PUC

Series: Monografias em Ciência da Computação

Nº 3/81

THE THEORY OF FUNCTIONAL AND SUBSET DEPENDENCIES OVER RELATIONAL EXPRESSIONS

by

Marco A. Casanova

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP-22453

RIO DE JANEIRO - BRASIL

Series: Monografias em Ciência da Computação

Nº 3/81

Series Editor: Marco A. Casanova Janeiro 1981

THE THEORY OF FUNCTIONAL AND SUBSET DEPENDENCIES OVER RELATIONAL EXPRESSIONS*

by

Marco A. Casanova

* Rsearch supported by the Brazilian Government Agency FINEP

ABSTRACT:

A formal system for reasoning about functional dependencies (FDs) and subset dependecies (SDs) defined over relational expressions is described. An FD e:X→Y indicates that Y is functionally dependent on X in the relation denoted by expression e; an SD e ⊂ 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.

Key-Words: relational databases, schema design, functional dependencies, subset dependencies, subschema constraints

RESUMO:

Um sistema formal para dependências funcionais (DFs) e dependências de subconjunto (DSs) definidas sobre expressões relacionais é descrito. Uma DF e: X→Y indica que Y é funcionalmente dependente em X na relação denotada por e; uma DS e ⊂ f indica que a relação denotada por e é um subconjunto daquela denotada por f. Prova-se que o sistema é consistente e completo recorrendo-se ao método dos tableaux analíticos. Aplicações do sistema inluem o problema de determinar se uma restrição de integridade de um subesquema é consequência lógica das restrições do esquema, e o desenvolvimento de metodologias para projeto de banco de dados similares a normalização.

Palavras-chave: banco de dados relacionais, projeto de subesquemas, dependências funcionais, dependências de subconjunto, restrições de integridade em subesquemas.

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.

We view System S as a restriction of first-order logic to formulas of very special types, the FDs and SDs. Therefore, proofs in S tend to be shorter than their counter-parts in first-order logic. The consistency and completeness of S are obtained by resorting to the analytic tableaux method, along the lines of [Sm]. This method is quite attractive and can be used to obtain results about other families of dependencies.

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 $r_1,\ldots,r_n$ defined by expressions $e_1,\ldots,e_n$ (involving only 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 $e_1,\ldots,e_n$ iff $e_i:X \rightarrow Y$ is a logical consequence of the constraints of σ. This last assertion can then be resolved in S.

In general, when relations of a schema σ' are defined in terms of those of another schema σ, a situation similar to the subschema constraint problem may occur. For example, in a database restructuring operation, each relation

name r_i of σ' is associated with a defining expression e_i in σ $(1 \leq i \leq n)$, which indicates how to create a state I' of σ' from the current state I of σ. If I' is required to be consistent, the subschema constraint problem must then be faced (indeed σ' behaves as a subschema of σ).

System S also provides the theoretical background for a database design methodology that may eventually replace normalization. This observation is justified as follows. Some of the work on normalization depends on a theory of FDs based on the universal relation assumption (see [BBG]), which requires that all relations in a database be projections of a single (universal) relation. This assumption helps formalize the notion of schema equivalence and it assures the adequacy of the inference rules of FDs [Ar]. However, it does not usually hold in practice and is incompatible with some of the purposes of normalization [BG]. Subset dependencies replace this assumption in the sense that they permit to selectively define what data must be duplicated in what relations. This is specially useful when a relation is split into several relations by the design process. Therefore, the theory of FDs under the universal relation assumption could be successifully substituted by a theory of FDs and SDs.

The subschema constraint problem was studied in [K12], which considered only FDs defined over relational expressions. The general schema mapping problem was investigated in [JK], assuming the relational model, and in [Ja2], under the setting of Database Logic [Ja1]. A comprehensive survey of normalization appears in [BBG]. Formal systems for reasoning about several classes of data dependencies over relations (not relational expressions) were extensively studied in the literature (see, e.g., [Ar] for FDs, [Fa] for MVDs, [BFH] for FDs and MVDs, [SU] for template dependencies and [MM] for mutual

dependencies). Subset dependencies were used in the models of [Kl1,WM] and a formal system for FDs and SDs over relations appears in [Ca] (SDs should not be confused with the homonymous dependencies considered in [SW]). Other examples of the use of analytic tableaux can be found in [Sm,Pr,RU].

This paper is organized as follows. Section 2 defines FDs and SDs over relational expressions. Section 3 precisely states the subschema constraint problem. Section 4 describes the formal system S and the analytic tableaux method. Section 5 proves the soundness and the completeness of S. Finally, Section 6 contains conclusions and directions for future 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). The formulas of an FD-SD language are essentially boolean combinations of FDs and SDs, although for technical reasons we allow other types of atomic formulas as well. An FD-SD language L contains the following symbols:

parameters

(1) relation names: for each $n>0$, there is a non-empty set of n-ary relation names;

(2) constant symbols: a non-empty set of symbols, distinct from the relation names;

logical symbols

(3) the usual connectives and special symbols: $\neg,\wedge,\vee,\Rightarrow,(,),[,],\rightarrow,\subset$

(4) equality among constants: $=$

(5) the usual relation operators: $\times,\cup,-$ and projection, restriction and selection

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, $T,U,V,X \in ATTR(n)$ 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 $e[U=Y]$ 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.

We also introduce the join $(e[X=Y]f)$ as an abbreviation for $(e \times f)[X=Y']$, where Y' is obtained by adding n to each element of Y, if e is an 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 $\bar{a}=\bar{b}$, $e(\bar{a})$, a functional dependency $e:X \to Y$ 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 $X,Y \in ATTR(n)$, $n>0$. A well-formed formula (wff) of 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.

In essence, we can talk about boolean combinations of FDs and SDs over relational expressions in L. We also included atomic formulas of the form $\bar{a}=\bar{b}$ and $e(\bar{a})$, which assert equality of tuples of constants and the existence of a tuple in the relation denoted by an expression, respectively. This device is necessary due to the nature of the rules of S, as we will see in Section 4.

A <u>structure</u> I with <u>domain</u> 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 k-ary tuple of constants $\bar{a}$, a k-ary tuple $I(\bar{a}) \in D_I^k$, $k>0$. I is extended to the relational expressions of 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) $I(e[U=Y]) = \{\bar{a} \mid \bar{a} \in I(e) \wedge \bar{a}_U=\bar{a}_Y\}$

(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) $I(e-f) = \{\bar{a} \mid \bar{a} \in I(e) \wedge \neg\bar{a} \in I(f)\}$

where $\bar{t}_Z$ denotes $(t_{Z_1},\ldots,t_{Z_k})$ and $\bar{t}\bar{u}$ denotes $(t_1,\ldots,t_n,u_1,\ldots,u_m)$, if $\bar{t} = (t_1,\ldots,t_n)$, $\bar{u} = (u_1,\ldots,u_m)$ and $Z = (Z_1,\ldots,Z_k) \in ATTR(n)$.

We now extend I to a boolean valuation of 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) $I(e:X\rightarrow Y)$ = true iff $\forall\bar{a}\forall\bar{b}(\bar{a} \in I(e) \wedge \bar{b} \in I(e) \wedge \bar{a}_X=\bar{b}_X \Rightarrow \bar{a}_Y=\bar{b}_Y)$

otherwise $I(e:X\rightarrow Y)$ = false

(4) $I(e \subset f)$ = true iff $\forall \bar{a} \exists \bar{b} (\bar{a} \in I(e) \Rightarrow \bar{b} \in I(f))$, otherwise $I(e \subset f)$ = false

(5) I is extended to the non-atomic wffs using the rules of Propositional Calculus

The meaning of the wffs of L should be clear, we only stress that $e: X \rightarrow Y$ is true in I iff the Y-columns of the relation denoted by e in I are functionally dependent on the X-columns. Likewise, $e \subset f$ is true in I iff $I(e)$ is a subset of $I(f)$.

We conclude our list of basic definitions by saying that a set P of wffs is satisfiable iff all wffs in P are true in some structure of L. A set P of wffs logically implies a wff P iff P is true in every structure of L where all wffs in P are true (written $P \models P$). Finally, 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).

3. THE SUBSCHEMA CONSTRAINT PROBLEM

Recall from the Introduction that 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. In this section we give a precise definition of this problem. The concepts we introduce have identical parallels in First-Order Logic, however we prefer to use the relational model terms for them.

We begin with a few definitions. A database schema is a pair $S = (L, C)$, where L is an FD-SA language and C is a set of wffs of L, the integrity

constraints of C. Usually the set of relation names of L is finite and relation columns are given names, but this need not concern us here. A structure I of L is called a state of S; if all wffs in C are true in I, I is called a consistent state of S.

A subschema of a schema $S = (L,C)$ is a triple $S' = (J,M,\mathcal{D})$ where $(M,\mathcal{D})$ is a schema and J maps each n-ary relation name r of M into an n-ary relational expression r^J of L, $n>0$, and each constant a of M into a constant a^J of L. All definitions related to schemas translate to S' via $(M,\mathcal{D})$. The function J can be used for two different purposes. First, given a structure I for L with domain D_I, we can construct a structure I^J for M as follows. The domain of I^J is D_I; for each relation name r of M, $I^J(r) = I(r^J)$; for each constant a of M, $I^J(a) = I(a^J)$. I^J is called the structure of M (or the state of S') induced by I. Second, given any wff P of M, we can construct a wff P^J of L by replacing each occurrence of a relation name r and each constant a of M by r^J and a^J, respectively. Note that P^J is well-defined because r and r^J have the same arity, FD-SD languages allow FDs and SDs over relational expressions and, if we replace r by r^J in an expression e, the result is still a relational expression. We can relate these two constructions by the following lemma.

LEMMA 3.1: For any structure I of L and any wff P of M,

$$I^J(P) = \text{true} \quad \text{iff} \quad I(P^J) = \text{true}$$

Sketch of Proof

Follows exactly as Lemma 27B in [En, p. 161]. □

We can now precisely define the subschema constraint problem as the question of determining if, given a schema $S = (L,C)$, a subschema $S' = (J,M,D)$ of S, and $Q \in D$, $\bigwedge_{P \in C} I(P) = \text{true}$ implies $I^J(Q) = \text{true}$, for any structure I of S. This formalization of the problem is not entirely satisfactory, though, because it involves formulas of two different languages (the constraints of S are wffs of L and Q is a wff of M). However, by Lemma 3.1 and using the notion of logical consequence, the subschema constraint problem is equivalent to determining if $C \models Q^J$. Note that Q^J and all wffs in C are wffs of L.

Finally, we observe that System S is relevant to the subschema constraint problem because, as shown in Sections 4 and 5, $C \models Q^J$ holds iff Q^J is a theorem of C in S. Hence, System S induces a syntactical characterization of the subschema constraint problem. We conclude this section with an example illustrating the concepts introduced.

EXAMPLE 3.1: Consider the suppliers-parts database of [Da, Chap. 4 and 9]. It can be described by a schema $S = (L,C)$, where L is an FD-SD language whose parameters include two ternary relation names SUPPLIER and SP, a binary relation name CS and a 5-ary relation name PART. To increase readability, we name relation columns via the following relation schemas:

(1) SUPPLIER[SN,SNAME,SCITY]

(2) CS[CITY,STATUS]

(3) PART[PN,PNAME,COLOR,WEIGHT,CITY]

(4) SP[SN,PN,QTY]

The set $\mathcal{C}$ of constraints of S consists of the following wffs (where column names replace column numbers):

(5) SUPPLIER:SN→SNAME,CITY

(6) CS:CITY→STATUS

(7) PART:PN→PNAME,COLOR,WEIGHT,CITY

(8) SP:SN,PN→QTY

(9) SP[SN]⊂SUPPLIER[SN]

(10) SP[PN]⊂PART[PN]

(11) SUPPLIER[SCITY]⊂CS[CITY]

The last three integrity constraints are not included in the original example. However, they reflect natural restrictions that must be present in the database description. We also define a subschema $S' = (\mathcal{J},\mathcal{M},\mathcal{D})$ of S, where $\mathcal{M}$ contains just one quaternary relation name C. The values assigned to C relate SN,SNAME,SCITY and STATUS. This is indicated by the following relation schema:

(12) C[SN,SNAME,SCITY,STATUS]

and is made precise by defining $\mathcal{J}$ so that

(13) $C^{\mathcal{J}}$ = (SUPPLIER[SCITY=CITY]CS)[SN,SNAME,SCITY,STATUS]

Finally, $\mathcal{D}$ consists of the single wff below:

(14) C:SN→SNAME,SCITY,STATUS

The above wff is valid in any state of S' induced by any consistent state of because we have that

(15) C ⊨ (SUPPLIER[SCITY=CITY]CS)[SN,SNAME,SCITY,STATUS]:SN→SNAME,SCITY,STATUS .

4. A FORMAL SYSTEM FOR REASONING ABOUT FUNCTIONAL AND SUBSET DEPENDENCIES

Let L be an FD-SD 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 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 [Sm]. 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, $\mathcal{P} \models P$ does not hold.

Reasoning by cases is captured by using rules of the following type R_i. $\dfrac{\mathcal{P}_i}{\mathcal{Q}_{i1} \mid \cdots \mid \mathcal{Q}_{in_i}}$ where $\mathcal{P}_i$ and $\mathcal{Q}_{ij}$ $(1 \leq j \leq n_i)$ are finite sets of wffs. Intuitively, 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 R_i and $\mathcal{Q}_{i1}, \ldots, \mathcal{Q}_{in_i}$, the <u>consequents</u> of 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 4.1:

(a) The set of analytic tableaux for a set 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 P is an analytic tableau for P;

(ii) Suppose that τ is an analytic tableaux for 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 $\lambda_j \subset Q_{ij}$ $(1\leq j\leq n_i)$.

(b) A set H of wffs is a Hintikka set iff no wff and its negation are in H and if there is a rule R_i with antecedent P_i and consequents $Q_{i1},\ldots,Q_{in_i}$ such that $P_i \neq \emptyset$ and $P_i \subset H$, then $Q_{ij} \subset H$, for some $j\varepsilon[1,n_i]$.

(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 P is a closed tableau for $P\cup\{\neg P\}$. In this case, P is a theorem of P in S (written $P \vdash P$). □

Before proceeding further, we illustrate the above definitions with an example from Propositional Calculus, which is actually a subsystem of System S. The rules we adopt for Propositional Calculus are the following ones [Sm]:

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
$\neg(P \wedge Q)$	$\neg P$	Q
$P \vee Q$	P	Q
$P \Rightarrow Q$	$\neg P$	Q

Table 4.2

The A-rules indicate that from A we can deduce both A_1 and A_2; the B-rules indicate that from B we can deduce either B_1 or B_2. For example, assuming $P \vee Q$, we have two cases to consider: either P is true or Q is true.

EXAMPLE 4.1: Let P be the following wff of Propositional Calculus:

(1) $(p \vee (q \wedge r)) \Rightarrow ((p \vee q) \wedge (q \vee r))$

We try to prove that P is a tautology by exhibiting a closed tableau starting with $\neg P$. We indicate the structure of a tableau either spacially or by resorting to Dewey notation and the closing of a branch by 'X':

1. $\neg((p \vee (q \wedge r)) \Rightarrow ((p \vee q) \wedge (q \vee r)))$

2. $(p \vee (q \wedge r)),\ \neg((p \vee q) \wedge (q \vee r))$

3. p | 4. $q \wedge r$

7. $q,\ r$

5. $\neg(p \vee q)$ | 6. $\neg(q \vee r)$ | 10. $\neg(p \vee q)$ | 11. $\neg(p \vee r)$

8. $\neg p,\ \neg q$ | 9. $\neg q,\ \neg r$ | 12. $\neg p,\ \neg q$ | 13. $\neg p,\ \neg r$

X | X | X

The above tableau was constructed following Definition 4.1(a) and using the rules of Propositional Calculus. Thus, for example, nodes 5 and 6 were appended as sons of node 3 using the B-rule $\dfrac{\neg(P \wedge Q)}{\neg P \mid \neg Q}$ with P, Q replaced by $(p \vee q)$ and $(q \vee r)$, respectively. That is, if we have $\neg((p \vee q) \wedge (p \vee r))$, we must consider two cases: $\neg(p \vee q)$ or $\neg(p \vee r)$.

All branchs of the above tableau are closed, except the branch ending on node 9. However, this branch is complete, since it forms a Hintikka set. That is, the above tableau is complete, but not closed, which indicates that we have exhaustively applied all rules without arriving at contradictions in all possible cases. Therefore, the above tableau is not a proof that P is a tautology.

But, on the other hand, the open branch indicates how to construct a counter-example to P: just assign true to p and false to q and r (since p, $\neg q$ and $\neg r$ occur in that branch). Hence, the above tableau in fact indicates that P is not a tautology.

These remarks will be generalized in Section 5 into a completeness proof for S. □

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 $t = (t_1,\ldots,t_n,t_{n+1},\ldots,t_{n+m})$, then $t_{[1,n]}$ denotes $(t_1,\ldots,t_n)$ and $t_{[n+1,n+m]}$ denotes $(t_{n+1},\ldots,t_{n+m})$.

FD-rules:

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

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

SD-rules:

$\neg$SD. $\dfrac{\neg e\subset f}{e(\bar{a}),\neg f(\bar{a})}$ — $\bar{a}$ is a tuple of new constants

SD. $\dfrac{e(\bar{a}),e\subset f}{f(\bar{a})}$ — $\bar{a}$ is any tuple of constants

Projection Rules

$\neg$PR. $\dfrac{\neg e[X](\bar{a}),e(\bar{b})}{\neg\bar{b}_X=\bar{a}}$ — $\bar{a},\bar{b}$ are any tuples of constants

PR. $\dfrac{e[X](\bar{a})}{e(\bar{b}),\bar{b}_X=\bar{a}}$ — $\bar{a}$ is any tuple of constants and $\bar{b}$ is a tuple of new constants

Restriction rules

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

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

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

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

Union rules

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

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

Difference rules

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

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

Equality rules

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

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

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

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

A-rules and B-rules:

As for Propositional Calculus.

Intuitively, the rules of S capture patterns of reasoning commonly used in Mathematics. Rule ¬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, ¬SD and SD are similarly explained. The rules for relational expressions reflect directly previous definitions. Finally, the equality, the A-rules and the B-rules are quite familiar, except for rules EP and ¬EP, that 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 many-sorted first-order logic. Following [CB, En. Chap. 4], we select a many-sorted language M with an individual sort (for the constants of L) and an n-predicate sort, for each n>0 (for the n-ary relation names of L). M also contains a predicate symbol ε^n, for each n>0, of sort (n-predicate,ind,...,ind); $\varepsilon^n(r,\bar{a})$ indicates that the tuple denoted by $\bar{a}$ is in the n-ary relation denoted by r. The relational algebra operators may also be added as function symbols since they denote functions mapping relations into relations. Hence, we may translate wffs of L into equivalent wffs of M.

As an example, we indicate how to derive rules SD and ¬SD. Consider the following translation of the definition of $e \subset f$ to M:

(1) $e \subset f \equiv \forall\bar{a}(\varepsilon^n(e,\bar{a}) \Rightarrow \varepsilon^n(f,\bar{a}))$

Then, (1) is equivalent to the following pair of wffs:

(2) $\forall\bar{a}(e \subset f \wedge \varepsilon^n(e,\bar{a}) \Rightarrow \varepsilon^n(f,\bar{a}))$

(3) $\exists\bar{a}(\neg e \subset f \Rightarrow \varepsilon^n(e,\bar{a}) \wedge \neg\varepsilon^n(f,\bar{a}))$

By applying the usual rules of first-order logic (based on tableaux [Sm]), (2) generates rule SD and (3) generates rule ¬SD.

One may conclude from the above discussion that System S is superfluous, since its rules can be viewed as derived rules of first-order logic. However, we observe that derivations in S tend to be much shorter because the rules of S hide quantifiers and certain standard derivations pertaining to FDs and SDs. Moreover, by the results in Section 5, the rules of S suffice to check if a wff of L is a logical consequence of a set of wffs of L.

We may augment S with derived rules that help obtain even shorter proofs. A possible set is given below:

F1. $\dfrac{\quad}{e:X\rightarrow Y}$ $\quad Y \subset X$ $\qquad$ S1. $\dfrac{\quad}{e \subset e}$

F2. $\dfrac{e:X\rightarrow Y, e:Y\rightarrow Z}{e:X\rightarrow Z}$ $\qquad$ S2. $\dfrac{e \subset f, f \subset g}{e \subset g}$

F3. $\dfrac{e:X\rightarrow Y}{e:XZ\rightarrow YW}$ $\quad W \subset Z$ $\qquad$ S3. $\dfrac{e \subset f, f:X\rightarrow Y}{e:X\rightarrow Y}$

where $Y \subset X$ indicates that all integers in tuple Y also occur in X, and similarly for $W \subset Z$. Rules F1, F2 and F3 reflect the usual FD rules; rules S1 and S2 express the reflexivity and transitivity of SDs; finally, rule S3 says that we can transfer an FD from one expression to another via an SD.

Additional rules involving relational expressions may also be introduced:

PR' $\dfrac{e(\bar{a})}{e[X](\bar{a}_X)}$ RE' $\dfrac{e(\bar{a}),\bar{a}_X=\bar{a}_Z}{e[X=Z](\bar{a})}$

SE' $\dfrac{e(\bar{a}),\bar{a}_X=\bar{d}}{e[X=\bar{d}](\bar{a})}$ PT' $\dfrac{e(\bar{a}),f(\bar{b})}{(e\times f)(\bar{a}\bar{b})}$

UN' $\dfrac{e(\bar{a})}{(e\cup f)(\bar{a}),(f\cup e)(\bar{a})}$ DI' $\dfrac{e(\bar{a}),\neg f(\bar{a})}{(e-f)(\bar{a})}$

these rules may be derived directly from rules ¬PR, ¬RE, ¬SE, ¬PT, ¬UN and ¬DI, respectively.

We close this section with examples illustrating the use of S.

EXAMPLE 4.2: We prove that P is a tautology, where P is the following wff (r and s are ternary relation names):

(1) $r:1\to 2 \wedge s[2,3]\subset r[1,2] \Rightarrow s:2\to 3$

That is, we exhibit a closed tableau starting with ¬P. We first observe that for wffs of the form $P_1\wedge\ldots\wedge P_n\Rightarrow Q$ we may start the tableau directly with $\{P_1,\ldots,P_n,\neg Q\}$. The tableau goes as follows:

1. $r:1\to 2,\ s[2,3]\subset r[1,2],\ \neg s:2\to 3$
2. $s(a_1,a_2,a_3),\ s(a_1',a_2',a_3'),\ a_2=a_2',\ \neg a_3=a_3'$. 1,¬FD
3. $s[2,3](a_2,a_3),\ s[2,3](a_2',a_3')$. 2, PR'
4. $r[1,2](a_2,a_3),\ r[1,2](a_2',a_3')$. 1,3,SD
5. $r(b_1,b_2,b_3),\ b_1=a_2,\ b_2=a_3$. 4, PR
6. $r(b_1',b_2',b_3'),\ b_1'=a_2',\ b_2'=a_3'$. 4, PR
7. $b_1=b_1'$. 2,5,6,ET
8. $b_2=b_2'$. 1,5,6,7,FD
9. $a_3=a_3'$. 5,6,8,ET

X

. □

EXAMPLE 4.3: 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 a relation name r into X,Y,Z, if r:X→Y or r:X→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) $r{:}X\to Y \vee r{:}X\to Z \Rightarrow ((r[XY] \times r[XZ])[X=X'])[XYZ'] \subset r$

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

Following the conventions introduced in Examples 4.1 and 4.2, we offer the following closed tableau as a proof that Q is indeed a tautology:

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

EXAMPLE 4.4: We prove in this example the claim made at the end of Example 3.1. It suffices to show that the following wff (with the join already translated)

(1) e:SN→SNAME,SCITY,STATUS

with e = ((SUPPLIER×CS)[SCITY=CITY])[SN,SNAME,SCITY,STATUS]

is a theorem of the following set of wffs

(2) SUPPLIER:SN→SNAME,SCITY, CS:CITY→STATUS

We offer the following tableau as a proof:

1.	SUPPLIER:SN→SNAME,SCITY , CS:CITY→STATUS , ¬e:SN→SNAME,SCITY,STATUS	
2.	e(s,n,sc,st), e(s',n',sc',st'), s=s',¬(n,sc,st)=(n',sc',st')	. 1,¬FD
3.	f(s,n,sc,c,st), f(s',n',sc',c',st')	. 2,PR
	with f = (SUPPLIER × 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 × 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	¬n=n' 10.2 ¬sc=sc' 10.3 ¬st=st'	. 2,¬EP
	X X X	

note: to save space, some steps such as 8. summarize several derivations. □

5. SOUDNESS 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). This assumption can be dropped along the lines of [Sh, Chap. 4.2].

The soundness of S follows trivially.

THEOREM 5.1: S is sound.

Sketch of Proof

The proof follows 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 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 $\neg P$. This formalizes the observations made in Example 4.1.

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 $\rho = \{(a,a)/a \in U\} \cup \{(a,b)/"a=b" \in H\}$. 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, $I(r) = \{(a_1^0,\ldots,a_n^0) \in E^n/"r(a_1,\ldots,a_n)" \in H\}$. 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 occurrences of $\rightarrow,\subset,\neg,\vee,\wedge,\Rightarrow$ and the relational operators 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 $P \in H$ be a wff of degree i.

Rather than proceeding with a detailed case analysis, we summarize all cases into the following case schemas:

case schema 1: P is either $\neg e{:}X \to Y$, $\neg e \subset f$, $e[X](\bar{a})$, $e[X=Z](\bar{a})$, $e[X=\bar{d}](\bar{a})$, $(e - f)(\bar{a})$ $(e \times f)(\bar{a})$, $\neg(e \cup f)(\bar{a})$, 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.

case schema 2: P is either $(e \cup f)(\bar{a})$ 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 $e{:}X \to Y$, $e \subset f$, $\neg e[X](\bar{a})$, $\neg e[X=Z](\bar{a})$, $\neg e[X=\bar{d}](\bar{a})$, $\neg(e \times f)(\bar{a})$ or $\neg(e - f)(\bar{a})$.

Suppose that P is false in I. Then, by construction of I, it is possible to find an application of a rule R with antecedent P and consequent $\{Q\}$ such that (i) $P \in P$, $P \subset H$ and all wffs in $P-\{P\}$ have degree lower than P and are true in I; (ii) Q has degree lower than P and is false in I. But, by definition of Hintikka set and since $P \subset H$, Q is also in H. By the induction hypothesis, Q is true in I. Contradiction.

(We note that the Equality rules contribute to the above case schemas only indirectly through the construction of I).

As an illustration, we prove the case that P is a negated FD $\neg e:X\rightarrow Y$. Since H is a Hintikka set (using rule $\neg$FD), there are tuples of constants $\bar{a}$ and $\bar{b}$ such that $e(\bar{a})$, $e(\bar{b})$, $\bar{a}_X=\bar{b}_X$ and $\neg\bar{a}_Y=\bar{b}_Y$ are in H. By the ind. hypothesis and since these wffs have degree less than $\neg e:X\rightarrow Y$, they are true in I. Therefore, by construction of I, $\bar{a}^0\in I(e)$, $\bar{b}^0\in I(e)$, $\bar{a}^0_X=\bar{b}^0_X$ and $\bar{a}^0_Y\neq\bar{b}^0_Y$. Hence, $\neg e:X\rightarrow Y$ is true in I.

This concludes the proof. □

In order to use Lemma 5.1 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 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 $\neg$FD, $\neg$SD and PR may be repeadly 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 so that rules are non-redundantly applied in a cyclical pattern.

In what follows we assume that the 29 rules of S are numbered from 0 to 28. Let P and Q be sets of wffs occurring in branchs β and γ (not necessarily distinct) of a tableau τ, respectively. We say that P is <u>higher than</u> Q iff some wff in P occurs in a node of β with level higher than the level of any node of γ containing a wff of 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 <u>higher than</u> $\bar{b}$ 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 R_i be the last rule that was considered for application in τ. Consider now for application rule R_j, $j = (i+1) \bmod 29$:

case 1: R_j is not ES.

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

case 2: 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 branchs 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 unsuccessifully 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 5.2:

(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.

Proof

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

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

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

We conclude this section with some observations about the decidability of the tautology problem for an FD-SD language L (i.e., the problem of determining if a wff of L is a tautology). This problem is undecidable because it can be trivially reduced to the equivalence problem for relational expressions of L, which is undecidable by Theorem 6 of [K12]. The equivalence problem for relational expressions of L is to determine whether two arbitrary expressions e and f of L, with the same arity, have the same value on any structure of 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 (Theorem 7 of [K12]).

By the above observations, there is no decision procedure for the tautology problem for FD-SD languages, that is, there is no procedure that receives as input a wff P of an 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, as the following example indicates.

EXAMPLE 5.1: Let P be the following wff

(1) $r[A] \subset r[B] \wedge r: A \rightarrow B \Rightarrow r[B] \subset r[A]$

If we try to prove that P is a tautology using the procedure for constructing systematic tableaux, we obtain the following (unbounded) tableau:

1. $r[A] \subset r[B]$, $r: A \rightarrow B$, $\neg r[B] \subset r[A]$
2. $r[B](a)$, $\neg r[A](a)$. 1,$\neg$SA
3. $r(\bar{b}^0)$, $\bar{b}^0_B = a$. 2,PR
4. $r[A](\bar{b}^0_A)$. 3,PR'
5. $r[B](\bar{b}^0_A)$. 1,4,SA
6. $r(\bar{b}^1)$, $\bar{b}^1_B = \bar{b}^0_A$. 5,PR
7. $r[A](\bar{b}^1_A)$. 6,PR'
8. $r[B](\bar{b}^1_A)$. 1,7,SA

⋮

(for simplicity, we replaced rule $\neg$PR by its equivalent PR').

There is a deeper reason why the procedure in question does not stop. We first observe that P is indeed true if the value of r is any finite relation. However, P is not a tautology. For example, if the value of r is $\mathcal{r} = \{(2i,i) / \ i \in \mathbb{N}\}$, then $r[A] \subset r[B]$ and $r: B \rightarrow A$ are both true, but not $r[B] \subset r[A]$

We now recall that the procedure for constructing systematic tableaux actually tries to build a structure where $\neg P$ is true. Hence, it has to run forever to find an infinite relation $\mathcal{R}$ such that, if r is given $\mathcal{R}$ as value, $\neg P$ becomes true. □

6. CONCLUSIONS

This paper described a formal system for reasoning about functional and subset dependencies over relational expressions and an associated proof procedure based on the analytic tableaux method. Subset dependencies were included because they express frequently used data dependencies that cannot be described by FDs alone. The basic motivation of this paper was provided by the subschema constraint problem, that is, the problem of determining if a constraint of a subschema is a logical consequence of the constraints of the base schema. But the system developed also provides the background for a database design methodology that may eventually replace normalization.

The analytic tableaux method proved to be quite attractive and easy to use manually. However, it may fail to stop, even for trivial, albeit pathological, wffs (see Example 5.1). This should not be viewed as a handicap of the method because the problem it tries to solve is indeed undecidable. Hence, decision procedures or at least reasonable heuristics for reducts of the full problem should be sought. It must be added that the procedure for constructing systematic tableaux is quite inneficient, since it requires bookkeeping sets of wffs already used as antecedents of each rule.

Another source of inefficiency is inherent to the way System S was set up. The rules of S depart from the traditional FD rules in that tuples of constants were explicitly used. The simplicity of the system, as compared to that in [K12] for FDs alone, derives exactly from this fact. However, proofs tend sometimes to be longer than what one would expect, although shorter than in first-order logic. We do not see an easy way to circumvent this problem, except by using derived rules whenever possible (c.f. Section 4).

In fact, our earlier experience in trying to develop a formal system for FDs over relations and SDs over projections was quite negative [Ca]. Whenever we tried to derive systems based on languages without equality and tuples of constants, we obtained systems such that:

(i) the languages involved generalizations of FDs and SDs which were not satisfactory;

(ii) the rules were difficult to describe and use, and they did not represent a significant improvement over System S.

Hence, we believe that System S provides an adequate tool to reason about FDs and SDs.

REFERENCES

[Ar] W.W.Armstrong."Dependency Structures of Database Relationships". Proc. IFIP 74 (1974),580-583

[BBG] C.Beeri, P.A.Bernstein, N.Goodman."A Sophisticate's Introduction to Database Normalization Theory". Proc. 5th Int. Conf. on Very Large Databases (1978),1-12

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

[BG] P.A.Bernstein, N.Goodman."What Does Boyce-Codd Normal Form Do?". Proc. 6th Int. Conf. on Very Large Databases (1980)

[Ca] M.A.Casanova,"Reasoning about Functional Dependencies and Subset Axioms" (submitted for publication)

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

[Co1] E.F.Codd."Further Normalization of the Data Base Relational Model". in Data Base System (R.Rustin, ed.), Prentice-Hall, N.J. (1972)

[Co2] E.F.Codd."Extending the Database Relational Model to Capture More Meaning". ACM TODS 4,4 (Dec. 1980)

[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)

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

[Ja1] B.Jacobs."On Database Logic". TR 737, Dept. of Computer Science, U. of Maryland (Dec. 1978)

[Ja2] B.Jacobs."Applications of Database Logic to Automatic Program Conversion". Dept. of Computer Science, U. of Maryland

[JK] B.Jacobs, A.Klug."On Interpretations of Relational Languages and Solutions to the Implied Constraint Problem". Dept. of Computer Science, U. of Maryland

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

[K12] A.Klug."Calculating Constraints on Relational Expressions". ACM TODS 5,3 (Sept. 1980)

[MM] A.O.Mendelzon, D.Maier."Generalized Mutual Dependencies and the Decomposition of Database Relations". Proc. 5th Int. Conf. on Very Large Databases (1978),360-367

[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

[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)

[Sh] J.R.Shoenfield."Mathematical Logic". Addison-Wesley (1967)

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

[SU] F.Sadri, J.D.Ullman."A Complete Axiomatization for a Large Class of Dependencies in Relational Databases". (to appear 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)

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