PUC

Series: Monografias em Ciência da Computação

Nº 12/81

A THEORY OF DATA 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 12/81

Editor: Marco Antônio Casanova Outubro, 1981

A THEORY OF DATA DEPENDENCIES OVER RELATIONAL EXPRESSIONS*

By

Marco A. Casanova

*Research supported in part by FINEP and CNPq grant 402090/80

ABSTRACT

A formal system for reasoning about implicational dependencies over relational expressions (IDEXs) is first described. The System is shown to be sound and complete by resorting to the analytic tableaux method. Then, a class of IDEXs is exhibited whose inference problem is solvable by the method.

RESUMO

Um sistema formal por depêndencias de implicação sobre expressões relacionais (IDEXs) é descrito. Em seguida, prova-se a consistência e completude do sistema recorrendo ao método dos tableaux analíticos. Finalmente, uma classe de IDEXs é exibida cujo problema de decisão é solúvel pelo método.

1. Introduction

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

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

(2) $(e(a,b,c), e(a,b',c') \rightarrow e(a,b',c))$

where e is a ternary relational expression. The first IDEX asserts that the FD $A \rightarrow B$ holds in the relation denoted by e and is abbreviated as $e: A \rightarrow B$. The second IDEX asserts that the MVD $A \twoheadrightarrow B|C$ holds in the relation denoted by e, and is abbreviated as $e: A \twoheadrightarrow B$.

IDEXs were chosen because they subsume (i.e, contain as special cases) all data dependencies defined in the literature(as far as we know), such as functional dependencies (FDs) [Col] , multivalued and embedded multivalued dependencies (MVDs, EMVDs) [Fal,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 [K&1, WM]).

The use of dependencies over relational expressions was motivated by various schema design problems discussed elsewhere [CCF] . As an example, consider the problem of determining if a constraint of a subschema σ' is valid is any state of σ' constructed

from a consistent state of the base schema σ [Kℓ2]. 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 $r_i: X \rightarrow Y$ be an FD of σ' that must hold for r_i. Then, the FD is valid in any state of σ' constructed from a consistent state of σ via $e_1,\ldots,e_n$ iff the FD $e_i: X \rightarrow Y$ is a logical consequence of the constraints of σ. Note that the last FD is defined over a relational expression.

The language of the formal system consists basically of boolean combinations of IDEXs and formulas of the form $e(\bar{c})$ or $\bar{c} = \bar{d}$, where e is a relational expression and $\bar{c},\bar{d}$ are tuples of constants. The use of other basic formulas, in addition to IDEXs, greatly contributed to the simplicity of the inference rules. Boolean combination of basic formulas were considered to enhance the expressive power of the language.

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].

It should be noted at this point that the inference problem for IDEXs is unsolvable. However, we exhibit a

decision procedure for a rich class of instances of the inference problem.

This paper is organized as follows. Section 2 defines the language of System S. Section 3 introduces the analytic tableaux method by way of examples. Section 4 describes the inference rules of S and the analytic tableaux method in detail. Section 5 proves the soundness and the completeness of S. Section 6 discusses the decision problem for IDEXs. Finally, Section 7 contains conclusions and directions for future research.

2. Implicational Dependency Languages

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

parameters

(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

logical symbols

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

(4) equality among constants: =

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

Note: the biconditional $\equiv$ is introduced by definition in the usual way.

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 [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 $e[U=V]$ 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 $(m+m)$-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

the intersection (e ∩ f) abbreviates e-(e-f)

Note that, following [Kl2] , 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 also 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 $\neg P$, $(P \wedge Q)$, $(P \vee Q)$, $(Q \Rightarrow Q)$, 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_j. 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 $I(r) \subseteq D_I^n$,n>0, and to each constant a, $I(a) \in D_I$ (if $\bar{a} = (a_1,\ldots,a_m)$, then $I(\bar{a})$ abbreviates $(I(a_1),\ldots,I(a_m))$). 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 $(t_1,..,t_n,u_1,\ldots,u_m)$, 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) $I((A_1,\ldots,A_n \rightarrow B))$ = 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 the 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. Loosily 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 $w_1,\ldots,w_k$ are the constants occurring visible in $A_1,\ldots,A_n$, $A'_i = A_i[\bar{x}/\bar{w}]$ and $B' = B[\bar{x}/\bar{w}]$. 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.

functions dependencies (FDs):

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

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

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

where e is an n-ary expression, $X_1,\ldots,X_k \in ATTR(n)$ are such that every $j \in [1,n]$ 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 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 $P \models P$). 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 P of dependencies in Σ, and any dependency P in Σ, if $P \mid P$.

We close this section with two examples illustrating the use of an ID language.

EXAMPLE 2.1: Examples of formulas involving defined dependencies and their translations are (r is ternary and s is binary):

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

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

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

(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.

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

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

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. □

EXAMPLE 2.2: Consider the suppliers-parts database of [Da, Chap. 4 and 9]. It can be described using an ID language L 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 of constraints 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.) .□

3. An Overview of the Analytic Tableaux Method

In this section, the analytic tableaux method is introduced from the perspective of database theory and compared with two other formalisms - the axiom system for FDs and MVDs of [BFH] and the chase procedure of [MMS, SU] . A series of gradually more complex examples will help compare the formalisms. Each example shows that a certain decomposition of a quaternary relation name r[ABCD] is lossless (to improve readability, we name the columns of r). The examples also emphasize the expressiveness of ID languages. We conclude with further examples illustrating the usefulness of the method.

We first consider the axiom system of [BFH] for FDs and MVDs over a relation, which is summarized in Figure 3.1. The system is one of the best known formalisms in database theory since it solves the inference problem for FDs and MVDs. It produces concise proofs and, in fact, it is the basis for fast decision procedures for the inference problem in question.

Figure 3.1

An Axiom System for FDs and MVDs

language: the set of FDs and MVDs over relations

axioms and rules:

FD1. $r: X \to Y$, $Y \subseteq X$

FD2. $\dfrac{r: X \to Y}{r: XW \to YZ}$

FD3. $\dfrac{r: X \to Y,\ r: Y \to Z}{r: X \to Z}$

MVD0. $\dfrac{r: X \twoheadrightarrow Y}{r: X \twoheadrightarrow Z}$, $X \cup Y \cup Z$ covers all attributes of U and $Y \cap Z \subseteq X$

MVD1. $r: X \twoheadrightarrow Y$, $Y \subseteq X$

MVD2. $\dfrac{r: X \twoheadrightarrow Y}{r: XW \twoheadrightarrow YZ}$, $Z \subseteq W$

MVD3: $\dfrac{r: X \twoheadrightarrow Y,\ r: Y \twoheadrightarrow Z}{r: X \twoheadrightarrow Z-Y}$

FD-MVD1. $\dfrac{r: X \to Y}{r: X \twoheadrightarrow Y}$

FD-MVD2: $\dfrac{r: X \twoheadrightarrow Z,\ r: Y \to Z'}{r: X \to Z'}$, $Z' \subseteq Z$ and $Y \cap Z = \emptyset$

note: to help future references, we used the notation introduced in Section 2 for FDs and MVDs, instead of the move familiar $X \to Y$ and $X \twoheadrightarrow Y$ notation.

EXAMPLE 3.1: As an example of the usefulness of the system, consider the decomposition of r into r[ABC] and r[AD] in the presence of either the FDs r: A→B and r: A→C or the FD r: A→D. We want to show that the decomposition is lossless. From Theorem 2 of [Fal], we know that the decomposition is lossless iff the MVD r: A↠BC holds. Hence we must show that

(1) $((r: A \rightarrow B \wedge r: A \rightarrow C) \vee r: A \rightarrow D) \models r: A \twoheadrightarrow BC$

The proof goes as follows:

Case 1: assume r: A→B ∧ r: A→C , Then, we have

1.	r: A → B	. assumption
2.	r: A → C	. assumption
3.	r: A → AB	. 1, FD2
4.	r: AB → BC	. 2, FD2
5.	r: A → BC	. 3,4 FD3
6.	r: A ↠ BC	. 5, FD-MVD1

Case 2: assume r: A→D. Then , we have

1.	r: A → D	. assumption
2.	r: A ↠ D	. 1, FD-MVD1
3.	r: A ↠ BC	. 2, MVD0

□

The axiom system for FDs and MVDs can be criticized on the grounds that its language lacks expressiveness. Strictly speaking, the problem posed in Example 3.1 cannot be expressed within the language of the system, though this objection was easily circumvented. However, there are lossless decompositions that can only be expressed using a generalization of MVDs, the join dependencies [Ri2, MMS].

To illustrate this last remark, consider the decomposition of r into r[AB], r[ACD]and r[ACD] in the presence of $r: A \rightarrow C$ and $r: B \rightarrow D$. The decomposition is lossless iff r satisfies the JD $r: \{AB, ACD, BCD\}$ [MMS] . Hence, we want to prove that

(2) $r: A \rightarrow C,\ r: B \rightarrow D \models r: \{AB, ACD, BCD\}$

Note that (2) is again not in the language of the system. But this time there is nothing we can do to rephrase (2) into the appropriate language (in fact, the decomposition in question cannot be expressed as a cascade of two-way lossless decompositions [ABU]).

It is an interesting open question whether there is a complete axiom system for reasoning about FDs and JDs over relations whose language includes just these dependencies [MMS] . However, there is a decision procedure for the inference problem for FDs and JDs over a single relation, called the chase method [MMS] . Very superficially, the chase method works as follows. Let Σ be a set of FDs and JDs and σ be an FD or JD. To decide if $\Sigma \models \sigma$, start with a table not satisfying σ and, for each $\delta \in \Sigma$, force the

table to satisfy δ . This process always terminates with a counter-example for $\Sigma \models \sigma$ or with a table implying that indeed σ holds, if so does every dependency in Σ . The following example illustrates the chase method.

EXAMPLE 3.2: Consider the problems of proving (2). We have to show that, for every quaternary relation $\mathcal{R}$ satisfying r: A → C and r: B→D then $\mathcal{R}$ also satisfies the JD r: { AB, ACD, BCD } . But the JD will be satisfied if, for any three tuples in $\mathcal{R}$ of the form (a,b,c',d'), (a,b',c,d) and (a',b,c,d), the tuple (a,b,c,d) is also in $\mathcal{R}$. So, let us construct a table (a "tableau" in the sense of [ABU, MMS]) of the form

r_1

A	B	C	D
a	b	c'	d'
a	b'	c	d
a'	b	c	c

Since the FD r: A→C must hold, we equate C-entries of tuples that agree on A-entries, obtaining the following modified table:

r_2

A	B	C	D
a	b	c	d'
a	b'	c	d
a'	b	c	d

As a rule, the unprimed ("distinguished" in the sense of [MMS]) constants prevail. Similarly, using the FD r: B→D we obtain

r_3

A	B	C	D
a	b	c	d
a	b'	c	d
a'	b	c	d

Hence, we may conclude that whenever r contains tuples of the form (a,b,c',d'), (a,b',c,d) and (a',b,c,d), the tuple (a,b,c,d) is also in r^*. That is, the JD r:{AD,ACD,BCD} is a logical consequence of r: A→C and r:B→D .□

The chase method, as described, applies only when all dependencies are FDs and JDs over the same relation. But it is not difficult to extend the method to other dependencies involving several relations. Even when relational expressions are involved we may imagine a table T_e associated with each expression e indicating which tuples are currently in the value of e. However, a difficulty arises when set difference is considered. For example, we would need a rule asserting that, if t is in T_e and e = f-g, then t is in T_f and t cannot be in T_g. This last fact is difficult to express in the chase method.

Another example might help at this point. 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 $s:A\rightarrow B$ and $t:A\rightarrow C$, we can reconstruct r as $(s*u) \cup (t*v)$. To prove this, it suffices to show that:

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

Concrete illustrations of horizontal and vertical fragmentation can be found in [SS] .

The problem posed in (3) can be solved using the analytic tableaux method for ID languages developed in Section 4. From the point of view of classic Mathematics the analytic tableaux method formalizes the following familiar strategy to prove that $P \models P$. Start with P and $\neg P$ and work out all possible cases. More precisely, organize the proof as a tree whose root contains P and $\neg P$ and is such that the sons of a node correspond to braching cases. A proof organized this way is called an analytic tableau . Terminate the proof when each branch either contains a contradiction (i.e, closes) or cannot be extended further without repetition (i.e, completes). If all branchs close, $P \cup \{ P \}$ is unsatisfiable and, hence, $P \models P$. If some branch completes without closing, $P \cup \{\neg P\}$ is satisfiable (this is the main lemma of Section 5) and, hence, $P \models P$ does not hold.

Although the construction of analytic tableaux for ID languages is governed by the inference rules shown in Section 4, we can exhibit a proof of (3) and explain it in intuitive terms.

EXAMPLE 3.3: Consider the problem of showing that

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

where $s = (r[A=1])[AB]$, $t = (r - r[A=1])[AC]$,
$u = (r[A=1])[ACD]$ and $v = (r - r[A=1])[ABD]$.

Using the analytic tableaux method, we obtain:

1. $s: A \rightarrow B,\ t: A \rightarrow C, \neg\ (s*u) \cup (t*v) \subseteq r$

2. $((s*u) \cup (t*v))(a,b,c,d), \neg r(a,b,c,d)$

Case 1	Case 2
3.1 $(s*u)\ (a,b,c,d)$	3.2 $(t*v)(a,b,c,d)$
4.1 $s(a,b),\ u(a,c,d)$	4.2 $t(a,c),\ v(a,b,d)$
5.1 $r(a,b',c,d),\ a=1$	5.2 $(r - r[A=1])(a,b,c'',d)$
6.1 $s(a,b')$	6.2 $r(a,b,c'',d),\ \neg r[A=1](a,b,c'',d)$
7.1 $b = b'$	7.2 $c=c''$
8.1 $r(a,b,c,d)$	8.2 $r(a,b,c,d)$
X	X

(A cross indicates a closed branch).

We now explain some of the steps. Step 1 assumes the antecedents and negates the conclusion. Step 2 is obtained from Step 1 by observing that, if the IND $(s*u) \cup (t*v) \subseteq r$ does not hold, then there must be a tuple (a,b,c,d) in the value of $(s*u) \cup (t*v)$ but not in the value of r. The proof then branchs into two cases: (a,b,c,d) is in $s*u$ or (a,b,c,d) is in $t*v$. Steps 3.1 to 6.1 and 3.2 to 6.2 follow by definition of relational expressions. For example, 6.2 follows from 5.2 by definition

of set difference. Step 7.1 follows from 4.1, 6.1 and the FD s: A→B (similarly for 7.2). Finally, since b = b' by 7.1 and r(a,b',c,d) by 5.1, we may conclude that r(a,b,c,d) (similarly for (8.2)). Hence, in both cases we obtain a contradiction. □

From the point of view of database theory, the analytic tableaux method is closely connected with the chase method, 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 in necessary, as we have seen, when set difference is considered (c.f. the derivation of 6.2 from 5.2 in Example 5.3).

Another closely connected observation is that the analytic tableaux method is driven by formal inference rules indicating how to introduce new formulas in a proof and when to branch into a case analysis. Similarly, the chase method is driven by rules describing how to construct and modify tables of constants and (presumably) by

other rules indicating when to start a case analysis.

To substantiate these remarks, we rework Example 3.2 using the analytic tableaux method.

EXAMPLE 3.4: We prove that the logical implication in (2) holds using the analytic tableaux method.

1. r: A→C, r: B→D, ¬r:{AB,ACD,BCD}
2. r(a,b,c',d'), r(a,b',c,d), r(a',b,c,d), ¬ r(a,b,c,d)
3. c = c'
4. d = d'
5. r(a,b,c,d)

X

Step 1 is obtained by assuming the antecedents and negating the conclusion. Step 2 follows from assuming ¬ r: { AB, ACD, BCD } . Steps 3 and 4 follow from 2 and the FDs in 1. Finally, from c=c', d=d' and r(a,b,c',d'), we conclude r(a,b,c,d). Contradiction. □

The major drawback of the analytic tableaux method lies in that it is difficult to implement. However, we view this problem as a consequence of the generality and degree of expressions of ID languages, rather than an intrinsic problem of the method. Indeed, if we restrict ourselves to a language containing only FDs, JDs, relational formulas and equalities among constants, the analytic tableaux method and the chase method behave very similarly, as

illustrated by Example 3.4. In fact, all these observations suggest that the analytic tableaux method may be implemented by a chase-like procedure.

In this section, we concentrated on the relation decomposition problem, but the framework we develop is also useful to investigate several other database design problem [CCF]. To illustrate this remark, we conclude this section with a brief discussion of two such problems.

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 $(A_1,\ldots,A_n \rightarrow B)$ 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 $(1 \leq i \leq n)$. For example, if P is the FD $r_i : X \rightarrow Y$, Q will be $e_i : X \rightarrow Y$; 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.

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]($\bar{a}$) => 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 P of wffs of L iff P is a theorem of 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 $P \models P$. Start with P and $\neg P$ and work out all possible cases. If every case leads to a contradiction, then $P \cup \{\neg P\}$ is unsatisfiable and, hence, $P \models P$. On the other hand, if the analysis of some case is exhausted without arriving at a contradiction, then $P \cup \{\neg P\}$ is satisfiable (this has to be proved, since it is not an immediate property of the system) and, hence, $P \models P$ does not hold.

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

$R_i \cdot \dfrac{P_i}{Q_{i1} \mid \ldots \mid Q_{in_i}}$ where P_i and Q_{ij} $(1 \leq j \leq n_i)$ are finite sets of wffs. Intuitively, R_i means that from P_i we can derive all wffs in Q_{ij}, for some $j \in [1, n_i]$. We call P_i the antecedent of R_i and $Q_{i1}, \ldots, Q_{in_i}$, the consequents 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 tableau 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 $\lambda_j \subseteq Q_{ij}$ $(1 \leq j \leq n_i)$

(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, distinct from rules ID and $\neg$PR, with antecedent P_i and consequents $Q_{i1},\ldots,Q_{in_i}$ 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 $(A_1,\ldots,A_n \rightarrow B) \in H$, where $w=(w_1,\ldots,w_k)$ are the constants occurring visible in $A_1,\ldots,A_n$, then for any tuple $\bar{a}=(a_1,\ldots,a_k)$ of constants in U 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, where $\bar{b}_x$ is equal to $\bar{a}$, $e(\bar{b}) \in H$.

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

(d) A branch of a tableau is <u>complete</u> 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 <u>closed</u> iff every branch is closed.

(f) A tableau is <u>complete</u> iff every branch is closed or some branch is 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 <u>theorem</u> of $\mathcal{P}$ in S (written $\mathcal{P} \vdash P$). □

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	$\neg Q$
$\neg P$	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

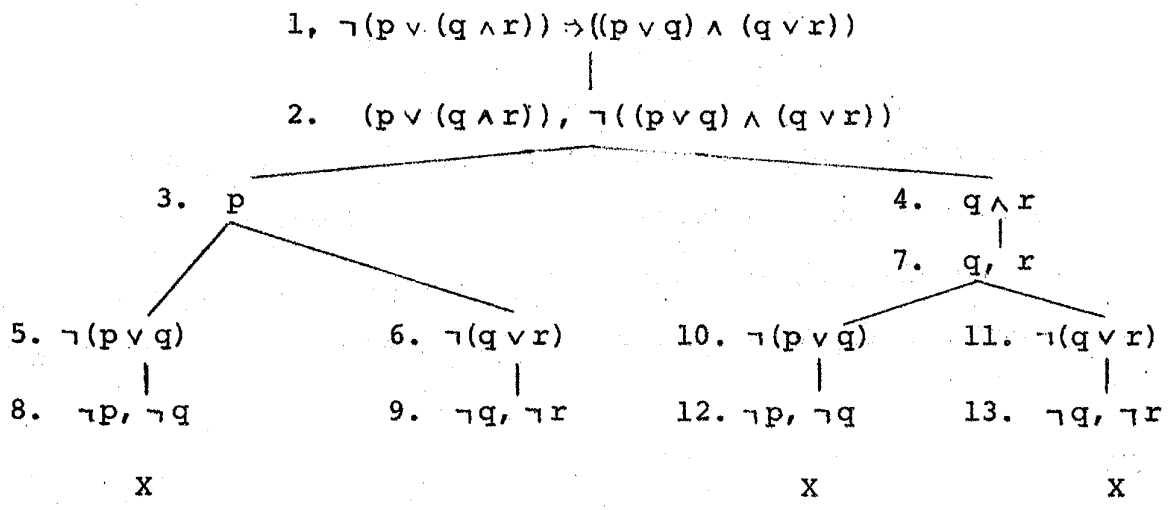
Table 4.2

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':



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 $\frac{\neg(P \wedge Q)}{\neg P \mid \neg Q}$ with P, Q replaced by $(p \vee q)$ and $(q \vee r)$, respectively. That is, if we have $\neg((p \vee q) \wedge (q \vee r))$, we must consider two cases: $\neg(p \vee q)$ or $\neg(q \vee r)$.

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. □

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 $(t_{n+1},\ldots,t_{n+m})$.

ID-rules:

$$\neg\text{ID}.\ \frac{\neg(A_1,\ldots,A_n \to B)}{A'_1,\ldots,A'_n,\ \neg B'} \qquad \bar{a} = (a_1,\ldots,a_k) \text{ are new constants}$$

$$\text{ID}.\ \frac{(A_1,\ldots,A_n \to B)}{\neg A'_1 \mid \ldots \mid \neg A'_n \mid B} \qquad \bar{a} = (a_1,\ldots,a_k) \text{ are any constants}$$

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

Projection Rules

$$\neg\text{PR}.\ \frac{\neg e[X](\bar{a})}{\neg e(\bar{b})} \qquad \bar{a},\ \bar{b} \text{ are any tuples of constants, except that } \bar{b}_X \text{ is equal to } \bar{a}$$

$$\text{PR}.\ \frac{e[X](\bar{a})}{e(\bar{b})} \qquad \bar{a} \text{ is any tuple of constants and } \bar{b} \text{ is a tuple of new constants, except that } \bar{b}_X \text{ is equal to } \bar{a}$$

Restriction rules

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

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

Selection rules

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

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

Product rules

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

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

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

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

note: we assume that e is n-ary and f is m-ary.

Union rules

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

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

Difference rules

$\neg$DI. $\dfrac{\neg(e-f)(\bar{a})}{\neg e(\bar{a}) \mid f(\bar{a})}$ DI. $\dfrac{(e-f)(\bar{a})}{e(\bar{a}), \neg f(\bar{a})}$ $\bar{a}$ is any tuple of constants

Equality rules

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

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

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

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

A-rules and B-rules : As for Propositional Calculus

Intuitively, the rules of S capture patterns of reasoning commonly used is Mathematics. Rule $\neg$ID when applied to $\neg(e(a,b,c), e(a,b',c') \rightarrow e(a,b',c))$, for example, captures the following pattern: "... Assume $\neg(e(a,b,c), e(a,b',c') \rightarrow e(a,b',c))$. Then, there must be two tuples in the value of e, (a_1, a_2, a_3) and (a_1, a_4, a_5), such that the tuple (a_1, a_4, a_3) is not in the value of e (where $a_1, \ldots, a_5$ have not been previously used). All other rules are similarly

explained using the definitions in Section 2.

An important feature of the above rules is that, except for the equality rules, the antecedent is always a formula more complex than any formula in the consequentes. That is, the system always breaks a formula into simpler formulas (hence, the adjective 'analytic').

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 ID and ¬ ID. Consider the following definition of $(A_1,\ldots,A_n \rightarrow B)$ in M

(1) $(A_1,\ldots,A_n \rightarrow B) \equiv \forall x_1 \ldots \forall x_k (A''_1 \wedge \ldots \wedge A''_n \Rightarrow B'')$

where $w_1,\ldots,w_k$ are the constants occurring visible in $A_1,\ldots,A_n$ and A''_i, B" are translations of A_i, B to M with w_j replaced by a variable x_j, $1 \leq i \leq n$, $1 \leq j \leq k$. Then, (1) is equivalent to the conjunction of:

(2) $\forall x_1 \ldots \forall x_k \; ((A_1,\ldots,A_n \rightarrow B) \Rightarrow (\neg A_1 \vee \ldots \vee \neg A_n \vee B))$

(3) $\exists x_1 \ldots \exists x_k \; (\neg((A_1,\ldots,A_n \rightarrow B) \Rightarrow (A_1 \wedge \ldots \wedge A_n \wedge \neg B))$

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

The rest of this section presents 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, MVDs, INDs and JDs , which were introduced by definition at the end of Section 2. In addition to the rules in Figure 3.1, we offer the following ones:

FD-rules:

$$\neg\text{FD.}\ \frac{\neg\, e{:}X\rightarrow Y}{e(\bar{a}),e(\bar{b}),\bar{a}_X=\bar{b}_X,\ \neg a_Y=b_Y} \qquad \bar{a},\bar{b} \text{ are tuples of new constants}$$

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

IND-rules:

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

$$\text{IND.}\quad \frac{e(\bar{a}),\ e \subseteq f}{f(\bar{a})} \qquad \bar{a} \text{ is any tuple of constants}$$

MVD-rules:

$$\neg\text{MVD.}\quad \frac{e:\ X \twoheadrightarrow Y}{e(\bar{a}),e(\bar{b}),\bar{c}_{XY}=\bar{a}_{XY},\bar{c}_{XZ}=\bar{b}_{XZ},\neg e(\bar{c})} \qquad \bar{a},\bar{b},\bar{c} \text{ new}$$

$$\text{MVD.}\quad \frac{e(\bar{a}),e(\bar{b}),\bar{c}_{XY}=\bar{a}_{XY},\bar{c}_{XZ}=\bar{b}_{XZ},e:X \twoheadrightarrow Y}{e(\bar{c})} \qquad \bar{a},\bar{b},\bar{c} \text{ any}$$

JD-rules:

$$\neg\text{JD.}\quad \frac{e:\{X_1,\ldots,X_n\}}{e(\bar{a}^1),\ldots,e(\bar{a}^n),\bar{b}_{X_1}=\bar{a}^1_{X_1},\ldots,\bar{b}_{X_n}=\bar{a}^n_{X_n},\neg e(\bar{b})} \qquad \bar{a}^i,\bar{b} \text{ new}$$

$$\text{JD.}\quad \frac{e(\bar{a}^1),\ldots,e(\bar{a}^n),\bar{b}_{X_1}=\bar{a}^1_{X_1},\ldots,\bar{b}_{X_n}=\bar{a}^n_{X_n},e:\{X_1,\ldots,X_n\}}{e(\bar{b})} \qquad a^i,\bar{b} \text{ any}$$

Additional rules involving IDEXs and relational expressions may also be introduced:

$$\neg\text{ID'.}\quad \frac{A'_1,\ldots,A'_n,\ (A_1,\ldots,A_n \rightarrow B)}{B'}$$

where $A'_1,\ldots,A'_n,B'$ follow the conventions of rule ID.

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. The examples also reinforce how expressive ID languages are.

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

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

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. $r\colon 1\to 2, s[2,3]\subseteq r[1,2], \neg s\colon 2\to 3$
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, IND

5. $r(a_2,a_3,b_3)$. 4, PR

6. $r(a_2',a_3',b_3')$. 4, PR

7. $a_3 = a_3'$. 1,2,5,6,FD

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

where X',Z' are obtained by adding k to each element of X,Z, respectively , if r[XY] is a k-ary expression.

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. To simplify the notation, we rearrange the columns of r so that X=(1,...,i), Y=(i+1,...,i+j) and Z=(i+j+1,...,n), where n is the arity of r . This convention is used in Step 6 of the proof.

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	.□

EXAMPLE 4.4: 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→Y or r: X→Z. More formally, we have

(1) $F \vdash e \subseteq r$ implies $F \vdash (r: X\to Y \vee r: X\to Y)$ 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[XY] is a k-ary expression. Again, to simplify notation, we assume that X,Y,Z are as in Example 4.3 (this convention is used in Step 6 below). (Note that (1) is actually a metatheorem).

Proof

Assume $F \vdash e \subset r$. Then there is a closed tableau τ starting with $\delta_o = F \cup \{\neg e \subset r\}$. Moreover, τ can be constructed using rules ID, $\neg$IND and those for relational expressions. Since only rule $\neg$IND can be applied do δ_o, 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. $F, \neg e \subset r$
2. $e(\bar{a},\bar{b},\bar{c}), \neg r(\bar{a},\bar{b},\bar{c})$. 1, $\neg$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, 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. $r(\bar{a},\bar{b},\bar{z}), r(\bar{a}',\bar{b}',\bar{z}'), \bar{a}=\bar{a}', \neg\ \bar{b}=\bar{b}'$. 2,$\neg$FD
4. $r(\bar{x},\bar{y},\bar{c}), r(\bar{x}',\bar{y}',\bar{c}'), \bar{x}=\bar{x}', \neg\ \bar{c}=\bar{c}'$. 2,$\neg$FD

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

EXAMPLE 4.4: We prove in this example that the following wff

(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 and 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',cs'),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. Soudness 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.

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 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 $\neg P$. 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 occurrence of $\rightarrow, \neg, \vee, \wedge, \Rightarrow$ 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 $P \in H$ 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 into the following case schemas:

case schema 1: P is either $\neg(A_1,\ldots,A_n \rightarrow B)$, $e[X](\bar{a})$, $e[X=Z](\bar{a})$, $e[X=\bar{a}](\bar{a})$, $(e\times f)(\bar{a})$, $\neg(e \cup f)(\bar{a})$, $(e - f)(\bar{a})$, 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 $\neg(A_1,\ldots,A_n \rightarrow B)$. Let $\bar{w} = (w_1,\ldots,w_k)$ be the constants visible in $(A_1,\ldots,A_n \rightarrow B)$. Since H is a Hintikka set (using rule $\neg$ID), there are constants $\bar{a}=(a_1,\ldots,a_k)$ such that $A_i[\bar{a}/\bar{w}]$, $i\varepsilon[1,n]$, $\neg B[\bar{a}/\bar{w}]$ are in H. By the induction hypothesis and since these wffs have degree less than $(A_1,\ldots,A_n \rightarrow B)$, they are true in I. Therefore, $I(A'_1 \wedge \ldots \wedge A'_n \Rightarrow B') =$ false. But this implies that $I((A_1,\ldots,A_n \rightarrow B)$ is false, by definition.

case schema 2: P is either $\neg e[X=Z](\bar{a})$, $\neg e[X=\bar{a}](\bar{a})$, $\neg(e\times f)(\bar{a})$, $(e \cup f)(\bar{a})$, $\neg(e-f)(\bar{a})$ 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 $i \varepsilon [1,2]$. 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 $(A_1,\ldots,A_n \rightarrow B)$ 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 $P \in H$, $Q(\bar{a}) \in H$ where $Q(\bar{a})$ is either $A_i[\bar{a}/\bar{w}]$, for some $i \in [1,n]$, 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 $\neg$ID,ID, $\neg$PR,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.

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 non-redundantly 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 $(a_1,\ldots,a_n)$ <u>is older than</u> $(b_1,\ldots,b_n)$ iff a_i is older than or equal to b_i, for each $i\varepsilon[1,n]$, and a_j is older than b_j, for some $j\varepsilon[1,n]$.

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 a 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 $\neg$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 $\neg$EI to derive new formulas not occurring in θ ; if Q is $\bar{a} = \bar{b}$ also extend θ with $\bar{a} = \bar{a}$ and $\bar{b} = \bar{b}$, if they do not occur in θ

Intuitively, we extend the tableau from the root down so that each wff is used exactly once as antecedent of a rule. The two special cases are explained as follows. Rules ES,ET,EI,$\neg$EI do not have exactly one antecedent, so they are treated separately. The difficult part concerns rules ID and $\neg$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. Hence, each application of rule ID prepares the way to the next application by repeating the antecedent (tagged as unused). Similar remarks apply to rule $\neg$PR. (Strictly speaking the tree thus generated is not a tableau , but it can always be transformed into one by deleting the repeated formulas)

There is another important feature of the procedure, though. When rules ID and $\neg$PR are applied, constants are selected <u>from those</u>

used in the branch being extended, not from the set of all constants. Hence, the termination conditions guarantee 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.

Before proceeding further, we give an example of a systematic tableau.

EXAMPLE 5.1: (Compare with Example 4.2).

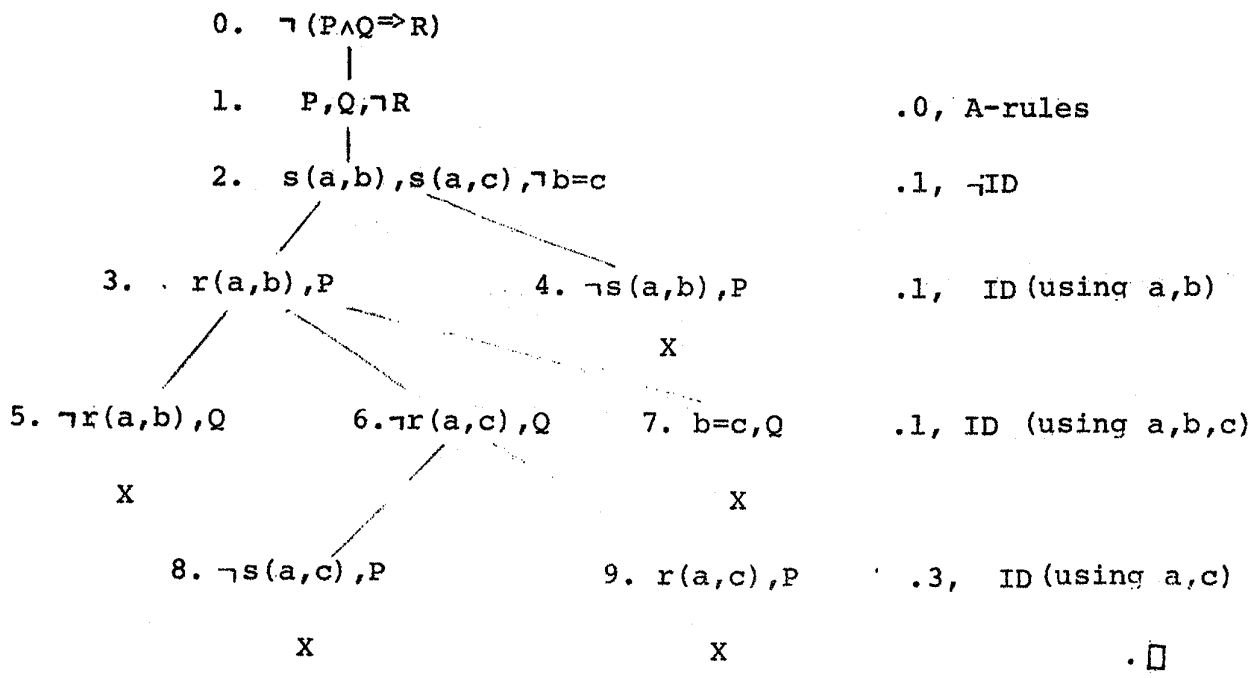
We prove that $P \wedge Q \Rightarrow R$ is a tautology, where

$P = (s(x,y) \rightarrow r(x,y))$ - the IND $s \subseteq r$

$Q = (r(x,y), r(x,z) \rightarrow y=z)$ - the FD r: $1 \rightarrow 2$

$R = (s(x,y), s(x,z) \rightarrow y=z)$ - the FD s: $1 \rightarrow 2$

The systematic tableau goes as follows:



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 $\neg P$ 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 $\neg P \in H$, $\neg P$ 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, $\models P \Rightarrow \vdash P$. □

6. Decidability Questions for ID Languages

In this section we discuss the inference problem for an ID language L, i.e, the problem of determining if $\mathcal{P} \models P$ holds, where $\mathcal{P}$ is a set of wffs of L and P is a wff of L. 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. Finally, we relate this class to a reduct of predicate calculus known to be decidable for satisfiability.

By the above observations, there is no decision procedure for the inference problem for ID languages, that is, there is no procedure that receives as input a set P of wffs and a wff P of an ID language and always stops with 'YES', if $P \models P$, and 'NO', otherwise. In particular, the procedure described in Section 5 may fail to stop even if only FDs over relations and INDs over projections are involved.

As an example, consider the following wff (call if W):

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

We first observe that W is true if the value of r is any finite relation. But W is not a tautology. For example, if the value of r is $\aleph = \{(2i,i)/i \in N\}$, then $r[A] \subseteq r[B]$ and $r: A \rightarrow B$ are both true, but not $r[B] \subseteq r[A]$. We now recall that the procedure for constructing systematic tableaux actually tries to build a structure where $\neg$ W is true. Hence, it has to run forever to find an infinite relation r such that, if r is given $\aleph$ as value, $\neg$ W becomes true.

The wff W also indicates that finite and infinite logical implication are not the same for IDEXs. That is, if we define $P \models_{fin} P$ to mean that P is true in every finite structure where all wffs in P are true, then it is not the case that $P \models_{fin} P$ iff $P \models P$.

In the rest of this section, we discuss classes of instances of the inference problem for which the analytic tableaux method (i.e., the refined procedure of Section 5) 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

(<=) Obvious, since rule PR is applied infinitely often in σ.

(⇒) 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 $\neg$ID in σ and at most as many applications of rule PR that are not consequences of applications of rules ID or $\neg$PR as there are projection operations occurring in wffs in P. Then, there are finitely many applications of rules $\neg$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 $\neg$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. □

From Lemma 6.1 we can easily obtain a class of instances of the inference problem which is decidable by the analytic tableaux method. Define an ID' language L exactly as an ID language, but without the projection operation. Then, we have.

THEOREM 6.1: The analytic tableaux method is a decision procedure for the inference problem for ID' languages.

Proof

Follows from Lemma 6.1, since rule PR is never applied in a proof for $P \models p$. □

COROLLARY 6.1: The analytic tableaux method is a decision procedure for the inference problem for the class of FDs, MVDs, INDs and JDs over relations (i.e., wffs of the form $r:X \rightarrow Y$, $r:X \twoheadrightarrow Y$, $r \subseteq s$, $r:\{X_1,\ldots,X_n\}$). □

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

(6) if P is $\neg Q$, then $i(p,P) = i(p,Q) + 1$

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

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

(9) if P is $(A_1,\ldots,A_n \rightarrow B)$ 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 <u>index</u> of an occurrence R of a subformula of a wff P as follows (i(R,P) counts negations prefixing R):

(11) if P is R then $i(R,P) = 0$

(12) if P is $\neg R$ then $i(R,P) = 1$

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

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

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

<u>note</u>: Recall that $A_1,\ldots,A_n$, B are all atomic formulas.

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 $(A_1,\ldots,A_n \rightarrow B)$ or $\neg f[X](\bar{a})$, 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 $\neg$ PR. But rule ID is applied iff a wff Q of the form $(A_1,\ldots,A_n \rightarrow B)$ 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) $r: X \rightarrow Y \vee r: X \rightarrow Z \models r[XY] * r[XZ] \subseteq r$

(2) $(r[A=1])[AB]: A \rightarrow B, (r-r[A=1])[AC]: A \rightarrow C$
$\models (r[A=1])[AB] * (r[A=1])[ACD] \cup$
$(r-r[A=1])[AC] * (r-r[A=1])[ABD] \subseteq r$

(3) $e_1: x_1 \rightarrow y_1, \ldots, e_n: x_n \rightarrow y_n \models e_0: x_0 \rightarrow y_0$,
where $e_0, \ldots, e_n$ are expressions that do not involve set difference.

We note this point that, by (3), our result then contains as a special case the main result in [Kl2]. In other words, our result extends the main result in [Kl2] 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 (compare with the last example of Section 3):

(4) $r: A \rightarrow B, r[A] \subseteq r[B] \models r[B] \subseteq r[A]$

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

There is nothing we can do about (4), as we explained at the beginning of this section. However, this is not the case with (5), since 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

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

This last example points out that we can expand the scope of Theorem 6.2 by pre-processing relational expressions in a systematic way.

In retrospect, we can explain the results obtained in terms of first-order logic as follows. Let $P \models P$ be an instance of the inference problem for ID languages satisfying the conditions of Theorem 6.2. We know that this problem is equivalent to testing if $\models Q$, where $Q = (P_1 \wedge \ldots \wedge P_n) \Rightarrow P$ and $P = \{P_1, \ldots, P_n\}$. If we rewrite Q as a first-order formula Q" in prenex normal form, the prefix of Q" will be $\exists^* \forall^*$. Moreover, Q" will have no function symbol other than constants. Hence, Q" belongs to the Bernays - Schönfinkel class, which was shown solvable for satisfiability in 1928 [Jo] (or, rather, to an extension of the class since our languages include equality).

We now observe that, loosily speaking, the analytic tableaux method essentially combines searching for a Herbrand interpretation satisfying Q" with searching for a proof for Q". Since the Herbrand universe of Q" is finite (it consists of a set of constants instantiating $\exists^*$), there are finitely many Herbrand interpretations for Q". This is, in principle, the reason why the procedure always stops when input with formulas such as Q.

To conclude, we observe that it might be fruitful to explore other solvable cases of the decision problem for predicate calculus. Formulas in some of these cases might have meaningful (to database theory) counterparts in ID languages.

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 subschema design problems discussed elsewhere [CCF].

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.

REFERENCES

[ABU] A.V. Aho, C. Beerim 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.

[CB] M.A.Casanova, P.A.Bernstein, "A Formal System for Reasoning about Programs Accesing 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" RC8757 , IBM T.J. Watson Research Center, Yorktown Heights, N.Y. (March 1981)

[COL] E.F.Codd. "Further Normalization of the Data Base Relational Model". in Data Base System (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 Normal 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).

[Jo] Joyner, W. "Resolution Strategies as Decision Procedures" JACM 23,3 (July, 1976), 398-417.

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

[Kl2] A.Klug. "Calculating Constraints on Relational Expressions". ACM TODS 5,3 (Sept. 1980)

[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 of 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).