Equational Theories and Database Constraints

Stavros S. Cosmadakis

Paris C. Kanellakis[1]

Laboratory for Computer Science, MIT

Abstract

We present a novel way to formulate database dependencies as sentences of first-order logic, using equational statements instead of Horn clauses. Dependency implication is directly reduced to equational implication. Our approach is powerful enough to express functional and inclusion dependencies, which are the most common database constraints. We present a new proof procedure for these dependencies. We use our equational formulation to derive new upper and lower bounds for the complexity of their implication problems.

1. Introduction

In order to deal formally with the problems of logical database design and data processing, database theory models data as sets of tables (*relations*). These relations are required to satisfy integrity constraints (*dependencies*), which intend to capture the semantics of a particular application. Various kinds of dependencies have been proposed in the literature (see [25, 11] for reviews of the area). For example, a *functional dependency* (FD) is a formal statement of the form EMPLOYEE→SALARY, which intuitively states that every employee has a unique salary. An *inclusion dependency* (IND) is a statement of the form MANAGER⊆EMPLOYEE, which intuitively states that every manager is an employee (the more general IND MANAGER.MANAGER-SALARY⊆EMPLOYEE.EMPLOYEE-SALARY expresses also the fact that managers make the same salary as managers as they make as employees). FD's and IND's are the most common database constraints. A most general formulation of dependencies as sentences in first order logic (namely Horn clauses) was given in [11]. To handle the central computational problem of dependency *implication* a particular proof procedure was developed, the *chase* (see [25] for its wide applicability). Proof procedures for general data dependencies also appear in [26, 2, 3]. The chase was seen to be a special case of a classical theorem proving technique, namely *resolution* [2, 3].

Alternative methods for theorem proving have been developed in the context of *equational theories*. This is a fragment of first order logic which has attracted a lot of attention because of its wide applicability in areas such as applicative languages, interpreters, and data types. See [14] for a survey of the area.

Given the formulation of database constraints as first order sentences, one would expect database theory to have been influenced by the developments in equational theories. However, not only did this never happen, but a constant effort has been made to minimize the role of equality in data dependencies (*multivalued dependencies*, the most widely studied after FD's, do not involve equality). This is even more impressive in view of the fact that the best algorithm for *losslessness of joins*, a basic computational problem, was derived from an efficient algorithm for *congruence closure* [10], and the best algorithm for implication of FD's [1] can be seen directly as a special case of an algorithm of [18] for the *generator problem in finitely presented algebras*.

This paper is a first attempt to rectify this situation. We demonstrate that there is a close connection between dependencies and equational statements. This strongly suggests the possibility of using the tools of equational theories to handle implication of dependencies. We explain our transformation of IND and FD implication into equational implication in Section 3 (Theorem 1). This transformation vastly simplifies arguments about provability of dependencies (compared to arguments using the

[1] On leave from Brown University; supported partly by NSF grant MCS-8210830 and partly by ONR-DARPA grant N00014-83-K-0146, ARPA Order No. 4786.

chase), and enables us to prove a number of results on implication problems for FD's and IND's.

We illustrate our basic approach with an example: An FD $A \rightarrow B$ is transformed into a string equation $fa=b$, and an IND $CD \subseteq AB$ is transformed into the equations $ai=c$, $bi=d$. Now we can easily infer the equation $fc=d$: $fc=fai=bi=d$. This corresponds to inferring the FD $C \rightarrow D$. In general, proofs in equational theories have a clean combinatorial structure, due to the existence of a simple, intuitive proof system [4].

A number of results are known about FD and IND implication. For IND's alone and FD's alone we have *finite controllability*, i.e. implication and *finite* implication coincide. While FD implication is decidable in linear time [1], IND implication is PSPACE-complete [5]. Syntactic restrictions on the IND's simplify the implication problem: bounded width IND's [5] and *typed* IND's [6] have polynomial time implication problems. The problem becomes NP-complete for *acyclic* IND's [24, 9]. The combination of FD's and IND's is not finitely controllable [5], or even decidable [22, 7]. The combination of FD's and *unary* IND's is also not finitely controllable, but both implication and finite implication can be decided in polynomial time [16]. A fundamental difficulty with IND's is that the chase does not necessarily terminate, and even in special cases delicate analysis is required [15]. A proof procedure for FD and IND implication, which differs from the chase, is presented in [22]. Using a variant of this procedure, inference of unary FD's from typed IND's and *acyclic* unary FD's is shown decidable in [9]. The chase is guaranteed to terminate if the IND's are acyclic. Thus, acyclic IND's and FD's are finitely controllable and in exponential time; NP-hardness (even if the IND's are typed) is shown in [9]). Finally, if all possible typed IND's are present, we have a variant of the *universal instance assumption* [25] known as *pairwise consistency* [19].

Our results apply to generalizations of FD's called *coupled* FD's (CFD's). These statements can express the additional fact that two FD's represent the same function in the database. Using our central Theorem 1, we can show:

1. Coupled unary FD's and binary IND's are dual statements. This is a direct consequence of our transformation (Corollary 1.4, Section 3).
2. Completeness of a new proof procedure for CFD's and IND's. This procedure differs from the chase and the formal system in [22], and treats CFD's and IND's in a symmetric fashion (Theorem 2, Section 3).
3. FD and IND implication is undecidable, even with only two FD's (Theorem 3, Section 4).
4. An exponential lower bound for acyclic IND and FD implication. This considerably improves the NP-hardness lower bounds in [9] (Theorem 4, Section 4).
5. Completeness of a proof procedure for CFD implication from a set of CFD's and typed IND's. This generalizes the result in [9] and shows that the problem is decidable for acyclic CFD's (Theorem 5, Section 5).
6. Implication of unary FD's in the presence of pairwise consistency is undecidable. The proof uses a variant of the semidecision procedure from Theorem 5 and a rather involved reduction from the word problem for semigroups (Theorem 6, Section 5).

For finite implication we cannot use the full power of our equational technique. However, we can show:

7. The implication problem for acyclic FD's in the presence of pairwise consistency is finitely controllable (and thus our transformation is also meaningful in the finite case). This does not follow from Theorem 5; an entirely different proof technique has to be developed (Theorem 7, Section
8. A weaker version of our transformation can handle finite implication of FD's and unary IND's. The proof uses the formal system of [16] (Theorem 8, Section 6).

2. Definitions

2.1. Equational Theories

Let M be a set of symbols and ARITY a function from M to the nonnegative integers $\mathcal{N}$. The set of finite strings over M is M^*. Partition M into two sets:

$G=\{g \in M \mid \text{ARITY}(g)=0\}$ the *generators*,
$O=\{\theta \in M \mid \text{ARITY}(\theta)>0\}$ the *operators*.

Definition: $\mathcal{T}(M)$, the set of *terms* over M, is the smallest subset of M^* such that,

1) every g in G is a term,
2) if $\tau_1,\ldots,\tau_m$ are terms and θ is in O with $\text{ARITY}(\theta)=m$, then $\theta\tau_1\ldots\tau_m$ is a term.

A *subterm* of τ is a substring of τ, which is also a term. Let $V=\{x,x_1,x_2,\ldots\}$ be a set of *variables*. Then the set of terms over operators O and generators $G \cup V$ will be denoted by $\mathcal{T}^+(M)$. For terms $\tau_1,\ldots,\tau_k$ in $\mathcal{T}^+(M)$ we can define the substitution $\varphi=\{\, (x_i \leftarrow \tau_i) \mid 1 \leq i \leq k \,\}$ to be a function from $\mathcal{T}^+(M)$ to $\mathcal{T}^+(M)$. We use $\varphi(\tau)$ or $\tau[x_1/\tau_1,\ldots,x_k/\tau_k]$ for the result of replacing all occurences of variables x_i in term τ by term τ_i $(1 \leq i \leq k)$, where these changes are made simultaneously.

Definition: A binary relation $\approx$ on $\mathcal{T}(M)$ or $\mathcal{T}^+(M)$ is a *congruence* provided that,

1) $\approx$ is an equivalence relation,
2) if $\text{ARITY}(\theta)=m$ and $\tau_i \approx \tau_i'$ $(1 \leq i \leq m)$ then $\theta\tau_1\ldots\tau_m \approx \theta\tau_1'\ldots\tau_m'$.

An *equation* e is a string of the form $\tau=\tau'$, where τ,τ' are in $\mathcal{T}^+(M)$. We use the symbol E for a set of equations. We will be dealing with models for sets of equations, i.e., algebras. We consider each equation e as a sentence of first-order predicate calculus (with equality), where all the variables from V are *universally quantified*.

Definition: An *algebra* $\mathcal{A}=(A,F)$ is a pair, where A is a nonempty set and F a set of functions. Each f in F is a function from A^n to A, for some n in $\mathcal{N}$ which we call the *type(f)*.

Examples: (a) A *semigroup* $(A,\{+\})$ is an algebra with one associative binary operator, i.e., for all x,y,z in A $(x+y)+z=x+(y+z)$. An example of a semigroup is the algebra of the set of functions from $\mathcal{N}$ to $\mathcal{N}$, together with the composition operation. In semigroups we use ab instead of a+b and w.l.o.g. omit parentheses.
(b) $\mathcal{A}_M$ is an algebra with $A=\mathcal{T}(M)$. For each θ in O we define a function θ in F with $type(\theta)=\text{ARITY}(\theta)$; here we use the same symbol for the syntactic object θ and its interpretation. The function θ maps terms $\tau_1,...,\tau_m$ from $\mathcal{T}(M)$ to the term $\theta\tau_1...\tau_m$, (i.e., $\theta(\tau_1,...,\tau_m)=\theta\tau_1...\tau_m$). We will refer to $\mathcal{A}_M$ as the *free algebra* on M. From this example it is clear that we can without ambiguity use both $\theta\tau_1...\tau_m$ and $\theta(\tau_1,...,\tau_m)$ to denote the same term.
(c)Let $\approx$ be a congruence on $\mathcal{T}(M)$. Condition 2) of the congruence definition guarantees that the operations in O are well-defined on $\approx$-equivalence (or congruence) classes. Thus we can form a *quotient* algebra $\mathcal{T}(M)/\approx$ with domain $\{[\tau] \mid \tau$ in $\mathcal{T}(M)$, $[\tau]$ is the $\approx$-congruence class of $\tau\}$ and with functions corresponding to O's operators.
(d) Similar observations with (b) and (c) can be made for the set of terms $\mathcal{T}^+(M)$.

Implication: Let e be an equation and $\mathcal{A}$ an algebra. $\mathcal{A}$ satisfies e, or is a model for e, if e becomes true when its operators and nonvariable generators are interpreted as the functions of $\mathcal{A}$ and its variables take *any* values in $\mathcal{A}$'s domain. The class of all algebras which are models for a set of equations E is called a *variety* or an *equational class*. We say that E implies e ($E\models e$) if equation e is true in every model of E.

Definition: An *equational theory* is a set of equalities E (of terms over $\mathcal{T}^+(M)$), closed under *implication*.

We write $E\vdash e$, if there exists a finite proof of e starting from E and using only the following five rules:

$\tau=\tau$,
from $\tau_1=\tau_2$ *deduce* $\tau_2=\tau_1$,
from $\tau_1=\tau_2$ *and* $\tau_2=\tau_3$ *deduce* $\tau_1=\tau_3$,
from $\tau_i=\tau_i'$ $(1\leq i\leq m)$ *deduce* $\theta\tau_1...\tau_m=\theta\tau_1'...\tau_m'$ $(\text{ARITY}(\theta)=m)$,
from $\tau_1=\tau_2$ *deduce* $\varphi(\tau_1)=\varphi(\tau_2)$ (φ is any substitution).

Proposition 1: [4] $E\models\tau=\tau'$ iff $E\vdash\tau=\tau'$. ∎

Let Γ be a set of equations over terms in $\mathcal{T}(M)$ (i.e., containing no variables). Consider the equational theory consisting of all $\tau=\tau'$ such that, $\Gamma\models\tau=\tau'$. By Proposition 1 this theory induces a congruence $=_\Gamma$ on $\mathcal{T}(M)$, where $\tau=_\Gamma\tau'$ iff $\Gamma\models\tau=\tau'$. From example (c) above we see that this congruence naturally defines an algebra $\mathcal{T}(M)/=_\Gamma$. If Γ is a finite set $\mathcal{T}(M)/=_\Gamma$ is known as a *finitely presented algebra* [18].

2.2. Relational Database Theory

Let $\mathcal{U}$ be a finite set of *attributes* and $\mathcal{V}$ a countably infinite set of *values*, such that $\mathcal{U}\cap\mathcal{V}=\emptyset$. A *relation scheme* is an object R[U], where R is the *name* of the relation scheme and $U\subseteq\mathcal{U}$. A *tuple* t over U is a function from U to $\mathcal{V}$. Let A_i be an attribute in U and a_i a value, where $1\leq i\leq|U|$; if $t[A_i]=a_i$ then we represent tuple t over U as $a_1a_2...a_{|U|}$. We represent the restriction of tuple t on attributes $A_1...A_n$ of U as $t[A_1...A_n]$. A *relation* r over U (named R) is a (possibly infinite) nonempty set of tuples over U. A *database scheme* D is a finite set of relation schemes $\{R_1[U_1],...,R_q[U_q]\}$ and a *database* $d=\{r_1,...,r_q\}$ associates each relation scheme $R_i[U_i]$ in d with a relation r_i over U_i. A database is finite if all of its relations are finite. A database can be visualized as a set of tables, one for each relation, whose headers are the relation schemes (each column headed by an attribute), and whose rows are the tuples.

The logical constraints, which determine the set of legal databases, are called *database dependencies*. We will be examining two very common types of dependencies.

CFD $(R:A_1...A_n\rightarrow A,\ S:B_1,...,B_n\rightarrow B)$ is a *coupled functional dependency*. Relations r,s (named R,S respectively), satisfy this CFD if,
for tuples t_1, t_2 in r, $t_1[A_1...A_n]=t_2[A_1...A_n]$ implies $t_1[A]=t_2[A]$ and
for tuples t_1, t_2 in s, $t_1[B_1...B_n]=t_2[B_1...B_n]$ implies $t_1[B]=t_2[B]$ and
for tuples t_1 in r, t_2 in s, $t_1[A_1...A_n]=t_2[B_1...B_n]$ implies $t_1[A]=t_2[B]$.

If $R=S$, $A=B$, $A_1=B_1,...,A_n=B_n$ we have a *functional dependency* (FD). If $n=1$, i.e., single attribute left hand sides, then we have a *binary functional dependency* (b-FD). If for an FD we also have $n=1$ then we call the dependency a *unary functional dependency* (u-FD). Note that every u-FD is both a b-FD and an FD. For an FD we usually employ the less redundant notation $R:A_1...A_n\rightarrow A$.

IND $S:D_1...D_m\subseteq R:C_1...C_m$ is an *inclusion dependency*. Relations s,r (named S,R respectively) satisfy this IND if, for each tuple t in s, there is a tuple t_1 in r with $t_1[C_i]=t[D_i]$ for $1\leq i\leq m$. If $m=2$ we have a *binary inclusion dependency* (b-ID) and if $m=1$ a *unary inclusion dependency* (u-ID). Note that u-ID's are in fact special cases of b-ID's, since $S:D_1\subseteq R:C_1$ has the same meaning with $S:D_1D_1\subseteq R:C_1C_1$.

Equality of two columns headed by attributes A, B in a relation named R can be expressed as a special case of IND's or CFD's: either use a CFD, such as $(R:A\rightarrow A,\ R:A\rightarrow B)$, or use an IND, such as $R:AB\subseteq R:AA$. These dependencies are particularly illustrative of our analysis; we will use $A\equiv B$ to denote them.

Database Notation: We use a graph notation to represent an input database scheme D and set of dependencies Σ (*input schema*). We construct a labeled directed graph G_Σ (see Figure 1), which has exactly one node a_{ij} for each attribute A_i of each relation scheme R_j. Let $i=R_2:D_1...D_m \subseteq R_1:C_1...C_m$ be an IND in Σ. Then G_Σ contains m *black* arcs $(c_{11},d_{12}),...,(c_{m1},d_{m2})$; each arc labeled by the name i of the IND. Let $f=(R_1:A_1...A_n \rightarrow A_0,\ R_2:B_1...B_n \rightarrow B_0)$ be a CFD in Σ. Then G_Σ contains two groups of n *red* arcs $(a_{11},a_{01}),...,(a_{n1},a_{01})$ and $(b_{11},b_{01}),...,(b_{n1},b_{01})$; each group is labeled by the name f of the CFD and each group's arcs are ordered from 1 to n as listed above.

We also consider the following directed graphs I_Σ and F_Σ: I_Σ has one node for each relation scheme name in D and arc (R,S) if and only if G_Σ contains some black arc (RA,SB). F_Σ has one node for each attribute in D and arc (A,B) if and only if G_Σ contains some red arc (RA,RB). We now define special syntactic forms of input schemata:

Acyclic IND's: I_Σ is acyclic.

Acyclic CFD's: F_Σ is acyclic.

Typed IND's: The black arcs of G_Σ are all of the form (RA,SA) for relation names R, S and attribute A.

Typed IND's are between occurences of the same attribute names in different relation schemes. If we assume that all possible typed IND's are in the input schema, (i.e., with some abuse of notation $R:U \cap U' \subseteq S:U \cap U'$ for all $R[U],S[U']$ in database scheme D), then we have *pairwise consistency* PC(D).

Implication: We say that Σ implies σ ($\Sigma \models \sigma$) if, whenever a database d over scheme D satisfies Σ, it also satisfies σ. If we restrict ourselves to finite databases we have $\Sigma \models_{fin} \sigma$. Clearly if $\Sigma \models \sigma$ (*implication*) then $\Sigma \models_{fin} \sigma$ (*finite implication*), but the converse is not always true. Deciding implication of dependencies is a central problem in database theory. Since dependencies are sentences in first-order predicate calculus with equality, we have *proof procedures* for the implication problem (we denote proofs as $\Sigma \vdash \sigma$). A proof procedure is *sound* if when $\Sigma \vdash \sigma$ then $\Sigma \models \sigma$; and *complete* if it is sound and when $\Sigma \models \sigma$ then $\Sigma \vdash \sigma$, (similarly for finite implication). The standard complete proof procedure for database dependencies is the *chase*. The appropriate chase rules for our analysis are described in [15].

3. Database Constraints as Equations

Let Σ be a set of CFD's and IND's over a database scheme *D* and σ a CFD or IND. We will transform Σ into two sets of equations E_Σ and $\mathcal{E}_\Sigma$. We will show that $\Sigma \models \sigma$ iff $E_\Sigma \models E_\tau$ iff $\mathcal{E}_\Sigma \models \mathcal{E}_\tau$, for some sets of equations $E_\tau, \mathcal{E}_\tau$ whose form depends on Σ and σ. We assume that *D* only contains one relation scheme; this simplifies notation, and there is no loss of generality.

Transformation: From the dependencies in Σ construct the following sets of symbols,

$M_f = \{f_k |$ for each CFD with n attribute left-hand sides include one operator f_k of ARITY n$\}$,
$M_i = \{i_k |$ for each IND include one operator i_k of ARITY 1$\}$,
$M_a = \{a_k |$ for each attribute A_k include one operator a_k of ARITY 1$\}$,
$M_\alpha = \{\alpha_k |$ for each attribute A_k include one generator $\alpha_k\}$.

Now let $M = M_f \cup M_i \cup M_a \cup M_\alpha$ and $V=\{x,x_1,x_2,...\}$ be a set of variables. $\mathcal{T}^+(M_f)$ ($\mathcal{T}^+(M_i)$) are the sets of terms constructed using operators in M_f (M_i) and generators in V.

The set E_Σ consists of the following equations (presented both in string and parenthesized notation):

1) two equations for each $\sigma_k=(A_1...A_n \rightarrow A,\ B_1...B_n \rightarrow B)$: $f_k a_1 x ... a_n x = ax$ and $f_k b_1 x ... b_n x = bx$,
(or $f_k(a_1(x),...,a_n(x)) = a(x)$ and $f_k(b_1(x),...,b_n(x)) = b(x)$);

2) m equations for each $\sigma_k = B_1...B_m \subseteq A_1...A_m$: $a_1 i_k x = b_1 x$ and ... and $a_m i_k x = b_m x$,
(or $a_1(i_k(x)) = b_1(x)$ and ... and $a_m(i_k(x)) = b_m(x)$).

The set $\mathcal{E}_\Sigma$ consists of the following equations:

3) two equations for each $\sigma_k=(A_1...A_n \rightarrow A,\ B_1...B_n \rightarrow B)$: $f_k \alpha_1 ... \alpha_n = \alpha$ and $f_k \beta_1 ... \beta_n = \beta$,
(or $f_k(\alpha_1,...,\alpha_n) = \alpha$ and $f_k(\beta_1,...,\beta_n) = \beta$);

4) m equations for each $\sigma_k = B_1...B_m \subseteq A_1...A_m$: $i_k \alpha_1 = \beta_1$ and ... and $i_k \alpha_m = \beta_m$,
(or $i_k(\alpha_1) = \beta_1$ and ... and $i_k(\alpha_m) = \beta_m$),

5) for each pair of symbols f_p in M_f and i_q in M_i the equation $f_p i_q x_1 ... i_q x_n = i_q f_p x_1 ... x_n$ (ARITY$(f_p)=n$),
(or $f_p(i_q(x_1),...,i_q(x_n)) = i_q(f_p(x_1,...,x_n))$).

The transformation is illustrated in Figure 2. Note that in $\mathcal{E}_\Sigma$ only equations 5) contain variables. Equations 5) are *commutativity* conditions between f and i operators. We now present Theorem 1, which is central to our analysis.

Theorem 1: In each of the following three cases, (i),(ii),(iii) are equivalent.

$\equiv$ Case:
i) $\Sigma \models A \equiv B$
ii) $E_\Sigma \models ax = bx$
iii) $\mathcal{E}_\Sigma \models \alpha = \beta$.

CFD Case:
i) $\Sigma \models (A_1...A_n \rightarrow A,\ B_1...B_n \rightarrow B)$
ii) $E_\Sigma \models \tau[x_1/a_1x,...,x_n/a_nx] = ax$ and $\tau[x_1/b_1x,...,x_n/b_nx] = bx$, for some τ in $\mathcal{T}^+(M_f)$
iii) $\mathcal{E}_\Sigma \models \tau[x_1/\alpha_1,...,x_n/\alpha_n] = \alpha$ and $\tau[x_1/\beta_1,...,x_n/\beta_n] = \beta$, for some τ in $\mathcal{T}^+(M_f)$.

IND Case:

i) $\Sigma \models B_1...B_m \subseteq A_1...A_m$

ii) $E_\Sigma \models a_1\tau = b_1x$ and ... and $a_m\tau = b_mx$, for some τ in $\mathcal{T}^+(M_i)$

iii) $\mathcal{E}_\Sigma \models \tau[x/\alpha_1] = \beta_1$ and ... and $\tau[x/\alpha_m] = \beta_m$, for some τ in $\mathcal{T}^+(M_i)$.

Proof Sketch: We use E_τ ($\mathcal{E}_\tau$) to denote the set of equations corresponding to term τ in (ii),(iii).

(ii)$\Rightarrow$(i) Suppose $E_\Sigma \models E_\tau$, and let relation r satisfy Σ; we will show that r satisfies σ. Relation r is, by definition, nonempty and its entries can be w.l.o.g. positive integers. Number its tuples 1,2,... etc., (it could contain a countably infinite number of tuples). Define $A(.):\mathcal{N}\rightarrow\mathcal{N}$, such that, if x is the number of a tuple in r, then A(x) is the entry in tuple x at attribute A, else A(x) is 0, ($\mathcal{N}$ are the nonnegative integers). If f is the CFD $(D_1...D_k\rightarrow D,\ C_1...C_k\rightarrow C)$ in Σ define $F(...):\mathcal{N}^k\rightarrow\mathcal{N}$, such that, if x is the number of a tuple in r, then $F(D_1(x),...,D_k(x)) = D(x)$ and $F(C_1(x),...,C_k(x)) = C(x)$, else F is 0. This is a well defined function since r satisfies f. If i is the IND $D_1...D_k \subseteq C_1...C_k$ in Σ define $I(.):\mathcal{N}\rightarrow\mathcal{N}$, such that, if x is the number of a tuple in r and x' is the number of the first tuple in r where $t_x[D_1...D_k] = t_{x'}[C_1...C_k]$, then $I(x) = x'$, else I(x) is 0. This is also a well defined function since r satisfies i. We have constructed an algebra with domain $\mathcal{N}$ and functions A(.),...,F(...),...,I(.),..., which, as is easy to verify, is a model for E_Σ. Let σ be an IND. By interpreting each symbol in τ as an I(.), we see that when x is a tuple number $\tau(x)$ is another tuple number. Since $E_\Sigma \models E_\tau$, we must have $A_i(\tau) = B_i(x)$ $1 \leq i \leq m$, which means that r satisfies σ. The case of a CFD is similar.

(iii)$\Rightarrow$(ii) Suppose $\mathcal{E}_\Sigma \models \mathcal{E}_\tau$, and let $\mathcal{M}$ be a model of E_Σ; we will show that $\mathcal{M}$ satisfies E_τ. From $\mathcal{M}$ we will construct a model $\mathcal{A}(\mathcal{M})$ for $\mathcal{E}_\Sigma$. The algebra $\mathcal{A}(\mathcal{M})$ will have domain all functions from $\mathcal{M}$ to $\mathcal{M}$, i.e., $\mathcal{M}\rightarrow\mathcal{M}$. In $\mathcal{A}(\mathcal{M})$ the interpretation of α will be the function $a(x)$, which is the interpretation of a(.) in $\mathcal{M}$. The interpretation of i(.) will be the function $\lambda h.h(i(x))$, where $i(x)$ is the interpretation of i(.) in $\mathcal{M}$ (this is a function from $\mathcal{M}\rightarrow\mathcal{M}$ to $\mathcal{M}\rightarrow\mathcal{M}$). The interpretation of f(...) will be the function $\lambda h_1...h_n.f(h_1(x),...,h_n(x))$, where $f(x_1,...,x_n)$ is the interpretation of f(...) in $\mathcal{M}$ (this is a function from $(\mathcal{M}\rightarrow\mathcal{M})^n$ to $\mathcal{M}\rightarrow\mathcal{M}$). It is straightforward to check that equations 3),4) hold in $\mathcal{A}(\mathcal{M})$, because $\mathcal{M}$ is a model for E_Σ. Also equations 5) hold in $\mathcal{A}(\mathcal{M})$: For example, if n=1 the interpretation of f(i(h)) in $\mathcal{A}(\mathcal{M})$ is $f(h(i(x)))$, which is also the interpretation of i(f(h)) (h is any element of $\mathcal{M}\rightarrow\mathcal{M}$). Thus $\mathcal{A}(\mathcal{M})$ is a model for $\mathcal{E}_\Sigma$. Since $\mathcal{E}_\Sigma \models \mathcal{E}_\tau$, $\mathcal{A}(\mathcal{M})$ satisfies $\mathcal{E}_\tau$, and it easily follows that $\mathcal{M}$ satisfies E_τ.

(i)$\Rightarrow$(iii) By induction on the number of steps of a chase proof of σ from Σ. ∎

An alternative proof procedure for IND's and FD's only is given in [22]. We can show that each of the rules in [22] can be simulated using the equational reasoning of Proposition 1 (this provides an alternate proof of the (i)$\Rightarrow$(iii) step for the FD and IND case). Let us illustrate it with an example: From $A\rightarrow B$ and $CD \subseteq AB$ the *pullback rule* of [22] derives $C\rightarrow D$. In equational language $f\alpha = \beta$, $i\alpha = \gamma$, $i\beta = \delta$ and $fix = ifx$ imply $f\gamma = fi\alpha = if\alpha = i\beta = \delta$.

Corollary 1.1: Let Σ be a set of FD's and σ an FD. The implication problem $\Sigma \models \sigma$ is equivalent to a *generator problem for a finitely presented algebra* [18].

Proof: $\mathcal{E}_\Sigma$ is now a finite set of equations with no variables. If $\approx$ is the congruence induced by $\mathcal{E}_\Sigma$ on $\mathcal{T}(M)$ then $\mathcal{T}(M)/\approx$ is a finitely presented algebra. The equational implication in Theorem 1 is known, in this case, as a generator problem for the finitely presented algebra $\mathcal{T}(M)/\approx$. ∎

Using Corollary 1.1, one can observe that the linear time algorithm of [1] for FD inference can be derived in a straightforward way from the algorithm of [18] for the generator problem.

Corollary 1.2: Let Σ be a set of CFD's. The implication problem $\Sigma \models A \equiv B$ is a *uniform word problem for a finitely presented algebra* [18]. ∎

Semigroup Transformation: Let Σ be a set of IND's and b-FD's. Produce the set of symbols M_s from M as follows: for each $f_k(.)$ in M_f add one generator f_k in M_s; for each $i_k(.)$ in M_i add one generator i_k in M_s; for each $a_k(.)$ in M_a add one generator a_k in M_s; add one binary operator + in M_s.

E_S consists of the associative axiom for + and the following word (string) equations (we omit + and parentheses):

1) two equations for each b-FD $\sigma_k = (A_1\rightarrow A,\ B_1\rightarrow B)$: $f_ka_1 = a$ and $f_kb_1 = b$

2) m equations for each IND $\sigma_k = B_1...B_m \subseteq A_1...A_m$: $a_1i_k = b_1$ and ... and $a_mi_k = b_m$.

Corollary 1.3: Let Σ be a set of b-FD's and IND's

$\Sigma \models A \equiv B$ iff $E_S \models a = b$

$\Sigma \models (A_1\rightarrow A,\ B_1\rightarrow B)$ iff $E_S \models wa_1 = a$ and $wb_1 = b$, for some string w in M_s^*

$\Sigma \models B_1...B_m \subseteq A_1...A_m$ iff $E_S \models a_1w = b_1$ and ... and $a_mw = b_m$, for some string w in M_s^*. ∎

Note that the first case is an instance of the *uniform word problem for semigroups*. The other two cases are known as E_S-*unification* problems [14]. By the symmetry in Corollary 1.3, we have

Corollary 1.4 Duality: Let Σ be a set of b-FD's and b-ID's. and σ a b-FD or b-ID. Transform every b-FD $(A\rightarrow B,\ C\rightarrow D)$ into the b-ID $BD \subseteq AC$, and every b-ID $BD \subseteq AC$ into the b-FD $(A\rightarrow B,\ C\rightarrow D)$. If this transformation changes Σ into Σ' and σ into σ', then $\Sigma \models \sigma$ iff $\Sigma' \models \sigma'$. ∎

A similar duality theorem for u-FD's, u-ID's and $\models_{fin}$, follows from the analysis in [16]. In [22] it is shown that implication ($\models$) is undecidable for u-FD's and b-ID's. By Corollary 1.4, it is undecidable for b-FD's and u-ID's. This was also clear from the form of the undecidability reduction used in [22]. One might imagine formal statements, such as an m-FD $(A_1 \to B_1, \ldots, A_m \to B_m)$ as duals for INDs $B_1 \ldots B_m \subseteq A_1 \ldots A_m$. Here the duality would hold in the equational theory, but these statements for $m>2$ have no natural meaning in database theory.

We will now describe a proof procedure for CFD and IND implication, using the special structure of the equational theories from Theorem 1.

The Proof Procedure G: Given a set Σ of CFD's and IND's construct their graphical representation G_Σ defined in Section 2.2. Each attribute name in Σ is associated with one of the nodes of G_Σ.

Rules: Apply some finite sequence of the graph manipulation rules 1,2,3 and 4 of Figure 3 on G_Σ. Rules 1 and 2 introduce new unnamed nodes. Rules 3 and 4 identify two existing nodes; the node resulting from this identification is associated with the union of the two sets of attribute names, that were associated with each of the identified nodes. Note that rules 1,2 w.l.o.g. need be applied at most once to every left-hand side configuration.

Let G be the resulting graph. Associate a unique new name with every unnamed node in G. The black part of G now naturally represents a set of IND's Σ_I and the red part a set of CFD's Σ_F.

We say that $\Sigma \vdash_G \sigma$ when:

σ is $A \equiv B$ and A,B are associated with the same node;

σ is a CFD and σ can be proved from Σ_F by *chase* (for CFD's only, the chase is an efficient decision procedure);

σ is an IND $B_1 \ldots B_m \subseteq A_1 \ldots A_m$ and there are m black directed paths in Σ_I, all with the same sequence of labels, path i starting at A_i and ending at B_i.

Theorem 2: $\Sigma \models \sigma$ iff $\Sigma \vdash_G \sigma$.

Proof Sketch: We outline the proof for σ being $A \equiv B$.

($\Leftarrow$): Rules 3,4 are obviously sound. Rules 1 and 2 are sound in the sense of the *attribute introduction rule* of [22], which we illustrate as rule 5 of Figure 3.

($\Rightarrow$): We assume that we cannot prove σ, and construct a model for $\mathcal{E}_\Sigma$ in which $\alpha \neq \beta$; then by Theorem 1 Σ does not imply σ. If σ is not provable, then there is a (possibly infinite) graph G which represents Σ, is closed under the rules, and in which the names A and B correspond to different nodes. We add one special node $\perp$ to G. The labels of G are symbols corresponding to INDs (i symbols) or CFDs (f symbols) of Σ. The groups of red arcs labeled with an f are also ordered. If a node in $G \cup \{\perp\}$ has no outgoing arc labeled with some i, add one going to $\perp$. If an n-tuple of nodes does not have a group of n arcs leaving it labeled by f (of ARITY n) and ordered 1 to n, add such a group going to $\perp$. The resulting graph represents functions interpreting the operators and generators in $\mathcal{E}_\Sigma$: This is because closure with respect to rules 3 and 4 and the padding of G we performed, guarantees functionality. The node A (B) is the interpretation of α (β). Now closure with respect to rules 1 and 2 guarantees the commutativity conditions of $\mathcal{E}_\Sigma$, and the fact that G represents Σ guarantees equations 3),4) of $\mathcal{E}_\Sigma$. Thus there is a model of $\mathcal{E}_\Sigma$ in which $\alpha \neq \beta$. ∎

4. Computations as Inferences

It has been known, since at least Post's proof of the unsolvability of the word problem for Thue systems [23, 20], that arbitrary computations can be simulated by inferences in semigroups. By our Corollary 1.3, one can therefore simulate computations by inferences of IND's and *unary* FD's, and thus obtain lower bounds on the complexity of the implication problem for IND's and CFD's.

We first describe our machine model: A *deterministic two-stack* machine M is a 5-tuple $(Q, \Pi, q_{start}, h, \delta)$, where Q is a finite set of *states*, Π is a finite set of *symbols* ($Q \cap \Pi = \emptyset$), $q_{start} \in Q$ is the *start state*, $h \in Q$ is the *halt state*, and δ is the *transition function*. Each *move* of M falls into one of the following two types:

1. $\delta(q,\alpha) = (p, \text{POP}_1)$: This means that, if M is in state q and $\alpha \in \Pi$ is the top symbol of STACK_1, then on the next step M goes to state p and *pops* STACK_1.
2. $\delta(q) = (p, \text{PUSH}_1(\beta))$: If M is in state q, then on the next step M goes to state p and *pushes* $\beta \in \Pi$ on STACK_1.

Of course, analogous instructions can manipulate STACK_2.

An *instantaneous description* (ID) of M is a string $x_1 \ldots x_n q y_m \ldots y_1$, where $q \in Q$, $x_i, y_i \in \Pi$: the string $x_1 \ldots x_n$ is the contents of STACK_1 (the top symbol is x_n); the string $y_m \ldots y_1$ is the contents of STACK_2 (the top symbol is y_m). The relation $w_1 \Rightarrow_M w_2$ (ID w_1 *yields* ID w_2 via one step of M) is defined in the standard way [20, 13]; $\Rightarrow_M^*$ is the *reflexive, transitive closure* of $\Rightarrow_M$.

Let us now define a set S of *word equations* (over generators $Q \cup \Pi$) which capture the computation of M:

1. If $\delta(q,\alpha) = (p, \text{POP}_1)$, then $\alpha q = p$ is in S.
   If $\delta(q,\alpha) = (p, \text{POP}_2)$, then $q\alpha = p$ is in S.
2. If $\delta(q) = (p, \text{PUSH}_1(\beta))$, then $q = \beta p$ is in S.
   If $\delta(q) = (p, \text{PUSH}_2(\beta))$, then $q = p\beta$ is in S.

We write $u =_S v$ iff $S \models u = v$. By a standard argument, based on the fact that M is *deterministic* [23, 20], we have

Lemma 1: $q_{start} \Rightarrow_M^* h$ iff $q_{start} =_S h$. ∎

To prove our first lower bound, we transform S into another set of equations T which looks like the sets obtained (as in Corollary 1.3) from IND's and u-FD's. The set of generators is now

$Q \cup \{A_\alpha, B_\alpha, f_\alpha \mid \alpha \in \Pi\} \cup \{i_\alpha \mid \alpha \in \Pi\} \cup \{j_e \mid e \in S\}$.

1. If $q\alpha = p$ is in S, then $qi_\alpha = p$ is in T.

2. If $\alpha q = p$ is in S, then T contains the equations $q = A_\alpha j_e$, $f_\alpha A_\alpha = B_\alpha$, $B_\alpha j_e = p$, where e is $\alpha q = p$.

Lemma 2: $q_{start} =_S h$ iff $q_{start} =_T h$.

Proof Sketch: Observe that if $\alpha q = p$ is in S then $f_\alpha q =_T p$, because $f_\alpha q =_T f_\alpha A_\alpha j_e =_T B_\alpha j_e =_T p$. ∎

Theorem 3: The implication problem for IND's and two u-FD's is undecidable.

Proof Sketch: Given a deterministic two-stack machine M, it is undecidable if $q_{start} \Rightarrow^*_M h$, even if $|\Pi| = 2$ [21, 13]. By Lemmas 1 and 2, $q_{start} \Rightarrow^*_M h$ iff $q_{start} =_T h$, and by Corollary 1.3, $q_{start} =_T h$ iff $\Sigma \models Q_{start} \equiv H$, where Σ is the set of IND's and FD's which gives rise to T. But now observe that Σ only contains FD's of the form $A_\alpha \rightarrow B_\alpha$, $\alpha \in \Pi$, and thus since $|\Pi| = 2$, Σ only contains two unary FD's. ∎

Undecidability of the implication problem for IND's and FD's has already been proved [22, 7]. By way of comparison, these reductions use arbitrarily many b-ID's and u-FD's, while our reduction uses arbitrarily many IND's and only two u-FD's. To prove our second lower bound, we consider computations of a deterministic two-stack machine M where one of the two stacks has *bounded size.* Let us write $w_1 \Rightarrow^k_M w_2$ iff ID w_2 follows from ID w_1 by a computation of M during which STACK$_2$ contains at most k symbols.

Let S be the set of word equations described before: this time we transform S into a set T^k of equations which can be obtained (as in Corollary 1.3) from *acyclic* IND's and u-FD's. The set of generators now is $Q^0 \cup ... \cup Q^k \cup \{A_\alpha, B_\alpha, f_\alpha \mid \alpha \in \Pi\} \cup \{i_{\alpha,m} \mid \alpha \in \Pi, m = 1,...,k\} \cup \{j_{e,m} \mid e \in S, m = 0,...,k\}$, where $Q^m = \{q^m \mid q \in Q\}$, $m = 0,...,k$.

1. If $q\alpha = p$ is in S, then $q^{m+1} i_{\alpha,m+1} = p^m$ is in T^k, $m = 0,..., k-1$.

2. If $\alpha q = p$ is in S, then T^k contains the equations $q^m = A_\alpha j_{e,m}$, $f_\alpha A_\alpha = B_\alpha$, $B_\alpha j_{e,m} = p^m$, $m = 0,...,k$, where e is $\alpha q = p$.

It is not hard to see that T^k can be taken to represent a set Σ^k of acyclic IND's and u-FD's: the relation names would be $R[A_\alpha B_\alpha \mid \alpha \in \Pi]$, $R^m[Q^m]$, $m = 0,...,k$. It is also easy to see the following

Lemma 3: $q_{start} \Rightarrow^k_M h$ iff $q^0_{start} =_{T^k} h^0$, iff $\Sigma^k \models R^0: Q^0_{start} \equiv H^0$. ∎

Theorem 4: There are constants $c_1, c_2 > 0$ such that, given a set Σ of acyclic IND's and CFD's and an IND (CFD) σ, $\Sigma \models \sigma$ can be decided in time c_1^n but not in time $c_2^{\sqrt{n/\log n}}$.

Proof Sketch: Since the IND's in Σ are acyclic, the *chase* gives us a decision procedure, running in exponential time.

To prove the lower bound, let L be any language in DTIME(d^n), d>0. We will show that L is polynomial-time reducible to the implication problem for acyclic IND's and u-FD's.

Let M be a deterministic n-AuxiliaryPushdownAutomaton accepting L [13]. Given string x, we construct a deterministic two-stack machine M_x which first puts x on STACK$_2$ and then simulates M. This simulation is done as follows: if M is in state q, its auxiliary storage contains $\alpha_1 ... \alpha_n \alpha w$ (α is the symbol scanned) and its stack contains $u\beta$ (β is the top symbol), then the ID of M_x is $u\beta\alpha_{1,\beta}...\alpha_{n,\beta} q\alpha w$; it is not hard to see how M_x can simulate a move of M. Thus, M accepts x iff M_x halts and STACK$_2$ always contains at most |x| symbols, i.e. $x \in L$ iff $q_{start} \Rightarrow^{|x|}_{M_x} h$. Note also that $|M_x|$ is $O(|x|)$.

Now let $\Sigma^{|x|}$ be the set of acyclic IND's and u-FD's corresponding to M_x. Using Lemma 3, $x \in L$ iff $\Sigma^{|x|} \models R^0: Q^0_{start} \equiv H^0$. To complete the proof, observe that $\Sigma^{|x|}$ can be computed from x in polynomial time, and that $|\Sigma^{|x|}|$ is $O(|M_x|\,|x| \log|x|)$, i.e. $O(|x|^2 \log|x|)$. ∎

5. An Application to Typed IND's

We show how the tools developed in Section 3 can be applied to the particular case of inferring a CFD from CFD's and *typed* IND's. We first give a formal system for implication, similar in spirit to the formal system of Theorem 2. The semi-decision procedure thus obtained becomes a decision procedure if the CFD's are *acyclic*, generalizing a result of [9] about acyclic unary FD's. On the other hand, by analyzing derivations we can show that the general problem is *undecidable*, even if only unary FD's and *all possible* typed IND's are given.

Let Σ be a given set of CFD's and typed IND's over database scheme $D = \{R_k[U_k]: k = 1,...,q\}$, $U_k \subseteq \mathcal{U}$. We represent attribute $A \in U_k$ by a *node* a_k. A CFD (R_k: $AB \rightarrow C$, R_j: $A'B' \rightarrow C'$) in Σ is represented as shown in Figure 4a by introducing nodes fa_kb_k, $fa_j'b_j'$ (we use a different function symbol f for each given CFD), directed arcs (a_k, fa_kb_k), $(a_j', fa_j'b_j')$ labeled 1 and (b_k, fa_kb_k), $(b_j', fa_j'b_j')$ labeled 2, and undirected arcs $\langle fa_kb_k, c_k\rangle$, $\langle fa_j'b_j', c_j'\rangle$. The undirected arcs are the only modification to our graph notation of Section 2.2. Their purpose is to represent the equalities $fa_kb_k = c_k$, $fa_j'b_j' = c_j'$. A typed IND R_k:$AB \subseteq R_j$:AB in Σ is represented (see Figure 4a) by introducing directed arcs (a_j, a_k), (b_j, b_k) labeled i (we use a different label for each given IND).

Let H_Σ be the mixed graph obtained from Σ as described above. Repeatedly apply rules T(*transitivity*), E_{1-3} (*equality*), I_{1-3} (*introduction*) (see Figure 4b) on H_Σ, in some arbitrary fixed order, until no more rules are applicable. As was the case with Rules 1,2 in Theorem 2, the introduction rules need only be applied once for each left-hand side configuration.

Let $H=(N_H,A_H,E_H)$ be the mixed graph obtained this way (N_H is a set of nodes, A_H is a set of labeled directed arcs on N_H, and E_H is a set of undirected arcs on N_H): notice that each node of H is labeled F(a,...,b), where F is a term over the function symbols and a,...,b are nodes representing attributes. F(a,...,b) is a shorthand for $F[x_1/a,...,x_n/b]$. Moreover, every subterm of F(a,...,b) appears as a node of H. The graph H fully captures implication of CFD's from Σ:

Theorem 5: $\Sigma \models (R_k\text{: } A...B \to C,\ R_j\text{: } A'...B' \to C')$ iff there are nodes $F(a_k,...,b_k)$, $F(a_j',...,b_j')$ of H such that $\langle F(a_k,...,b_k),c_k\rangle \in E_H$, $\langle F(a_j',...,b_j'),c_j\rangle \in E_H$.

Proof: Omitted. See [8]. ∎

If Σ consists of acyclic CFD's and typed IND's it is easy to see that the graph H is *finite*, and in particular its size is *exponential* in the size of S. We thus obtain an exponential time upper bound; whether it can be improved is an open question.

Corollary 5.1: Inferring a CFD from acyclic CFD's and typed IND's is decidable in exponential time. ∎

We can also show that inferring a CFD from (general) CFD's and typed IND's is undecidable, even if the set of IND's is restricted to be PC(D) (pairwise consistency) and all CFD's are unary FD's. Let Σ be a set of u-FD's. Note that if a database d over D satisfies PC(D), then $R_i{:}X \to Y$ holds in relation r_i iff $R_j{:}X \to Y$ holds in r_j, where $R_i[U_i]$, $R_j[U_j]$ both contain attributes X and Y. For this reason we can suppress relation names from FD's.

Let F_Σ be a directed graph with a node A for each attribute A and an arc (X,Y) for each FD $X \to Y$ in Σ. For each attribute A, let T_A be the following (possibly infinite) directed tree:

the set of nodes $P_A \subseteq A\mathcal{U}^*$ is the set of all *paths* in F_Σ which start at A (denoted as sequences of nodes);

the set of arcs is $\{(pX,pXY) \mid p \in \mathcal{U}^*, pX \in P_A, X \to Y \in \Sigma\}$.

Let $P=\cup_{A \in \mathcal{U}} P_A$; define E to be the smallest set of undirected arcs on P which contains $\langle p,p\rangle$ for all $p \in P$ and $\langle XY,Y\rangle$ for all $X \to Y \in \Sigma$, and is closed under the following rules:

1. Propagation: If $\langle pX,qX\rangle \in E$, then $\langle pXY,qXY\rangle \in E$ for all $X \to Y \in \Sigma$.
2. Pseudo-Transitivity: If $\langle p_1,p_2\rangle$, $\langle p_2,p_3\rangle$ are in E, $p_i \in P_{X_i}$, and there is a relation scheme in D which contains X_1,X_2,X_3, then $\langle p_1,p_3\rangle$ is in E.

Example: Figure 5 has an example where $D=\{R_1[AYX], R_2[YXB], R_3[AXB]\}$, and $\Sigma=\{A \to Y, Y \to B, X \to B\}$.

It is not difficult to see that the structure defined above is essentially a succinct representation of the graph H of Theorem 5, and thus it captures implication of u-FD's:

Lemma 4: $PC(D) \cup \Sigma \models X \to Y$ iff $\langle p,Y\rangle \in E$ for some $p \in P_X$. ∎

Example: In the case depicted in Figure 5, $PC(D) \cup \Sigma \models A \to B$.

Theorem 6: The problem of u-FD implication in the presence of pairwise consistency is undecidable.

Proof: Omitted. See [8]. ∎

6. Equational Theories and Finite Implication

We now examine to what extent the tools we developed can handle *finite implication* of database constraints. Ideally, we would like to be able to replace $\models$ by $\models_{fin}$ throughout Theorem 1. However, our proof does not work anymore: To be sure, the same arguments can show that (iii)⇒(ii) and (ii)⇒(i) in the finite case (the constructions given map finite counterexamples to finite counterexamples); on the other hand, the argument for (i)⇒(iii) hinges on the existence of a complete formal system for implication (namely the chase), and such a formal system cannot exist for finite implication [22, 7]. Incidentally, the same *syntactic* nature of the proofs of Theorems 3 and 6 prevents us from proving undecidability of finite implication. The weaker proofs of [22, 7], because of their *semantic* nature, can easily be done for the finite case.

Thus, we have to contend ourselves with partial results about restricted cases. First, by the discussion above one can see that $\models$ can be replaced by $\models_{fin}$ in Theorem 1 if we have a *finitely controllable* class of CFD's and IND's, i.e. a class where $\models_{fin}$ is the same as $\models$. An easy example of such a class is provided by CFD's and *acyclic* IND's, because the chase in this case constructs a *finite counterexample* if the implication does not hold (thus, Theorem 4 also holds for the finite case). Another such class is given in the following

Theorem 7: The implication problem for acyclic FD's under pairwise consistency is finitely controllable.

Proof Sketch: We will only consider unary FD's for simplicity. Let Σ consist of unary FD's; we will show that if $PC(D) \cup \Sigma$ does not imply $X \to Y$, then there is a *finite* pairwise consistent database d which satisfies Σ but violates $X \to Y$. We use the notation of Lemma 4. Define d as follows:

For each attribute $A \in \mathcal{U}$ the *domain* of A, $\mathfrak{I}_A$, consists of all *functions* f: $P_A \to \{0,1\}$ such that, if $\langle p,q\rangle \in E$, p,q in P_A, then $f(p)=f(q)$.

Let $U_i=A_1...A_s$, a tuple $f_{A_1}...f_{A_s}$ ($f_{A_j} \in \mathfrak{I}_{A_j}$) is in relation r_i when: for any p in P_X, q in P_Y with $\langle p,q\rangle \in E$, $f_X(p)=f_Y(q)$ ($X,Y \in U_i$).

Since Σ is acyclic, each P_A is finite, and thus d is finite. It is not hard to reason that d satisfies the FD's in Σ (by the definition of the set F). We also claim that d is pairwise consistent: The key fact is that, if XY...Z is any subset of U_i, then the tuples t[XY...Z] in r_i are exactly the tuples $f_X f_Y ... f_Z$ for which $f_A(p)=f_B(q)$ whenever $\langle p,q\rangle \in E$ $(A,B \in U_i)$. Finally, if $\langle p,Y\rangle$ is not in E for any p in P_X (Lemma 4), then one can verify that d violates $X \to Y$. ∎

Observe that the construction given above provides us with an alternative proof of the "only-if" direction of Lemma 4 (the counterexample obtained is, in general, *uncountable*).

If $\models_{fin}$ is different from $\models$, we might still be able to handle the finite case if there is a complete formal system for finite implication. A class with this property is FD's with *unary* IND's [15]: The formal system consists of standard rules for FD's and IND's [24, 5] and of the following *cycle rules*:

from $A_0 \to A_1$ *and* $A_1 \supseteq A_2$ *and...and* $A_{k-1} \to A_k$ *and* $A_k \supseteq A_0$
derive $A_1 \to A_0$ *and* $A_2 \supseteq A_1$ *and...and* $A_k \to A_{k-1}$ *and* $A_0 \supseteq A_k$ (k odd).

However, it turns out that we cannot replace $\models$ by $\models_{fin}$ in Theorem 1, but have to settle for something weaker. Let Σ be a set of FD's and u-ID's, σ an FD (u-ID); $\Sigma \models_{fin} \sigma$ can be characterized as follows ($\bigvee$ stands for an infinitary disjunction of equations).

Theorem 8 (σ is an FD): In each of the following two cases, (i),(ii),(iii) are equivalent:

FD Case:

i) $\Sigma \models_{fin} A_1 ... A_n \to A$.

ii) $E_\Sigma \models_{fin} \bigvee_{\tau \in \mathcal{T}_f} \tau[x_1/a_1x,...,x_n/a_nx] = ax$.

iii) $\mathcal{E}_\Sigma \models_{fin} \bigvee_{\tau \in \mathcal{T}_f} \tau[x_1/\alpha_1,...,x_n/\alpha_n] = \alpha$.

u-ID Case:

i) $\Sigma \models_{fin} B \subseteq A$.

ii) $E_\Sigma \models_{fin} \bigvee_{\tau \in \mathcal{T}_i} a\tau = bx$.

iii) $\mathcal{E}_\Sigma \models_{fin} \bigvee_{\tau \in \mathcal{T}_i} \tau[x/\alpha] = \beta$.

Proof Sketch: The implications (iii)⇒(ii), (ii)⇒(i) can be proved by the same argument as in Theorem 1. We show (i)⇒(iii) by induction on the length m of a proof of σ from Σ. The basis case (m=0) is obvious. For the induction step, we only check the cycle rule corresponding to k=1 (the argument generalizes to arbitrary k). We write $\tau(\alpha)$ as a shorthand for $\tau[x/\alpha]$.

By induction on m, if $\mathcal{A}$ is a finite model of $\mathcal{E}_\Sigma$, then $\mathcal{A}$ satisfies $\rho(\alpha_0)=\alpha_1$, $\tau(\alpha_1)=\alpha_0$, for some $\rho \in \mathcal{T}_f$, $\tau \in \mathcal{T}_i$. We are ready to apply the m+1 step of the derivation, which will be the cycle rule step for k=1. Consider now the set $K=\{\rho^k(\alpha_1) : k \geq 0\}$ (ρ^k is ρ composed with itself k times): if $\alpha_0 \in K$, then $\mathcal{A}$ satisfies $\rho^k(\alpha_1)=\alpha_0$ for some k, and $\rho^k \in \mathcal{T}_f$. This term ρ^k gives us the proof for the m+1st step. If such a term did not exist, then let r be the least integer such that $\rho^r(\alpha_1)=\rho^s(\alpha_1)$, for some s>r>1 (K is finite since $\mathcal{A}$ is finite): by commutativity, $\tau(\rho^r(\alpha_1))=\rho^r(\tau(\alpha_1))=\rho^r(\alpha_0)=\rho^{r-1}(\rho(\alpha_0))=$ $=\rho^{r-1}(\alpha_1)$, and similarly $\tau(\rho^s(\alpha_1))=\rho^{s-1}(\alpha_1)$. But this means $\rho^{r-1}(\alpha_1)=\rho^{s-1}(\alpha_1)$, which contradicts the choice of r. For r=1 we get a similar contradiction. ∎

Finally, observe that if the FD's are also unary, we have (by analogy to Corollary 1.3) the *finite* E_S *unification* problem.

7. Conclusions and Open Problems

We have demonstrated a close relationship between implication of equations and implication of database constraints. We used this relationship to derive better bounds for the implication of FD's and IND's, which are the most common database dependencies.

An interesting practical question is how well conventional theorem proving systems perform on database dependency questions [17, 12]. Some theoretical questions also remain unresolved. For the common case of FD's and typed IND's, there is a considerable gap between the exponential upper bounds (for acyclic IND's and FD's, and for typed IND's and acyclic FD's) and the NP-hardness lower bound of [9]. The undecidability of finite implication for FD's in the presence of pairwise consistency is open, as well as the finite controllability of acyclic FD's and (general) typed IND's.

References

1. Beeri, C. and Bernstein, P.A. "Computational Problems Related to the Design of Normal Form Relational Schemas". *ACM Transactions on Database Systems 4*, 1 (March 1979), 30-59..

2. Beeri, C. and Vardi, M.Y. "Formal Systems for Tuple and Equality Generating Dependencies". *SIAM Journal of Computing 13*, 1 (February 1984), 76-98..

3. Beeri, C. and Vardi, M.Y. "A Proof Procedure for Data Dependencies". *Journal of the Association for Computing Machinery 31*, 4 (October 1984), 718-741..

4. Birkhoff, G. "On the Structure of Abstract Algebras". *Proceedings of the Cambridge Philosophical Society 31*, (1935).

5. Casanova, M.A., Fagin, R. and Papadimitriou, C.H. "Inclusion Dependencies and Their Interaction with Functional Dependencies". *Journal of Computer and System Sciences 28*, 1 (February 1984), 29-59..

6. Casanova, V. and Vidal, V.M.P. "Towards a Sound View Integration Methodology". *Proceedings of the 2nd ACM Symposium on Principles of Database Systems* (1983).

7. Chandra, A.K. and Vardi, M.Y. The Implication Problem for Functional and Inclusion Dependencies is Undecidable. IBM Tech. Rep. RC 9980, , , 1983.

8. Cosmadakis, S.S. *Equational Theories and Database Constraints.* Ph.D. Th., Massachusetts Institute of Technology, 1985.

9. Cosmadakis, S.S. and Kanellakis, P.C. "Functional and Inclusion Dependencies: A Graph Theoretic Approach". *Proceedings of the 3rd ACM Symposium on Principles of Database Systems* (April 1984), 24-37.

10. Downey, P.J., Sethi, R. and Tarjan. R.E. "Variations on the Common Subexpression Problem". *Journal of the Association for Computing Machinery 27*, 4 (October 1980), 758-771..

11. Fagin, R. "Horn Clauses and Database Dependencies". *Journal of the ACM 29*, 4 (October 1982), 952-985..

12. Forgaard, R. and Guttag, J.V. "REVE: A Term Rewriting System Generator with Failure Resistant Knuth-Bendix". *Proceedings of an NSF Workshop on the Rewrite Rule Laboratory* (April 1984), 5-31.

13. Hopcroft, J.E. and Ullman, J.D.. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publishing Company, Inc.,, 1979.

14. Huet, G. and Oppen, D. Equations and Rewrite Rules: a Survey. In *Formal Languages: Perspectives and Open Problems*, , Eds., Academic Press. , 1980.

15. Johnson, D.S. and Klug, A. "Testing Containment of Conjunctive Queries Under Functional and Inclusion Dependencies". *Journal of Computer and System Sciences 28*, 1 (February 1984), 167-189..

16. Kanellakis, P.C., Cosmadakis, S.S. and Vardi, M.Y. "Unary Inclusion Dependencies Have Polynomial Time Inference Problems". *Proceedings of the 15th Annual ACM Symposium on Theory of Computing* (1983).

17. Knuth, D.E. and Bendix, P.B. . Volume : Simple Word Problems in Universal Algebras. In *Computational Problems in Abstract Algebra*, , Eds., Pergamon, Oxford, 1970, ch..

18. Kozen, D. "Complexity of Finitely Presented Algebras". *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing, ACM SIGACT* (May 1977)..

19. Laver, K., Mendelzon, A.O. and Graham, M.H. "Functional Dependencies on Cyclic Database Schemes". *Proceedings ACM SIGMOD* (1983).

20. Lewis, H.R. and Papadimitriou, C.H.. *Elements of the Theory of Computation.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981. .

21. Minsky, M.L. "Recursive Unsolvability of Post's Problem of "Tag" and Other Topics in the Theory of Turing Machines". *Annals of Mathematics 74*, 3 (1961)..

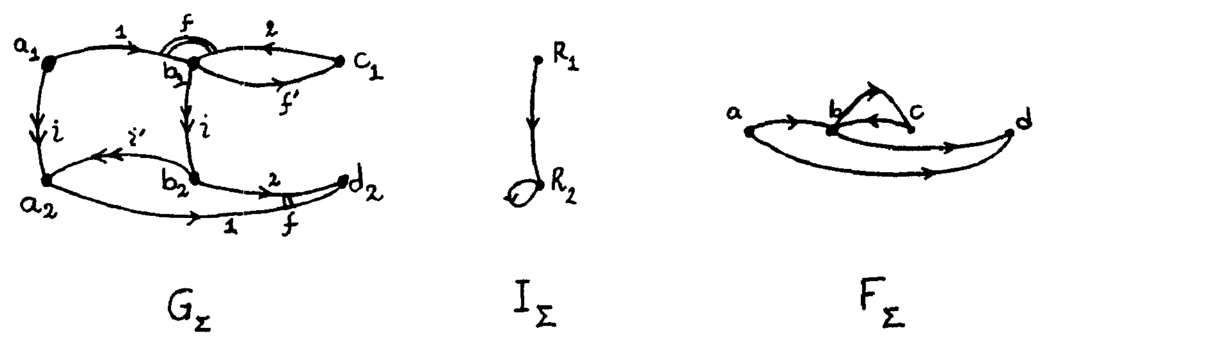
22. Mitchell, J.C. "The Implication Problem for Functional and Inclusion Dependencies". *Information and Control 56*, 3 (March 1983), 154-173..

23. E.L. Post. "Recursive Unsolvability of a Problem of Thue". *Journal of Symbolic Logic 13*, (1947).

24. Sciore, E. "Inclusion Dependencies and the Universal Instance". *Proceedings of the 2nd ACM Symposium on Principles of Database Systems* (1983).

25. Ullman, J.D.. *Principles of Database Systems.* Computer Science Press, Inc., , 1983.

26. Yannakakis, M. and Papadimitriou C.H. "Algebraic Dependencies". *J. Comput. Systems Sci. 21*, 1 (August 1982), 2-41..



$\mathfrak{D} = \{ R_1[ABC],\ R_2[ABD] \}$

$\Sigma = \{ (R_1 : AC \to B,\ R_2 : AB \to D),$
$R_1 : B \to C,$
$R_1 : AB \subseteq R_2 : AB,$
$R_2 : A \subseteq R_2 : B \}$

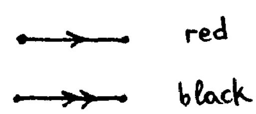


Figure 1

$\underline{\Sigma}$: $(A \to B,\ C \to D)$
$CD \subseteq AB$
$A'B' \to C'$

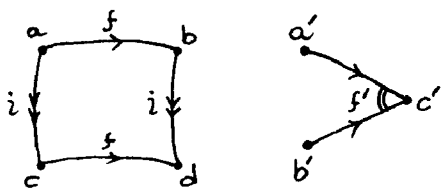
$\underline{M}$:

M_f: $f(\cdot)$ $f'(\cdot,\cdot)$

M_i: $i(\cdot)$

M_a: $a(\cdot)$ $b(\cdot)$ $c(\cdot)$ $d(\cdot)$
$a'(\cdot)$ $b'(\cdot)$ $c'(\cdot)$

M_α: α β γ δ
α' β' γ'

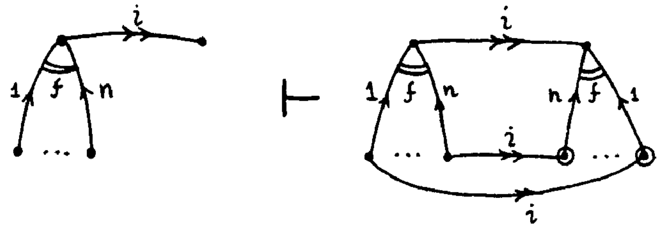


$\underline{E_\Sigma}$:
$$fax = bx$$
$$fcx = dx$$
$$aix = cx$$
$$bix = dx$$
$$f'a'xb'x = c'x$$

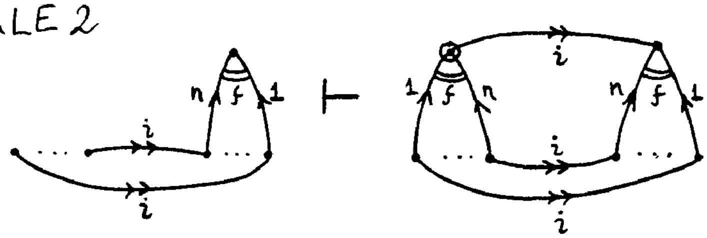
$\underline{\mathcal{E}_\Sigma}$:
$$f\alpha = \beta$$
$$f\gamma = \delta$$
$$i\alpha = \gamma$$
$$i\beta = \delta$$
$$f\alpha'\beta' = \gamma'$$
$$fix = ifx$$
$$f'ix_1ix_2 = if'x_1x_2$$

Figure 2

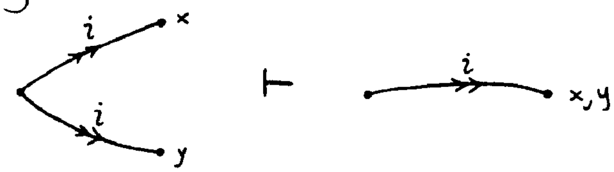
RULE 1



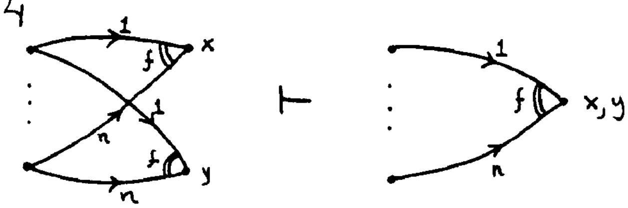
RULE 2



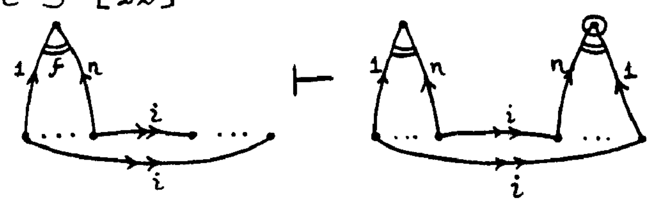
RULE 3



RULE 4



RULE 5 [22]



⊙: new node

Figure 3

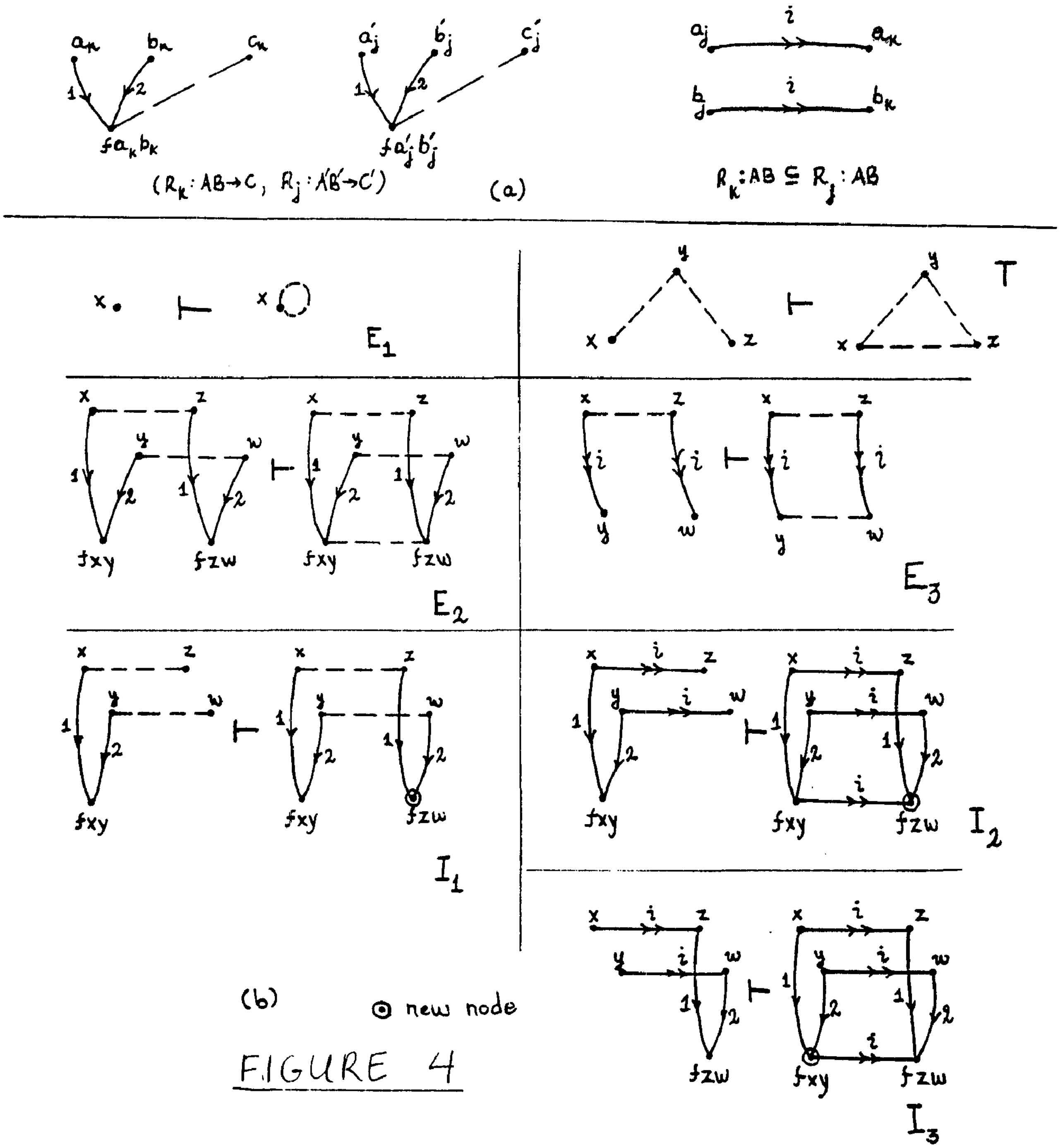


FIGURE 4

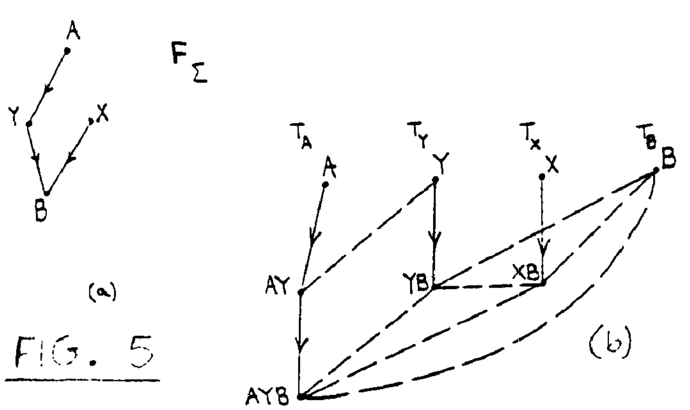


FIG. 5