

THE THEORY OF FUNCTIONAL AND SUBSET DEPENDENCIES OVER RELATIONAL EXPRESSIONS

Marco A. CASANOVA

Pontificia Universidade Catolica do RJ, 22453 Rio de Janeiro, RJ, Brasil

Communicated by M. Paul

Received 22 June 1981

Revised 1 June 1982

A formal system for reasoning about functional dependencies (FDs) and subset dependencies (SDs) defined over relational expressions is described. An FD $e: X \rightarrow Y$ indicates that Y is functionally dependent on X in the relation denoted by expression e ; an SD $e \subset f$ indicates that the relation denoted by e is a subset of that denoted by f . The system is shown to be sound and complete by resorting to the analytic tableaux method. Applications of the system include the problem of determining if a constraint of a subschema is implied by the constraints of the base schema and the development of database design methodologies similar to normalization.

Keywords: Formal languages, relational expressions, functional dependencies

1. Introduction

We describe in this paper a formal system S for reasoning about functional dependencies (FDs) and subset dependencies (SDs) defined over relational expressions. The class of FDs considered consists of statements of the form $e: X \rightarrow Y$ which assert that the Y -columns are functionally dependent on the X -columns in the relation defined by the relational expression e . Likewise, SDs are statements of the form $e \subset f$ specifying that the relation defined by e is a subset of that defined by f .

The development of system S was motivated primarily by the subschema constraint problem, viz., whether a constraint of a subschema σ' is valid in any state of σ' constructed from a consistent state of the base schema σ . For example, suppose that σ' has relation names r_1, \dots, r_n defined by expressions e_1, \dots, e_n (only involving relation names of σ). Let $r_i: X \rightarrow Y$ be an FD of σ' . Then, $r_i: X \rightarrow Y$ is valid in any state of σ' constructed from a consistent state of σ via e_1, \dots, e_n iff $e_i: X \rightarrow Y$ is a logical consequence of the con-

straints of σ . This last assertion can then be resolved in S .

The subschema constraint problem was studied in [11,12,14], but not for the class of FDs and SDs over expressions. Formal systems for several classes of dependencies over relations were studied, e.g., in [1,10,3,21,15]. Other examples of the use of tableaux can be found in [20,16,18].

This paper is organized as follows. Section 2 defines FDs and SDs over relational expressions. Section 3 describes the formal system S and the analytic tableaux method. Section 4 discusses the soundness and the completeness of S . Finally, Section 5 discusses on-going research.

2. Functional and subset dependency languages

This section defines a family of formal languages that we call *functional and subset dependency languages* (FD-SD languages). An FD-SD language \mathcal{L} contains the following symbols:

(1) relation names: for each $n > 0$, a nonempty set of n -ary relation names,

(2) constant symbols: a nonempty set of symbols, distinct from the above,

(3) the usual connectives and special symbols:

$\neg, \wedge, \vee, \Rightarrow, (,), [,], \rightarrow, \subset, =,$

(4) the usual relation operators: $\times, \cup, -$.

An n-ary *relational expression* of \mathcal{L} is defined inductively as follows. Let $\text{ATTR}(n)$ 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, $T, U, V, X \in \text{ATTR}(n)$ and \bar{a} is a tuple of constants such that $|U| = |V|$ and $|X| = |\bar{a}|$, then the *projection* $e[T]$ is a $|T|$ -ary expression and the *restriction* $e[U = V]$ and *selection* $e[X = \bar{a}]$ are n-ary expressions;

(3) if e and f are m-ary and n-ary expressions, respectively, then the *product* $(e \times f)$ is an $(n + m)$ -ary expression and, if $n = m$, the *union* $(e \cup f)$ and *difference* $(e - f)$ are n-ary expressions.

An *atomic formula* of \mathcal{L} is either $\bar{a} = \bar{b}$, $e(\bar{a})$, a *functional dependency* $e: X \rightarrow Y$ or a *subset dependency* $e \subset f$, where \bar{a}, \bar{b} are n-ary tuples of constants, e, f are n-ary expressions and $X, Y \in \text{ATTR}(n)$, $n > 0$. A *well-formed formula* (wff) of \mathcal{L} is either an atomic formula or of the form $\neg P$, $(P \wedge Q)$, $(P \vee Q)$ or $(P \Rightarrow Q)$ where P, Q are wffs.

A *structure* I with *domain* D_1 for \mathcal{L} is a function assigning to each n-ary relation name r of \mathcal{L} an n-ary relation $I(r) \subseteq D_1^n$, $n > 0$, and to each k-ary tuple of constants \bar{a} , a k-ary tuple $I(\bar{a}) \in D_1^k$, $k > 0$. I is extended to the relational expressions of \mathcal{L} as follows (note that I is already defined for the atomic expressions):

- (1) $I(e[T]) = \{\bar{a}_T \mid \bar{a} \in I(e)\}$,
- (2) $I(e[U = V]) = \{\bar{a} \mid \bar{a} \in I(e) \wedge \bar{a}_U = \bar{a}_V\}$,
- (3) $I(e[X = \bar{a}]) = \{\bar{a} \mid \bar{a} \in I(e) \wedge \bar{a}_X = I(\bar{a})\}$,
- (4) $I(e \times f) = \{\bar{a}\bar{b} \mid \bar{a} \in I(e) \wedge \bar{b} \in I(f)\}$,
- (5) $I(e \cup f) = \{\bar{a} \mid \bar{a} \in I(e) \vee \bar{a} \in I(f)\}$,
- (6) $I(e - f) = \{\bar{a} \mid \bar{a} \in I(e) \wedge \neg \bar{a} \in I(f)\}$.

We now extend I to a Boolean valuation of the wffs of \mathcal{L} as follows:

(1) $I(\bar{a} = \bar{b}) = \text{true}$ if $I(\bar{a}) = I(\bar{b})$, otherwise $I(\bar{a} = \bar{b}) = \text{false}$;

(2) $I(e(\bar{a})) = \text{true}$ if $I(\bar{a}) \in I(e)$, otherwise $I(e(\bar{a})) = \text{false}$;

(3) $I(e: X \rightarrow Y) = \text{true}$ if $\forall \bar{a} \forall \bar{b} (\bar{a} \in I(e) \wedge \bar{b} \in I(e) \wedge \bar{a}_X = \bar{b}_X \Rightarrow \bar{a}_Y = \bar{b}_Y)$

otherwise $I(e: X \rightarrow Y) = \text{false}$;

(4) $I(e \subset f) = \text{true}$ if $\forall \bar{a} (\bar{a} \in I(e) \Rightarrow \bar{a} \in I(f))$, otherwise $I(e \subset f) = \text{false}$;

(5) I is extended to the nonatomic wffs using the rules of Propositional Calculus.

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 \mathcal{L} . A set \mathcal{P} of wffs *logically implies* a wff P iff P is true in every structure of \mathcal{L} where all wffs in \mathcal{P} are true (written $\mathcal{P} \models P$). Finally, P is a *tautology* iff P is true in all structures of \mathcal{L} (written $\models P$, since P is a tautology iff P is logically implied by the empty set of wffs).

3. A formal system for reasoning about functional and subset dependencies

Let \mathcal{L} be an FD-SD language. We introduce in this section a formal system S , whose language is \mathcal{L} , and a proof procedure for S such that a wff P of \mathcal{L} is logically implied by a set \mathcal{P} of wffs of \mathcal{L} iff P is a theorem of \mathcal{P} in S . This result is proved in Section 4. Since the description of the rules of S depends on the proof procedure, we discuss it first.

The notion of a proof in S is a direct generalization of the analytic tableaux method for Propositional Calculus [20]. It 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. If every case leads to a contradiction, then $\mathcal{P} \cup \{\neg P\}$ is unsatisfiable and, hence, $\mathcal{P} \models P$. On the other hand, if the analysis of some case is exhausted without arriving at a contradiction, then $\mathcal{P} \cup \{\neg P\}$ is satisfiable (this has to be proved, since it is not an immediate property of the system) and, hence, $\mathcal{P} \not\models P$ does not hold.

Reasoning by cases is captured by using rules of the following type:

$$\mathcal{R}_i : \frac{\mathcal{P}_i}{\mathcal{Q}_{i1} \mid \dots \mid \mathcal{Q}_{in_i}}$$

where \mathcal{P}_i and \mathcal{Q}_{ij} ($1 \leq j \leq n_i$) 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 *antecedent* of \mathcal{R}_i and $\mathcal{Q}_{i1}, \dots, \mathcal{Q}_{in_i}$ the *consequents* of \mathcal{R}_i . A proof by case analysis can be organized

as a tree, where the sons of a node correspond to branching cases. A proof terminates when each branch either contains a contradiction or cannot be extended further without repetition. These observations are formalized as follows (by a branch of a tree we mean a path from the root to a leaf).

Definition 3.1. (a) The set of *analytic tableaux* 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 an 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 \mathcal{P} : if there is a rule \mathcal{R}_i with antecedent \mathcal{P}_i and consequents $\mathcal{Q}_{i1}, \dots, \mathcal{Q}_{in_i}$ such that all wffs in \mathcal{P}_i occur in the branch ending in λ , then n_i distinct sons $\lambda_1, \dots, \lambda_{n_i}$ may simultaneously be adjoined to λ , where $\lambda_j \subset \mathcal{Q}_{ij}$ ($1 \leq j \leq n_i$).

(b) A set \mathcal{H} of wffs is a *Hintikka set* iff no wff and its negation are in \mathcal{H} and if there is a rule \mathcal{R}_i with antecedent \mathcal{P}_i and consequents $\mathcal{Q}_{i1}, \dots, \mathcal{Q}_{in_i}$ such that $\mathcal{P}_i \neq \emptyset$ and $\mathcal{P}_i \subset \mathcal{H}$, then $\mathcal{Q}_{ij} \subset \mathcal{H}$ for some $j \in [1, n_i]$.

(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.

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

(f) A tableau is *complete* iff each branch is either closed or complete.

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

Product rules

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

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

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

Union rules

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

$$\text{UN.} \quad \frac{(e \cup f)(\bar{a})}{e(\bar{a}) | f(\bar{a})}$$

\bar{a} is any tuple of constants.

Difference rules

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

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

\bar{a} is any tuple of constants.

Equality rules

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

$$\text{EP.} \quad \frac{\bar{a} = \bar{b}}{a_1 = b_1, \dots, a_n = b_n}$$

$$\neg\text{EP.} \quad \frac{\neg \bar{a} = \bar{b}}{\neg a_1 = b_1 | \dots | \neg a_n = b_n}$$

$$\text{EI.} \quad \frac{\bar{a} = \bar{b}, e(\bar{a})}{e(\bar{b})}$$

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

A-rules and B-rules

$$\text{A-rules.} \quad \frac{A}{A_1, A_2} \quad \text{B-rules.} \quad \frac{B}{B_1 | B_2}$$

where A, A_1, A_2 and B, B_1, B_2 are given by Table 1.

Intuitively, the rules of S capture patterns of reasoning commonly used in mathematics. Rule $\neg\text{FD}$, for example, captures the following pattern: "... Assume $\neg e: X \rightarrow Y$. Then, there must be two tuples in the value of e agreeing on X , but not on Y . Let \bar{a} and \bar{b} denote these tuples ..." (where \bar{a} and \bar{b} have not been previously used). Rules FD , $\neg\text{SD}$ and SD are similarly explained. The rules for

Table 1

A	A ₁	A ₂	B	B ₁	B ₂
P ∧ Q	P	Q	¬(P ∧ Q)	¬P	Q
¬(P ∨ Q)	¬P	¬Q	P ∨ Q	P	Q
¬(P ⇒ Q)	P	¬Q	P ⇒ Q	¬P	Q
¬¬P	P	P			

relational expressions directly reflect previous definitions. Finally, the equality, the A-rules and the B-rules are quite familiar, except for rules EP and ¬EP, which follow if we interpret $\bar{a} = \bar{b}$ as

$$\bigwedge_{i=1}^n a_i = b_i.$$

At a more formal level the rules of S may be viewed as derived rules of first-order logic. This remark does not imply that S is superfluous, since S shortens proofs by hiding quantifiers and certain standard derivations pertaining to FDs and SDs. As an example of our claim we derive rules ¬SD and SD (ignoring certain technical difficulties raised by relational expressions, which can be circumvented along the lines of [5; 9, Chapter 4]). Consider the definition of $e \subset f$ cast as follows:

$$(i) \ e \subset f \equiv \forall \bar{a}(e(\bar{a}) \Rightarrow f(\bar{a})).$$

Then, (i) is equivalent to the conjunction of

$$(ii) \ \forall \bar{a}(e \subset f \wedge e(\bar{a}) \Rightarrow f(\bar{a})),$$

$$(iii) \ \exists \bar{a}(\neg e \subset f \Rightarrow e(\bar{a}) \wedge \neg f(\bar{a})).$$

By applying the usual rules of first-order logic (based on tableaux [20]), (ii) generates SD and (iii) generates ¬SD.

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

FD-rules

$$\neg\text{FD.} \quad \frac{\neg e : X \rightarrow Y}{e(\bar{a}), e(\bar{b}), \bar{a}_X = \bar{b}_X, \neg \bar{a}_Y = \bar{b}_Y}$$

\bar{a}, \bar{b} are tuples of new constants,

$$\text{FD.} \quad \frac{e(\bar{a}), e(\bar{b}), \bar{a}_X = \bar{b}_X, e : X \rightarrow Y}{\bar{a}_Y = \bar{b}_Y}$$

\bar{a}, \bar{b} are any tuples of constants.

SD-rules

$$\neg\text{SD.} \quad \frac{\neg e \subset f}{e(\bar{a}), \neg f(\bar{a})}$$

\bar{a} is a tuple of new constants,

$$\text{SD.} \quad \frac{e(\bar{a}), e \subset f}{f(\bar{a})}$$

\bar{a} is any tuple of constants.

Projection rules

$$\neg\text{PR.} \quad \frac{\neg e[X](\bar{a}), e(\bar{b})}{\neg \bar{b}_X = \bar{a}}$$

\bar{a}, \bar{b} are any tuples of constants,

$$\text{PR.} \quad \frac{e[X](\bar{a})}{e(\bar{b}), \bar{b}_X = \bar{a}}$$

\bar{a} is any tuple of constants and \bar{b} is a tuple of new constants.

Restriction rules

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

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

\bar{a} is any tuple of constants.

Selection rules

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

$$\text{SE.} \quad \frac{e[X = \bar{d}](\bar{a})}{e(\bar{a}), \bar{a}_X = \bar{d}}$$

\bar{a} is any tuple of constants.

Example 3.2. We exhibit a formal proof in S of the second half of Theorem 1 of [17]. This result essentially says that, given a partition of the columns of a relation name r into X, Y, Z , if $r : X \rightarrow Y$ or $r : X \rightarrow Z$ 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

Table 2 ^a

1.	$r: X \rightarrow Y \vee r: X \rightarrow Z, \neg((r[XY] \times r[XZ])[X = X'])[XYZ'] \subset r$				
2.	$((r[XY] \times r[XZ])[X = X'])[XYZ'](\bar{a}, \bar{b}, \bar{c}), \neg r(\bar{a}, \bar{b}, \bar{c})$.1, \neg SD			
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.	$r: X \rightarrow Z$	8.	$r: X \rightarrow Y$.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	

^a We indicate the tree structure spacially, so that 7. and 8. should be understood as sons of 6., and 10. as the only son of 8., for example.

the above assertion as the following wff (call it Q):

$$r: X \rightarrow Y \vee r: X \rightarrow Z \Rightarrow ((r[XY] \times r[XZ])[X = X'])[XYZ'] \subset r \quad (1)$$

where X', Z' are obtained by adding k to each element of X, Z, respectively, if r is a k-ary relation name.

In Table 2 we offer closed tableaux as a proof that Q is indeed a tautology (starting with the result of applying an A-rule to the negation of (1)).

Example 3.3. In this example we prove that the following wff,

$$e: SN \rightarrow SNAME, SCITY, STATUS \quad (2)$$

with

$$e = ((SUPPLIER \times CS) [SCITY = CITY]) [SN, SNAME, SCITY, STATUS],$$

is a theorem of the following set of wffs,

$$\begin{aligned} SUPPLIER: SN \rightarrow SNAME, SCITY, \\ CS: CITY \rightarrow STATUS. \end{aligned} \quad (3)$$

We offer the tableau in Table 3 as a proof.

4. Soundness and completeness of system S

We discuss in this section the soundness and

Table 3 ^a

1.	SUPPLIER: SN \rightarrow SNAME, SCITY, CS: CITY \rightarrow STATUS, $\neg e: SN \rightarrow SNAME, SCITY, STATUS$					
2.	$e(s, n, sc, st), e(s', n', sc', st'), s = s', \neg(n, sc, st) = (n', sc', st')$.1, \neg FD			
3.	$f(s, n, sc, c, st), f(s', n', sc', c', st')$.2, PR			
	with $f = (SUPPLIER \times CS)[SCITY = CITY]$					
4.	$g(s, n, sc, c, st), g(s', n', sc', c', st'), sc = c, sc' = c'$.3, RE			
	with $g = SUPPLIER \times CS$					
5.	SUPPLIER(s, n, sc), CS(c, st), SUPPLIER(s', n', sc'), CS(c', st')		.4, PT			
6.	$(n, sc) = (n', sc')$.1, 2, 5, FD			
7.	$n = n', sc = sc'$.6, EP			
8.	$c = c'$.4, 7, ET			
9.	$st = st'$.1, 5, 8, FD			
10.1	$\neg n = n'$	10.2	$\neg sc = sc'$	10.3	$\neg st = st'$.2, \neg EP
	X		X		X	

^a To save space, some steps such as 8. summarize several derivations.

Thus, step 2. exhibits two tuples in e that violate the FD in (2); steps 3. to 5. indicate that, as a consequence, certain tuples must exist in SUPPLIER and CS; steps 6. to 9. indicate the consequences of assuming the FDs in (3); steps 10.1, 10.2 and 10.3 expand the negated equality appearing in step 2., thus leading to contradictions.

completeness of S. Soundness means that

$$\mathcal{P} \vdash P \Rightarrow \mathcal{P} \models P$$

holds and completeness signifies that the converse holds. Since

$$\mathcal{P} \models P \quad \text{iff} \quad \models P_1 \wedge \dots \wedge P_n \Rightarrow P$$

and

$$\mathcal{P} \vdash P \quad \text{iff} \quad \vdash P_1 \wedge \dots \wedge P_n \Rightarrow P,$$

where $\mathcal{P} = \{P_1, \dots, P_n\}$, we may assume without loss of generality that \mathcal{P} is empty. (We tacitly assume that \mathcal{P} is always finite.) We also assume that the set of constants of the language \mathcal{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. (All proofs appear in [4].)

Theorem 4.1. *S is sound.*

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 every 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 4.2. *Any Hintikka set is satisfiable.*

In order to use Lemma 4.2 to obtain a completeness proof for S we must guarantee that every branch of a tableau that does not close eventually becomes a Hintikka set. But the procedure given in Definition 3.1(a) permits constructing tableaux with infinite open branches which are not Hintikka sets. This follows because: (i) rules may be applied redundantly to introduce wffs already derived, (ii) rules \neg FD, \neg SD and PR may be repeatedly applied to generate wffs that differ only on the tuples of constants used, and (iii) rule ES may always be

applied using any tuple of constants. These problems are avoided by refining the procedure for constructing tableaux so that rules are nonredundantly applied in a cyclical pattern.

In what follows we assume that the 29 rules of S are numbered from 0 to 28. Let \mathcal{P} and \mathcal{Q} be sets of wffs occurring in branches β and γ (not necessarily distinct) of a tableau τ , respectively. We say that \mathcal{P} is *higher than* \mathcal{Q} iff some wff in \mathcal{P} occurs in a node of β with level higher than the level of any node of γ containing a wff of \mathcal{Q} . Likewise, let \bar{a} and \bar{b} be tuples of constants occurring in wffs of β and γ , respectively. We say that \bar{a} is *higher than* \bar{b} iff \bar{a} occurs in a node of β with level higher than the level of any node of γ containing an occurrence of \bar{b} .

The refined procedure for constructing tableaux works as follows. Let τ be an open tableau. Let \mathcal{R}_i be the last rule that was considered for application in τ . Consider now for application rule \mathcal{R}_j , $j = i + 1 \pmod{29}$:

Case 1. \mathcal{R}_j is not ES.

Let \mathcal{P} be a set of wffs occurring in an open branch β of τ that was never used as antecedent of \mathcal{R}_j . If no such set \mathcal{P} exists, do not apply \mathcal{R}_j . Otherwise, assume that no other set \mathcal{Q} of wffs with the same property as \mathcal{P} is higher than \mathcal{P} . Let λ be the node lowest in β that contains some wff in \mathcal{P} . Then, extend all branches of τ that contain λ by applying \mathcal{R}_j . \mathcal{P} will never be used again as antecedent of \mathcal{R}_j . This completes Case 1.

Case 2. \mathcal{R}_j is ES.

Let \bar{a} be a tuple of constants occurring in an open branch β of τ that was never used by ES. If no such \bar{a} exists, do not apply ES. Otherwise, assume that no other tuple \bar{b} with the same property as \bar{a} is higher than \bar{a} . Let λ be the node highest in β that contains an occurrence of \bar{a} . Then, extend all branches of τ that contain λ by applying ES using $\bar{a} = \bar{a}$ as consequent. This concludes Case 2.

The procedure stops either when τ is closed or when all 29 rules of S were unsuccessfully considered for application. 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 can now state the Completeness Theorem for system S.

Theorem 4.3 (a) *Every open branch of every finished systematic tableau is a Hintikka set.*

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

(c) *System S is complete.*

We conclude this section with some observations about the decidability of the tautology problem for an FD-SD language \mathcal{L} (i.e., the problem of determining whether a wff of \mathcal{L} is a tautology). This problem is undecidable because it can be trivially reduced to the equivalence problem for relational expressions of \mathcal{L} , which is undecidable by Theorem 6 of [14]. The equivalence problem for relational expressions of \mathcal{L} is to determine whether two arbitrary expressions e and f of \mathcal{L} , with the same arity, have the same value on any structure of \mathcal{L} (denoted by $e \equiv f$). Since $e \equiv f$ holds iff $e \subset f \wedge f \subset e$, the tautology problem for SDs alone is undecidable. By a more elaborate technique, the tautology problem for FDs alone can also be reduced to the equivalence problem.

By the above observations there is no procedure that receives as input any wff of any FD-SD language and always stops with 'YES', if P is a tautology, and 'NO', otherwise. In particular, the procedure described in this section may fail to stop even if P involves only FDs over relations and SDs over projections (just consider

$$r[A] \subset r[B] \wedge r: A \rightarrow B \Rightarrow r[B] \subset r[A],$$

which is true in every finite structure, but not a tautology).

5. Conclusions and directions for future research

This paper described a formal system for FDs and SDs over expressions and an associated proof procedure based on the analytic tableaux method. The simplicity of the system, as compared to that in [14] for FDs alone, derives from the explicit use of tuples of constants and equality. In fact, our earlier experience indicates that, without this device, FDs and SDs have to be generalized into

complicated dependencies, least searching for a complete system becomes a very difficult, if not hopeless task.

The analytic tableaux method proved to be quite attractive and easy to use manually. However, it would be reasonable inefficient to implement the method as we described. This is complicated by the fact that the tautology problem for FD-SD languages is undecidable. Hence, fast decision procedures or at least efficient heuristics for reducts of the full problem should be sought.

Finally, we observe that the completeness result of Section 4 generalizes to other types of dependencies, if the induction carried out to prove Lemma 4.2 still holds.

References

- [1] W.W. Armstrong, Dependency structures of database relationships, Proc. IFIP 74 (1974) 580–583.
- [2] C. Beeri, P.A. Bernstein and N. Goodman, A sophisticated introduction to database normalization theory, Proc. 5th Internat. Conf. on Very Large Databases (1978) 1–12.
- [3] C. Beeri, R. Fagin and J.D. Howard, A complete axiomatization for functional and multivalued dependencies, Proc. ACM SIGMOD Conf. (1977) 47–61.
- [4] M.A. Casanova, The theory of functional and subset dependencies over relational expressions, Tech. Rept., Dept. of Informatics, Pontificia Universidade Catolica do RJ, Rio de Janeiro.
- [5] M.A. Casanova and P.A. Bernstein, A formal system for reasoning about programs accessing a relational database, ACM TOPLAS 2(2) (1980).
- [6] E.F. Codd, Further normalization of the data base relational model, in: R. Rustin, ed., Data Base System (Prentice-Hall, Englewood Cliffs, NJ, 1972).
- [7] E.F. Codd, Extending the database relational model to capture more meaning, ACM TODS 4(4) (1980).
- [8] C.J. Date, An Introduction to Database Systems (Addison-Wesley, Reading, MA, 2nd ed., 1977).
- [9] H.B. Enderton, A Mathematical Introduction to Logic (Academic Press, New York, 1972).
- [10] R. Fagin, Multivalued dependencies and a new normal form for relational databases, ACM TODS 2(3) (1977).
- [11] B. Jacobs, Applications of database logic to automatic program conversion, Dept. of Computer Science, Univ. of Maryland.
- [12] B. Jacobs and A. Klug, On interpretations of relational languages and solutions to the implied constraint problem, Dept. of Computer Science, Univ. of Maryland.
- [13] A. Klug, Entity-relationship views over uninterpreted enterprise schemas, Proc. Internat. Conf. on Entity-Relationship Approach to Systems Analysis and Design (1979) 52–72.

- [14] A. Klug, Calculating constraints on relational expressions, ACM TODS 5(3) (1980).
- [15] A.O. Mendelzon and D. Maier, Generalized mutual dependencies and the decomposition of database relations, Proc. 5th Internat. Conf. on Very Large Databases (1978) 360-367.
- [16] 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.
- [17] J. Rissanen, Independent components of relations, ACM TODS 2(2) (1977) 317-325.
- [18] N. Rescher and Urquhart, Temporal Logic (Springer, Berlin, 1971).
- [19] J.R. Shoenfield, Mathematical Logic (Addison-Wesley, Reading, MA, 1967).
- [20] R.M. Smullyan, First-Order Logic (Springer, Berlin, 1971).
- [21] F. Sadri and J.D Ullman, A complete axiomatization for a large class of dependencies in relational databases, ACM Symp. on the Theory of Computing, to appear.
- [22] Y. Sagiv and S. Walecka, Subset dependencies as an alternative to embedded multivalued dependencies, Tech. Rept. UIUCDCS-R-79-980, Univ. of Illinois at Urbana-Champaign, 1979.
- [23] G. Wiederhold and R.El-Masri, A structural model for database systems, Tech. Rept. STAN-CS-79-722, Stanford University, 1979.