_i and consequents $Q_{i1},\ldots,Q_{in_i}$, distinct from rules ID and $\neg$PR, such that $P_i \neq \emptyset$ and $P_i \subseteq H$, then $Q_{ij} \subseteq H$, for some $j \in [1,n_i]$;

(iii) if $(A_1,\ldots,A_n \to B) \in H$, where $\bar{w}$ are the constants occurring visible in $A_1,\ldots,A_n$, then for any tuple of constants $\bar{a}$ in U such that $|\bar{w}|=|\bar{a}|$, either $\neg A_i[\bar{a}/\bar{w}] \in H$, for some $i \in [1,n]$, or $B[\bar{a}/\bar{w}] \in H$

(iv) if $\neg e[X](\bar{a}) \in H$ then, for any tuple of constants $\bar{b}$ in U such that $\bar{b}_X = \bar{a}$ and $|\bar{b}|$ is equal to the arity of e, $\neg e(\bar{b}) \in H$

(c) A branch of a tableau is closed iff it contains a wff and its negation, otherwise it is open.

(d) A branch of a tableau is complete iff the union of all its nodes is a Hintikka set (with respect to the set of constants of the language).

(e) A tableau is closed iff every branch is closed.

(f) A tableau is complete iff every branch is closed or some branch is complete.

(g) A proof of a wff P from a set of wffs P is a closed tableau for $P \cup \{\neg P\}$. In this case, P is a theorem of P in S (written $P \vdash P$). □

We now describe the rules of S. By a new tuple of constants we mean a tuple of constants that do not occur in the tableau constructed thus far. It $t = (t_1,\ldots,t_n, t_{n+1},\ldots,t_{n+m})$, then $t_{[1,n]}$ denotes $(t_1,\ldots,t_n)$ and $t_{[n+1,n+m]}$ denotes $(t_{n+1},\ldots,t_{n+m})$.

ID-rules:

$\neg$ID. $\dfrac{\neg(A_1,\ldots,A_n \to B)}{A_1',\ldots,A_n',\ \neg B'}$ $\quad \bar{a} = (a_1,\ldots,a_k)$ is a new tuple of constants

ID. $\dfrac{(A_1,\ldots,A_n \to B)}{\neg A_1' \mid \ldots \mid \neg A_n' \mid B}$ $\quad \bar{a} = (a_1,\ldots,a_k)$ is any tuple of constants

where $\bar{w} = (w_1,\ldots,w_k)$ are the constants occurring visible in $A_1,\ldots,A_n$ and $A_i' = A_i[\bar{a}/\bar{w}]$, for $i \in [1,n]$, and $B' = B[\bar{a}/\bar{w}]$

Projection Rules

$\neg$PR. $\dfrac{\neg e[X](\bar{a})}{\neg e(\bar{b})}$ $\quad \bar{a},\bar{b}$ are any tuples of constants with b_X equal to a

PR. $\dfrac{e[X](\bar{a})}{e(\bar{b})}$ $\quad \bar{a}$ is any tuple of constants and $\bar{b}$ is a new tuple of constants such that b_X is equal to $\bar{a}$.

Restriction Rules

$\neg$RE. $\dfrac{\neg e[X=Z](\bar{a})}{\neg e(\bar{a}) \mid \neg \bar{a}_X = \bar{a}_Z}$ $\quad$ RE. $\dfrac{e[X=Z](\bar{a})}{e(\bar{a}),\ \bar{a}_X = \bar{a}_Z}$

$\bar{a}$ is any tuple of constants

Selection Rules

$\neg$SE. $\dfrac{\neg e[X=\bar{d}](\bar{a})}{\neg e(\bar{a}) \mid \neg \bar{a}_X = \bar{d}}$ $\quad$ SE. $\dfrac{e[X=\bar{d}](\bar{a})}{e(\bar{a}),\ \bar{a}_X = \bar{d}}$

$\bar{a}$ is any tuple of constants

Product Rules

$\neg$PT. $\dfrac{\neg(e \times f)(\bar{a})}{\neg e(\bar{a}_{[1,n]}) \mid \neg f(\bar{a}_{[n+1,n+m]})}$

PT. $\dfrac{(e \times f)(\bar{a})}{e(\bar{a}_{[1,n]}),\ f(\bar{a}_{[n+1,n+m]})}$

$\bar{a},\bar{b}$, are any tuples of constants and e is n-ary and f is m-ary

Union Rules

$\neg$UN. $\dfrac{\neg(e \cup f)(\bar{a})}{\neg e(\bar{a}),\ \neg f(\bar{a})}$ $\quad$ UN. $\dfrac{(e \cup f)(\bar{a})}{e(\bar{a}) \mid f(\bar{a})}$

$\bar{a}$ is any tuple of constants

Difference Rules

$\neg$DI. $\dfrac{\neg(e-f)(\bar{a})}{\neg e(\bar{a}) \mid f(\bar{a})}$ $\quad$ DI. $\dfrac{(e-f)(\bar{a})}{e(\bar{a}),\ \neg f(\bar{a})}$

$\bar{a}$ is any tuple of constants

Equality Rules

ES. $\dfrac{}{\bar{a} = \bar{a}}$ $\quad$ ER. $\dfrac{\bar{a}=\bar{b}}{\bar{b}=\bar{a}}$ $\quad$ ET. $\dfrac{\bar{a}=\bar{b},\ \bar{b}=\bar{c}}{\bar{a}=\bar{c}}$

EP. $\dfrac{\bar{a} = \bar{b}}{a_1=b_1,\ldots,a_n=b_n}$ $\quad$ $\neg$EP. $\dfrac{\neg \bar{a}=\bar{b}}{\neg a_1=b_1 \mid \ldots \mid \neg a_n=b_n}$

EI. $\dfrac{\bar{a}=\bar{b},\ e(\bar{a})}{e(\bar{b})}$ $\quad$ $\neg$EI. $\dfrac{a=b,\ \neg e(\bar{a})}{\neg e(\bar{b})}$

$\bar{a},\bar{b},\bar{c}$ are n-ary tuples of constants, n>0

A-rules. $\dfrac{A}{A_1, A_2}$ B-rules. $\dfrac{B}{B_1 \mid B_2}$

where A, A_1, A_2 and B, B_1, B_2 are given by the following tables:

A	A_1	A_2
$P \wedge Q$	P	Q
$\neg(P \vee Q$	$\neg P$	$\neg Q$
$\neg(P \Rightarrow Q$	P	$\neg Q$
$\neg\neg P$	P	P

Table 4.1

B	B_1	B_2
$\neg(P \wedge Q)$	$\neg P$	$\neg Q$
$P \vee Q$	P	Q
$P \Rightarrow Q$	$\neg P$	Q

Table 4.2

We now present proofs in S. As usual, examples are simplified if we make use of derived rules. Thus, we first augment S with (derived) rules for the FDs and INDs, which were introduced by definition at the end of Section 2.

FD-rules:

¬FD. $\dfrac{\neg e{:}X\to Y}{e(\bar{a}), e(\bar{b}), \bar{a}_X=\bar{b}_X, \neg \bar{a}_Y=\bar{b}_Y}$ $\bar{a},\bar{b}$ are tuples of new constants

FD. $\dfrac{e(\bar{a}), e(\bar{b}), \bar{a}_X=\bar{b}_X, e{:}X\to Y}{\bar{a}_Y=\bar{b}_Y}$ $\bar{a},\bar{b}$ are any tuples of constants

IND-rules:

¬IND. $\dfrac{\neg e \subseteq f}{e(\bar{a}), \neg f(\bar{a})}$ $\bar{a}$ is a tuple of new constants

IND. $\dfrac{e(\bar{a}),\ e \subseteq f}{f(\bar{a})}$ $\bar{a}$ is any tuple of constants

Example 4.1:

We exhibit a formal proof in S of the second half of Theorem 1 of [Ri]. This result essentially says that, given a partition of the columns of relation name r into X,Y,Z, if $r{:}X\to Y$ or $r{:}X\to Z$ hold, then the join of r[XY] and r[XZ] on X is a subset of r. Using the definition of join in terms of product and restriction, we formalize the above assertion as the following wff (call it Q):

(1) $r{:}X\to Y \vee r{:}X\to Z \models ((r[XY] \times r[XZ])[X=X'])[XYZ'] \subseteq r$

where X', Z' are obtained by adding k to each element of X,Z, respectively, if r is a k-ary relation name. We offer the following closed tableau as a proof that Q is indeed a tautology:

1. $r{:}X\to Y \vee r{:}X\to Z,\ \neg((r[XY] \times r[XZ])[X=X'])[XYZ'] \subseteq r$
2. $((r[XY] \times r[XZ])[X=X'])[XYZ'](\bar{a},\bar{b},\bar{c}),\ \neg r(\bar{a},\bar{b},\bar{c})$ .1, ¬IND
3. $((r[XY] \times r[XZ])[X=X'](\bar{a},\bar{b},\bar{a}',\bar{c})$ .2, PR
4. $(r[XY] \times r[XZ])(\bar{a},\bar{b},\bar{a}',\bar{c}),\ \bar{a}=\bar{a}'$ .3, RE
5. $r[XY](\bar{a},\bar{b}),\ r[XZ](\bar{a}',\bar{c})$ .4, PT
6. $r(\bar{a},\bar{b},\bar{c}'), r(\bar{a}',\bar{b}',\bar{c})$ .5, PR

7. $r{:}X\to Z$	8. $r{:}X\to Y$ .1, B-rule
9. $\bar{c}=\bar{c}'$ .4,6,7, FD	10. $\bar{b}=\bar{b}'$ .4,6,8, FD
11. $r(\bar{a},\bar{b},\bar{c})$ .6,9, EI	12. $r(\bar{a},\bar{b},\bar{c})$ .4,6,10, EI
X	X

note: the structure of the tableau (that is, the case structure) is indicated spacially. For example, 7 and 8 are sons of 6, and 10 is the only son of 8. A closed branch terminates with 'X'. □

Example 4.2

We now prove the first half to Theorem 1 of [Ri]. It says that, given a partition of the columns of a relation name r into X,Y,Z, if a set of FDs F implies that the join of r[XY] and r[XZ] on X is a subset of r, then F implies either $r{:}X\to Y$ or $r{:}X\to Z$. More formally, we have

(1) $F \vdash e \subseteq r$ implies $F \vdash (r{:}X\to Y \vee X\to Y)$ where

$e = (r[XY] \times r[XZ])[X=X'])[XYZ']$ and X',Z' are obtained by adding k to each element of X,Z, respectively, if r is a k-ary relation name. (Note that (1) is actually a metatheorem).

Proof

Assume $F \vdash e \subseteq r$. Then there is a closed tableau τ starting with $\delta_0 = F \cup \{\neg e \subseteq r\}$. Moreover, τ can be constructed using rules FD, ¬IND and those for relational expressions. Since only rule ¬IND can be applied to δ_0, it has only one son, which is $\delta_1 = \{e(\bar{a},\bar{b},\bar{c}), \neg r(\bar{a},\bar{b},\bar{c})\}$. Continuing to reason this way, we can show that τ has the following format:

1. $F,\ \neg e \subseteq r$
2. $e(\bar{a},\bar{b},\bar{c}),\ \neg r(\bar{a},\bar{b},\bar{c})$ .1, ¬IND
3. $((r[XY] \times r[XZ])[X=X'])\ (\bar{a},\bar{b},\bar{a},\bar{c})$ .2, PR, EI
4. $(r[XY] \times r[XZ])(\bar{a},\bar{b},\bar{a}',\bar{c}),\ \bar{a}=\bar{a}'$ .3, RE
5. $r[XY](\bar{a},\bar{b}),\ r[XZ](\bar{a}'\bar{c})$ .4, PT
6. $r(\bar{a},\bar{b},\bar{c}'),\ r(\bar{a}',\bar{b}',\bar{c})$ .5, PR, EI

⋮

n. E

n+1. $r(\bar{a},\bar{b},\bar{c})$ .6,n, EI

where E is either $\bar{b}=\bar{b}'$ or $\bar{c}=\bar{c}'$

Let us now try to prove that $F \vdash (r{:}X{\to}Y \vee r{:}X{\to}Z)$. We can start out a tableau σ as follows

1. $F,\ \neg(r{:}X{\to}Y \vee r{:}X{\to}Z)$
2. $\neg r{:}X{\to}Y,\ \neg r{:}X{\to}Z$. A-rule
3. $r(\bar{a},\bar{b},\bar{c}),\ r(\bar{a}',\bar{b}',\bar{c}'),\ \bar{d}=\bar{d}',\ \neg\bar{b}=\bar{b}'$. 2, $\neg$FD
4. $r(\bar{d},\bar{e},\bar{f}),\ r(\bar{d}',\bar{e}',\bar{f}'),\ \bar{d}=\bar{d}',\ \neg\bar{f}=\bar{f}'$. 2, $\neg$FD

But then the derivations between lines 6 and n of τ can be mimecked to extend σ to a closed tableau. That is, we can obtain either $\bar{b}=\bar{b}'$ or $\bar{f}=\bar{f}'$. Hence, $F \vdash (r{:}X{\to}Y \vee r{:}X{\to}Z)$ follows. □

We close this section with another perspective of the analytic tableaux method. From the point of view of database theory, the analytic tableaux method is closely connected with the chase method [MMS,SU] , if we imagine the latter extended to boolean combinations of dependencies involving relational expressions. However, the details of the two methods differ considerably. We first observe that indeed both methods talk about the existence of a tuple $\bar{a}$ in the relation denoted by an expression e. However, in the analytic tableaux method this is indicated by the formal statement $e(\bar{a})$, whereas in (the generalization of) the chase method the same would be asserted by entering $\bar{a}$ in a table T_e associated with e (different tables for e would have to be kept for different cases in a proof by case analysis). The analogy breaks down, though, when we observe that the analytic tableaux method also uses formulas of the form $\neg e(\bar{a})$ negating the existence of $\bar{a}$ in e. This is necessary when set difference is allowed. But consider what would happen if we tried to extend the chase method. We would need a rule, for example, to assert that if t is in table T_e and e=f-g, then t must be in table T_f, but t cannot appear in table T_g. This last fact is difficult to express in the chase method. Hence, the analytic tableaux method is more flexible in this case than the chase method.

5. Soundness and Completeness of System S

We prove in this section that S is sound and complete. Soundness means that $P \vdash P \Rightarrow P \models P$ holds and completeness signifies that the converse holds.

Since $P \models P$ iff $\models P_1 \wedge \ldots \wedge P_n \Rightarrow P$ and $P \vdash P$ iff $\vdash P_1 \wedge \ldots \wedge P_n \Rightarrow P$, where $P = \{P_1, \ldots P_n\}$, we may assume without loss of generality that P is empty. We also assume that the set of constants of the language L used by S is infinite (which assures that we do not run out of constants during a proof).

The soundness of S follows trivially by induction on the height of a tableau. To prove the completeness of S we have to show that if P is a tautology, then there is a closed tableau for $\neg P$ (i.e., that $\models P \Rightarrow\ \vdash P$). We actually prove that if P is a tautology, then every complete tableau for $\neg P$ closes. Or, equivalently, that if there is a complete open tableau for $\neg P$, then $\neg P$ is satisfiable and, hence, P is not a tautology. This result is obtained as follows. Recall that a tableau τ is complete and open iff some open branch β of τ forms a Hintikka set. We prove that, in fact, any Hintikka set is satisfiable. Hence, β is satisfiable and, since β starts with $\neg P$, so is $\neg P$.

Lemma 5.1: Any Hintikka set is satisfiable

Proof

Let H be a Hintikka set for L. We construct a structure I for L where all wffs in H are true. We first define a set E of classes of equivalence of constants. Let U be the set of constants of L and define $\rho = \{(a,a) / a \in U\} \cup \{(a,b) / \text{"}a{=}b\text{"} \in H\}$. By construction and since H is a Hintikka set (using the Equality rules), ρ is an equivalence relation. We take E as the set of equivalence classes of ρ. The equivalence class of a constant a is designated by a^0. I is constructed as follows. The domain of I is E; for each constant a, $I(a) = a^0$; for each n-ary relation name r, n > 0 , $I(r) = \{(a_1^0, \ldots, a_n^0) \in E^n / \text{"}r(a_1, \ldots, a_n)\text{"} \in H\}$.

Consider now I extended to a boolean valuation for the wffs of L. We show that each wff P in H is true in I by induction on the degree of P (the number of occurrence of $\to, \neg, \vee, \wedge, \Rightarrow$ and the relational operations in P).

basis: suppose that P has degree 0.

Then P is either $r(\bar{a})$ or $\bar{a}=\bar{b}$, where r is a relation name and $\bar{a},\bar{b}$ are tuples of constants. If P is $r(\bar{a})$ then, by construction of I, $\bar{a}^0 \in I(r)$. Hence, P is true in I. If P is $\bar{a}=\bar{b}$, the result follows likewise.

induction step: suppose that all wffs in H of degree less than i are true in I and let P∈H be a wff of degree i.

If P is $\neg r(\bar{a})$ or $\neg\bar{a}=\bar{b}$, then P is true in I by construction of I and definition of Hintikka set. Rather than proceeding with a detailed case analysis, we summarize all other cases as follows.

case schema 1: P is either $\neg(A_1, \ldots, A_n \to B)$, $e[X](\bar{a})$, $e[X{=}\bar{d}](\bar{a})$, $(e{\times}f)(\bar{a})$, $\neg(e{\cup}f)(\bar{a})$, $(e{-}f)(\bar{a})$, or the antecedent of an A-rule. Then, there is an instance of a rule R whose antecedent is P and whose consequent is $Q = \{Q_1, \ldots, Q_n\}$ where each Q_i has degree lower than P. Since H is a Hintikka set, each Q_i is in H. By the induction hypothesis, each Q_i is true in I. But, in each specific case, this implies that P is true in I. As an illustration, we prove the case that P is $\neg(A_1, \ldots, A_n \to B)$. Let $\bar{w} = (w_1, \ldots, w_k)$ be the constants visible in $(A_1, \ldots, A_n \to B)$. Since H is a Hintikka set (using rule $\neg$ID) , there are constants $\bar{a}=(a_1, \ldots, a_k)$ such that $A_i[\bar{a}/\bar{w}]$, $i \in [1,n]$ and $\neg B[a/w]$ are in H.

By the induction hypothesis and since these wffs have degree less than $\neg(A_1, \ldots, A_n \to B)$, they are true in I. Therefore, $I(A'_1 \wedge \ldots \wedge A'_n \Rightarrow B') =$ false. But this implies that $I((A_1, \ldots, A_n \to B))$ is false, by definition.

case schema 2: P is either $\neg e[X{=}Z](\bar{a})$, $\neg e[X{=}a](\bar{a})$, $\neg(e{\times}f)(\bar{a})$, $(e \cup f)(\bar{a})$, $\neg(e{-}f)(\bar{a})$ or the antecedent of a B-rule. Then, there is an instance of a rule R whose antecedent is P and whose consequents are $\{Q_1\}$ and $\{Q_2\}$, where Q_1 and Q_2 have degree lower than P. Since H is a Hintikka set, Q_i is in H, for some $i \in [1,2]$. By the induction hypothesis, Q_i is true in I. Again, in each specific case, this implies that P is true in I.

case schema 3: P is either $(A_1, \ldots, A_n \to B)$ or $\neg e[X](\bar{a})$. We prove only the first case.

Let $\bar{w} = (w_1, \ldots, w_k)$ be the constants visible in

$(A_1,\ldots,A_n \rightarrow B)$ and let $\bar{a} = (a_1,\ldots,a_k)$ be any tuple of constants in U. Since H is Hintikka set and P ε H, $Q(\bar{a})$ ε H where $Q(\bar{a})$ is either $A_i[\bar{a}/\bar{w}]$, for some iε[1,n], or $Q(\bar{a})$ is $B[\bar{a}/\bar{w}]$. Since $Q(\bar{a})$ has degree less than $(A_1,\ldots,A_n \rightarrow B)$, by the induction hypothesis, $Q(\bar{a})$ is true in I. Therefore, for any tuple $\bar{a}$ of constants in U, $A_1[\bar{a}/\bar{w}] \wedge \ldots \wedge A_n[\bar{a}/\bar{w}] \Rightarrow B[\bar{a}/\bar{w}]$ is true in I. But this implies that $(A_1,\ldots,A_n \rightarrow B)$ is true in I.

This conclude the proof. □

In order to use Lemma 5.1 to obtain a completeness proof for S we must guarantee that some branch of a tableau that does not close eventually becomes a Hintikka set. But the procedure given in Definition 4.1(a) permits constructing tableaux with infinite open branchs which are not Hintikka sets. This follows because: (i) rules may be applied redundantly to introduce wffs already derived; (ii) rules ¬ID, ID, ¬PR, PR may be repeatedly applied to generate wffs that differ only on the tuples of constants used; (iii) rule ES may always be applied using any tuple of constants. These problems are avoided by refining the procedure for constructing tableaux.

The refined procedure for constructing tableaux proceeds as in Definition 4.1(a), except that: (i) rules are never applied redundantly; (ii) as few constants as possible are used. To achieve these goals, additional bookkeeping is required. First, a tag is kept for each formula in a tableau indicating if that formula can still be used nonredundantly as antecedent of some rule. Second, a total order is defined among constants occurring in a tableau as follows. We say that a is older than b iff a occurs visible in a formula which was added to the tableau before any formula where b occurs visible. This partial order is then extended to a total order among constants. We also say that $(a_1,\ldots,a_n)$ is older than $(b_1,\ldots,b_n)$ iff a_i is older than or equal to b_i, for each iε[1,n], and a_j is older than b_j, for some jε[1,n].

The refined procedure constructs a tableau for a set of wffs P as follows. Initially, the tableau contains only one node, which is P. Let τ be the tableau constructed thus far. The procedure stops if any of the following conditions are satisfied.

T1. τ is closed.

T2. for some open branch θ, every wff in θ is tagged as used;

T3. for some open branch θ, the only unused wffs are of the form $(A_1,\ldots,A_n \rightarrow B)$ or $\neg e[X](\bar{a})$, and for each such wff Q there is no tuple of constants occurring in θ that was never used before with Q (in an application of the appropriate rule).

Otherwise, let λ be the node highest up in τ with an unused wff Q, which should not satisfy condition T3. τ is extended as follows. Take every open branch θ passing through λ and extend θ by applying all rules whose antecedent is Q (only two rules, EP and ER, have the same antecedent). Tag Q as used and each wff added as unused.

There are two special cases to consider:

(1) Q is of the form $(A_1,\ldots,A_n \rightarrow B)$ or $\neg e[X](\bar{a})$. Apply rule ID or ¬PR using Q and the oldest tuple of constants occurring in θ that was never used before with Q, and add Q along with each consequent. (We know that such tuple of constants exists because Q does not satisfy condition T3).

(2) Q is $\bar{a} = \bar{b}$, $e(\bar{a})$ or $\neg e(\bar{a})$. Try to apply rules ET, EI and ¬EI to derive new formulas not occurring in θ.

Intuitively, the refined procedure extends the tableau from the root down so that each wff is used exactly once as antecedent of a rule. The difficult part concerns rules ID and ¬PR. In order to generate a Hintikka set, if it is the case, rule ID has to be applied with all possible tuples of constants. This is achieved by a careful control of the constants already used and by repeating the antecedent of the rule along with the consequents. Similar remarks apply to rule ¬PR. (Strictly speaking the tree thus generated is not a tableau, but it can always be transformed into one by deleting the repeated formulas).

Another important feature of the procedure is that, when rules ID and ¬PR are applied, constants are selected from those used in the branch being extended, not from the set of all constants. Hence, the refined procedure guarantees that, if the tableau does not close, there is an open branch θ that forms a Hintikka set with respect to the set of constants occurring in θ, but not necessarily with respect to the set of all constants. But this does not affect the proof of Lemma 5.1 and opens the possibility of constructing finite Hintikka sets.

By a finished systematic tableau, we mean a tableau constructed by the refined procedure which is either infinite or else finite but cannot be extended further by the refined procedure.

We close this section with the completeness theorem for System S.

Theorem 5.2

(a) Every open finished systematic tableau has a branch which is a Hintikka set.

(b) If a wff P is a tautology, then every finished systematic tableau starting with ¬P must close.

(c) System S is complete.

Proof

(a) Follows by definition of the refined procedure for constructing tableaux.

(b) Suppose that there is a finished systematic tableaux starting with ¬P that is not closed. Then, by (a), it contains an open branch β which forms a Hintikka set H. By Lemma 5.1, H is satisfiable. Since ¬P ε H, ¬P is also satisfiable. Hence, P is not a tautology.

(c) Assume that P is a tautology. By (b), there is a closed tableau for ¬P. Hence, $\models P \Rightarrow \vdash P$. □

6. Decidability Questions for ID Languages

In this section we discuss the inference problem for ID languages. We first observe that this problem is undecidable since the inference problem for embedded implicational dependencies (EIDs) is undecidable [CLM] and EIDs are a special case of IDEXs. Thus, we concentrate on a class of instances of the inference problem for which the analytic tableaux method is a decision procedure.

We first state a lemma that gives a characterization of unbounded tableaux. We say that an application A of rule PR, with antecedent $e[X](\bar{a})$, is a consequence of an application A' of rule ID or rule ¬PR in a tableaux σ iff one of the wffs introduced in σ by A' generates the antecedent $e[X](\bar{a})$ of A, possibly after a sequence of applications of the rules for relational expressions.

Lemma 6.1: Let P be a finite set of wffs.

> σ is an unbounded systematic tableau starting with P iff rule PR is applied infinitely often in σ as a consequence of applications of rules ID or ¬PR.

Proof

($\Leftarrow$) Obvious, since rule PR is applied infinitely often in σ.

($\Rightarrow$) Suppose that σ is an unbounded systematic tableau for P (P is finite) but rule PR is applied finitely many times in σ as a consequence of applications of rules ID or ¬PR. By definition of the refined procedure, there are at most $|P|$ applications of rule ¬ID in σ and at most as many applications of rule PR that are not consequences of applications of rules ID or ¬PR as there are projection operations occurring in wffs in P. Then, there are finitely many applications of rules ¬ID and PR in σ.

Since these are the only two rules that introduce new constants, finitely many constants were used in σ. But then rules ID and ¬PR were also applied finitely many times in σ. This in turn implies that the refined procedure stops in finitely many steps. Hence σ is bounded. Contradiction. □

Lemma 6.1 suggests a way to guarantee that the refined procedure always stops. It suffices to restrict P and P so that, when trying to establish $P \models$ P, the refined procedure never applies rule PR as a consequence of applications of rules ID or ¬PR. This is not the case when P contains, for example, $(r(a,\overline{b,c}) \to s[ABC](a,b,c))$ or $((s-s[ABC])(a,b,c) \to r(a,b,c))$ since, after applying rule ID to each of these formulas, rule PR will be applied to $s[ABC](\bar{x})$, for some $\bar{x}$. The conditions on $P \models$ P discussed above can easily be translated to restrictions on the structure of P and P.

Towards this end, given an expression f of L and a specific occurrence p of a subexpression of f, we define the negation index or, simply, the index i(p,f) of p in f as the number of set difference operations prefixing p in f. For example, if f = e[A] - (g-e)[B], then the index of the leftmost occurrence of e is 0 and the index of the rightmost occurrence of e is 2.

Given a wff P and a specific occurrence p of an expression of P, we extend the index as follows (i(p,P) now counts set difference operations and negations):

(1) if P is $f(\bar{a})$, then i(p,P) = i(p,f)

(2) if P is ¬Q, then i(p,P) = i(p,Q) + 1

(3) if P is $(Q_1 \wedge Q_2)$ or $(Q_1 \vee Q_2)$ and p occurs in Q_i, then $i(p,P) = i(P,Q_i)$

(4) if P is $(Q_1 \Rightarrow Q_2)$ and p occurs in Q_1, then $i(p,P) = i(p,Q_1) + 1$ otherwise $i(p,P) = i(p,Q_2)$

(5) if P is $(A_1,\ldots,A_n \to B)$ and p occurs in A_i, then $i(p,P) = i(p,A_i) + 1$, otherwise i(p,P)=i(p,B)

Likewise, we define the index of an occurrence R of a subformula of a wff P as follows (i(R,P) counts negations prefixing R):

(6) if P is R then i(R,P) = 0

(7) if P is ¬R then i(R,P) = 1

(8) if P is $(Q_1 \wedge Q_2)$ or $(Q_1 \vee Q_2)$ and R occurs in Q_i, then $i(R,P) = i(R,Q_i)$

(9) if P is $(Q_1 \Rightarrow Q_2)$ and R occurs in Q_1, then $i(R,P) = i(R,Q_1) + 1$, otherwise $i(R,P) = i(R,Q_2)$

(10) if P is $(A_1,\ldots,A_n \to B)$ and R is A_i, $i \in [1,n]$, then i(R,P) = 1, otherwise i(R,P) = 0

We can now state the following theorem.

Theorem 6.2: The analytic tableaux method is a decision procedure for instances $P \models$ P of the inference problem for ID languages such that, for each subformula Q of a wff in $P \cup \{\neg P\}$ such that Q is of the form $(A_1,\ldots,A_n \to B)$ or ¬f[X](a), for each expression p occurring in Q such that p is of the form e[X], if Q has even degree, then p has odd degree.

Proof

Let $P \models$ P be an instance of the inference problem for ID languages satisfying the conditions of the theorem and suppose that the refined procedure does not stop when applied to $P \cup \{\neg P\}$.

By Lemma 6.1, the refined procedure does not stop iff rule PR is applied infinitely often as a consequence of rules ID or ¬PR. But rule ID is applied iff a wff Q of the form $(A_1,\ldots,A_n \to B)$ occurs in some node of the tableau, possibly after several applications of A- and B-rules. But this is possible only when Q occurs in some formula of $P \cup \{\neg P\}$ with even index. Now, this application of rule ID will have as a consequence an application of rule PR iff there is an occurrence p of an expression of the form e[X] in Q and the index of p is even. Similar observations apply when Q is of the form $\neg f[X](\bar{a})$. But in both cases, the conditions on $P \cup \{\neg P\}$ are violated. Contradiction. Hence, the procedure always stops when the input $P \cup \{\neg P\}$ satisfies the conditions of the theorem. □

We conclude this section with some examples and comments on the class of instances for which the analytic tableaux method (i.e., the refined procedure of Section 5) is a decision procedure (with appropriate translations for FDs, INDs and natural join):

(1) $r\colon X \to Y \vee r\colon X \to Z \models r[XY] * r[XZ] \subseteq r$

(2) $(r[A=1])[AB]\colon A \to B,\ (r-r[A=1])[AC]\colon A \to C$

$\models (r[A=1])[AB] * (r[A=1])[ACD] \cup$

$(r-r[A=1])[AC] * (r-r[A=1])[ABD] \subseteq r$

(3) $e_1\colon X_1 \to Y_1, \ldots, e_n\colon X_n \to Y_n \models e_0\colon X_0 \to Y_0$, where $e_0, \ldots, e_n$ are expressions that do not involve set difference.

We note at this point that, by (3), our result then contains as a special case the main result in [K12]. In other words, our result extends the main result in [K12] by considering a much wider class of dependencies (and not just FDs over expressions) and by allowing set difference (albeit in a restricted way).

The following instances do not satisfy the conditions of Theorem 6.2 and, in fact, the refined procedure diverges when applied to them:

(4) $r\colon A \to B,\ r[A] \subseteq r[B] \models r[B] \subseteq r[A]$

(5) $r[X] \subseteq s[Y] \models \neg r[X](\bar{a}) \Rightarrow r[X] \subseteq (s-s[Y=\bar{a}])[Y]$

However, we can rewrite (5) to conform with the conditions of Theorem 6.2. Indeed, we can transform (5) into:

(6) $t \subseteq u \models \neg t(\bar{a}) \Rightarrow t \subseteq (u-u[Y=\bar{a}])$

by defining $t=r[X]$ and $u=s[Y]$, and observing that

$(s-s[Y=\bar{a}])[Y] = (s[Y] - s[Y][Y=\bar{a}])$

7. Conclusions

This paper described a formal system for reasoning about implicational dependencies over relational expressions and an associated proof procedure based on the analytic tableaux method. The basic motivation was provided by various schema design problems briefly discussed in Section 3.

The analytic tableaux method proved to be quite attractive and easy to use manually. However, it may fail to stop, even in trivial, albeit pathological, cases. This should not be viewed as a handicap of the method because the problem it tries to solve is indeed undecidable. Moreover, we exhibited a rich class of instances of the problem for which the method is a decision procedure. But it must be added that the procedure for constructing systematic tableaux is quite inefficient, since it requires considerable extra bookkeeping. Hence, reasonable heuristics for reducts of the full problem should also be sought. However, the search for provably tractable reducts should never reduce the expressiveness power of the language beyond the point that it becomes irrelevant to the schema design problems that motivated this research.

Acknowledgements

This research was supported in part by FINEP and CNPq grant 402090/80. Support from IBM do Brasil is also gratefully acknowledged.

References

[ABU] A.V. Aho, C. Beeri and J.D. Ullman. "The Theory of Joins in Relational Databases", ACM TODS 4,3 (Sept. 1979), 297-314.

[BFH] C. Beeri, R. Fagin, J.H. Howard. "A Complete Axiomatization for Functional and Multivalued Dependencies". Proc. ACM SIGMOD Conf. (Aug. 1977), 47-61.

[BBG] C. Beeri, P.A. Bernstein, N. Goodman. "A Sophisticate's Introduction to Database Normalization" Proc. 1978 Very Large Database Conf., 113-124.

[BMSU] C. Beeri, A.O. Mendelzon, Y. Sagiv, J.D. Ullman. "Equivalence of Relational Database Schemes". Proc. 1979. ACM Symp. on the Theory of Computing, 319-329.

[BG] P.A. Bernstein, N. Goodman. "What does Boyce-Codd Normal Form do?". Proc. 1980 Very Large Database Conf.

[CB] M.A. Casanova, P.A. Bernstein, "A Formal System for Reasoning about Programs Accessing a Relational Database", ACM TOPLAS 2,2 (July 1980)

[CCF] M.A. Casanova, J.M.V. de Castilho, A.L. Furtado. "Properties of the Conceptual and External Schemas". (submitted for publication).

[CLM] A.K. Chandra, H.R. Lewis, J.A. Makowsky. "Embedded Implicational Dependencies and their Inference Problem". RC 8757, IBM T.J. Watson Research Center, Yorktown Heights, N.Y. (March 1981)

[Col] E.F. Codd. "Further Normalization of the Data Relational Model (R.Rustin,ed.), Prentice-Hall, N.J. (1972).

[DA] C.J. Date. "An Introduction to Database Systems". (2nd ed.), Addison-Wesley Pub. Co. (1977)

[EN] H.B. Enderton. "A Mathematical Introduction to Logic". Academic Press (1972)

[FA1] R. Fagin. "Multivalued Dependencies and a New Formal Form for Relational Databases". ACM TODS 2,3 (Sept. 1977)

[FA2] R. Fagin. "Horn Clauses and Database Dependencies". Proc. ACM SIGACT Symp. on the Theory of Computing (1980)

[FA3] R. Fagin. "A Normal Form for Relational Databases that is based on Domains and Keys" ACM TODS (to appear)

[GJ] J. Grant and B.E. Jacobs. "On Generalized Dependencies" (to appear).

[Ke] W. Kent. "Consequences of Assuming a Universal Relation". ACM TODS (to appear).

[Kl1] A. Klug. "Entity-Relationship Views Over Uninterpreted Enterprise Schemas". Proc. Int. Conf. Entity-Relationship Approach to Systems Analysis and Design (1979), 52-72.

[Dl2] A. Klug. "Calculating Constraints on Relational Expressions". ACM TODS 5,3 (Sept.1980)

[Me] A.O. Mendelzon. "Database States and their Tableaux". Comp. Syst. Research Group, Univ. of Toronto (unpublished TR).

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

[Pr] V.R. Pratt. "A Practical Decision Method for Propositional Dynamic Logic-Preliminary Report". Proc. 10th ACM Symp. on the Theory of Computing (1978), 326-337.

[Py] C. Papadimitriou and M. Yannakakis. "Algebraic Dependencies". Proc. 1980 IEEE Symp. on Foundations of Computer Science.

[Ri] J. Rissanen. "Independent Components of Relations". ACM TODS 2,2 (Dec. 1977) 317-325.

[RU] N. Rescher, A. Urquhart. "Temporal Logic". Springer-Verlag (1971)

[Sm] R.M. Smullyan. "First-Order Logic". Springer-Verlag (1971)

[SS] J.M. Smith and D.C.P. Smith. "Database Abstractions: Aggregation and Generalization". ACM TODS 2,2 (June 1977), 105-133

[SU] F. Sadri, J.D. Ullman. "A Complete Axiomatization for a Large Class Dependencies in Relational Databases". 1980 ACM Symp. on the Theory of Computing.

[SW] Y. Sagiv, S. Walecka. "Subset Dependencies as an Alternative to Embedded Multivalued Dependencies". TR UIUCDCS-R-79-980, U. of Illinois at Urbana-Champaign (July 1979).

[SY] Y. Sagiv, M. Yannakakis. "Equivalence among Relational Expressions with the Union and Difference Operations". JACM 27,4 (Oct. 1980), 633-655.

[WM] G. Wiederhold, R.El-Masri. "A Structural Model for Database Systems". TR STAN-CA-79-722, Stanford University (Feb. 1979).