Partition Semantics for Relations

Stavros S. Cosmadakis
MIT

Paris C. Kanellakis[1]
MIT

Nicolas Spyratos
Universite de Paris Sud

Abstract

Set-theoretic partitions were first used in [15] to provide models for relation schemes, relations and dependencies. This new point of view of the relational model demonstrates that there is a natural extension of functional dependencies (FD's), which is based on the duality between product and sum of partitions. We show that these *partition dependencies* (PD's) have the power to express both functional determination and transitive closure of undirected graphs. The inference problem of PD's is shown to be the uniform word problem in a lattice. We provide a polynomial time algorithm for this natural generalization of the FD inference problem. We show how partition semantics justify a number of variants of the weak instance assumption and investigate the expressive power of PD's. We also provide a polynomial time test for consistency of a set of relations with a set of PD's.

1. Introduction

It is customary in database theory to consider the database scheme and logical constraints as sentences from predicate calculus and the database itself as an interpretation. This relational interpretation either satisfies or falsifies the scheme and constraints. An interesting body of theory has developed around this approach, particularly with respect to special types of logical sentences about relations called *dependencies* (see [16, 12] for surveys of the area). In this paper we will take a slightly different view, in which database scheme, constraints and the database are strings of uninterpreted symbols. Interpretations are provided using *partitions*, that is, families of nonempty, disjoint sets whose union is a population of objects. This new approach allows us to better understand and provide nontrivial generalizations to such basic concepts as functional dependencies and weak instances [8, 18].

Partition semantics, first proposed in [15], reveal the algebraic nature of the most common database dependency, the *functional dependency* (FD). Using partition semantics we demonstrate that the inference problem for FD's is the problem of infering an equation between algebraic expressions from other such equations. In more mathematical terms, it is a restricted form of *the uniform word problem in a lattice* [3, 11]. This approach, of viewing dependency inference as a word problem in an equational theory, is similar in spirit to algebraic dependency inference [19], although much closer in its details to standard equational theories [10].

[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.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

©1985 ACM 0-89791-153-9/85/003/0261 $00.75

Let $+,\cdot$ be two operators satisfying the lattice axioms [3] (see also Proposition 1) and let the attributes of the database scheme be considered as constants. Partition semantics allow us, without any loss of generality, to treat an FD $AB \rightarrow CD$ as the equation $A \cdot B = A \cdot B \cdot C \cdot D$. Interestingly enough, partition semantics allow us to complete the analogy. An equation $A = B + C$, where $+$ is the natural dual of $\cdot$ in a lattice of partitions, is a dependency expressing a transitive closure condition. In general a *partition dependency* (PD) is an equation $e = e'$, where e, e' are algebraic expressions using $+,\cdot$ and the attributes. Since transitive closure cannot be expressed using relational algebra [1], and since, as we show, PD's cannot express multivalued dependencies, the expressive power of PD's is orthogonal to that of other generalizations of FD's [19, 5, 2, 14].

We show that the *inference problem* for PD's can be solved in *polynomial time*. For this we solve efficiently the uniform word problem in a lattice. We also use this algorithm to efficiently reduce the problem of *testing consistency of a set of relations with a set of* PD's, to that of testing consistency with a set of FD's [8].

An intuitive justification of partition semantics is that the world consists of a set of objects, for which the *tuples* in the database provide *names*. Two objects in a set, whose name is a particular tuple, cannot be distinguished on the basis of this name. The only names with meaning are those of nonempty sets. This distinction between nonempty sets (blocks of a partition) and the empty set $\emptyset$ provides us with a notion of truth.

We show that this formal approach leads to a number of variants of the *weak instance assumption*. These variants depend on whether we assume an open or closed world interpretation [13]. If we assume a closed world interpretation, which we formalize using the Complete Atomic Data Assumption (CAD) of [15], then testing consistency of a database with a set of FD's becomes NP-complete. If we assume an open world interpretation, testing consistency of a database with a set of FD's can be done using the efficient weak satisfaction test of [8]. We generalize this test to a set of PD's.

We assume that the reader is familiar with the standard relational terminology in [16, 12], as well as, with the concept of a weak instance for a database and a set of FD's [8, 18]. A useful survey of equational theories can be found in [10]. Computational aspects of these theories are explored in [11].

We define partition semantics in Section 2. The formal definitions are in Section 2.1. Partition dependencies and motivating examples, which illustrate functional determination and transitive closure, are in Section 2.2. This section also contains other possible restrictions on interpretations, such as the Complete Atomic Data Assumption (CAD). The expressive power of partitions is the subject of Section 3. Section 3.1 demonstrates the connection with weak instances and contains the lower bounds for testing consistency of a database and FD's when we assume CAD. Section 3.2 justifies why PD's can express (without loss of generality) FD's and symmetric transitive closure in a relation. We also show that PD's cannot express multivalued dependencies. A solution to the PD inference problem is contained in Section 4. We first present a new algebraic view of classical FD inference and then the efficient inference algorithm for PD's. The proof of this algorithm uses new techniques for database theory, which derive from algebra and the theory of lattices [3, 11]. In Section 5 we provide a polynomial time test for consistency of a set of relations with a set of PD's; this test combines the test of [8] with the efficient inference algorithm for PD's.

2. Partition Semantics

In this Section we summarize the basic concepts of partition semantics, first proposed in [15].

2.1. Relations and Partitions

Let $\mathfrak{U}$ be a finite set of *attributes* $\{A_1,...,A_k\}$ and $\mathfrak{D}$ a countably infinite set of *symbols* $\{a,a_1,...,b,b_1,...\}$, such that $\mathfrak{U}\cap\mathfrak{D}=\emptyset$. A *relation scheme* is an object R[U], where R is the *name* of the relation scheme and $U\subseteq\mathfrak{U}$. A *tuple* t over U is a function from U to $\mathfrak{D}$. If $t[A_i]=a_i$, $1\le i\le n=|U|$, then we represent tuple t as $a_1a_2...a_n$, and the restriction of t on a subset X of U as t[X]. A *relation* r over U is a set of tuples over U. We assume that, in general, there are both finite and infinite relations. A *database scheme* D is a finite set of relation schemes $\{R_1[U_1],...,R_m[U_m]\}$ and a *database* $d=\{r_1,...,r_m\}$ associates each relation scheme $R_i[U_i]$ with a relation r_i over U_i.

A database is represented, in the natural fashion, by a set of tables with the relation schemes as headers, the tuples as rows, and each column headed by an attribute. We use d[A] to denote the set of symbols appearing in database d under *all* columns headed by attribute A.

We view all the above relational database objects as syntactic objects, which we interpret as follows.

A *partition interpretation* $\mathfrak{I}$ is a set of triples $\{ (p(A), \pi_A, f_A) \mid A\in\mathfrak{U} \}$, where for each attribute A:

1. p(A) is a nonempty set, the *population* of A,
2. π_A is a partition of p(A), the *atomic partition* of A,
3. f_A is a function from $\mathfrak{D}$ to $\pi_A\cup\{\emptyset\}$, such that,
   $\pi_A=\{ f_A(x) \mid x\in\mathfrak{D}, f_A(x)\neq\emptyset \}$ *and*
   if $x\neq y$ *then* $f_A(x)\cap f_A(y)=\emptyset$.

Note that π_A is a partition, i.e., a family of nonempty, disjoint subsets of p(A) whose union is p(A). Also f_A maps a unique symbol from $\mathfrak{D}$ to each distinct member of π_A, and maps to $\emptyset$ otherwise.

Let us now define two natural operations on partitions. Given a partition π of set (or population) p and a partition π' of set (or population) p', then their *product* $\cdot$ is :

$$\pi\cdot\pi'=\{ x \mid x=y\cap z\neq\emptyset, y\in\pi, z\in\pi' \}$$

their *sum* $+$ is :

$$\pi+\pi'=\{ x \mid \text{two elements } \mu,\nu \text{ in } p\cup p' \text{ are in set } x \text{ if and only if there is a chain of sets } x_1,x_2,...x_q \text{ from } \pi \text{ or } \pi', \text{ such that } \mu\in x_1, \nu\in x_q \text{ and } x_i\cap x_{i+1}\neq\emptyset, \text{ for all } 1\le i<q \}$$

Note that $\pi\cdot\pi'$ is a partition of the population $p\cap p'$ and $\pi+\pi'$ is a partition of the population $p\cup p'$. It is standard terminology to refer to members of partitions as *blocks*. Also if $p=p'$ then $\cdot$ produces the *coarsest common refinement* of π and π' and $+$ produces their *finest common generalization*. It is also easy to verify that the product $\cdot$ is *associative, commutative,* and *idempotent.*

We can now use the attributes in $\mathfrak{U}$ and the *uninterpreted operator symbols* $\cdot,+$ to build algebraic expressions. We call such expressions *partition expressions*; every attribute is a partition expression and if e,e' are partition expressions then so are $(e\cdot e')$ and $(e+e')$.

Given a partition interpretation $\mathfrak{I}$ we define the semantics (*meaning*) of a partition expression using structural induction and interpreting the $\cdot,+$ symbols as partition product and sum respectively.

Let partition expressions e,e' in partition interpretation $\mathfrak{I}$, have meanings the partitions π,π' of populations p,p' respectively, then:

1. The meaning of attribute A is partition π_A of population p(A)
2. The meaning of $(e\cdot e')$ is partition $\pi\cdot\pi'$ of population $p\cap p'$.
3. The meaning of $(e+e')$ is partition $\pi+\pi'$ of population $p\cup p'$.

Given a partition interpretation $\mathfrak{I}$ we define the meaning of a relation scheme R[U], $U=\{ A_1,A_2,...,A_n \}$, to be the same as the meaning of the partition expression $A_1\cdot A_2\cdot...\cdot A_n$. This meaning is well defined because of the associativity, commutativity and idempotence of $\cdot$. For

example, the meaning of R[ABC] in $\mathfrak{I}$ is the (*composite*) partition $\pi_A \cdot \pi_B \cdot \pi_C$ of population $p(A) \cap p(B) \cap p(C)$.

A partition interpretation $\mathfrak{I}$ also allows us to define the meaning of a tuple t in relation r whose scheme is R[U]. The meaning of symbol t[A] (the symbol of t in the A column) is $f_A(t[A])$. This meaning is either $\emptyset$ or a block of the atomic partition π_A. The meaning of tuple t is $\cap_{A \in U} f_A(t[A])$. This meaning is either $\emptyset$ or a block of the composite partition, which is the semantics of R[U]. For example, let tuple abc be in a relation of database d whose scheme is R[ABC]. Also let us have in $\mathfrak{I}$ $f_A(a)=a$, $f_B(b)=b$, $f_C(c)=c$, where a, b, c are blocks from partitions π_A, π_B, π_C respectively. Then the meaning of t[A] in $\mathfrak{I}$ is a and the meaning of t is $a \cap b \cap c$. In a different partition interpretation $\mathfrak{I}'$ which is identical with $\mathfrak{I}$ with the exception of $f_A(a)=\emptyset$, the meaning of t[A] is $\emptyset$ and that of t is also $\emptyset$.

Note that when there is no ambiguity we will use a for $f_A(a)$ and abc for $a \cap b \cap c$.

We may now view both database scheme and database as syntactic objects with partition interpretations.

Definition 1: Let d be a database and $\mathfrak{I}$ a partition interpretation. $\mathfrak{I}$ is a model for d, or $\mathfrak{I}$ satisfies d, or $\mathfrak{I} \models d$, when:

$\forall r \in d, \forall t \in r \cap_{A \in U} f_A(t[A]) \neq \emptyset$,
where R[U] is the relation scheme of relation r.

2.2. Dependencies

In this section we provide means of expressing restrictions on partition interpretations, other than the restriction of being models for a database d, which was presented in Definition 1.

Definition 2: A *partition dependency* (PD) is an equation $e=e'$, where e,e′ are partition expressions. A partition interpretation $\mathfrak{I}$ is a model for $e=e'$, or $\mathfrak{I}$ satisfies $e=e'$, or $\mathfrak{I} \models e=e'$, if:

$\pi = \pi'$ and $p=p'$,
where partitions π, π' of populations p,p′ are the respective meanings of e,e′ in $\mathfrak{I}$.

Proposition 1: Let x,y,z be partition expressions. The following identities are true in all partition interpretations.

associativity:
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$, $(x+y)+z = x+(y+z)$
commutativity: $x \cdot y = y \cdot x$, $x+y = y+x$
idempotence: $x \cdot x = x$, $x+x=x$
absorption: $x+(x \cdot y)=x$, $x \cdot (x+y)=x$

∎

The above properties show that, given a partition interpretation $\mathfrak{I}$ the algebraic structure produced by closing its atomic partitions under product and sum is a *lattice* [3]. We denote this lattice as $L(\mathfrak{I})$.

Let U be a set of attributes $\{ A_1,...,A_n \}$. With a slight abuse of notation, in the context of a PD we use U for the partition expression $A_1 \cdot ... \cdot A_n$. This is consistent with our definition of the meaning of a relation scheme. We can now define a special important class of partition dependencies.

Let X,Y be sets of attributes. A *functional partition dependency* (FPD) is a partition dependency of the form $X = X \cdot Y$. It is easy to show that:

Proposition 2: If $\mathfrak{I} \models X = X \cdot Y$ and if partitions π, π' of populations p,p′ are the meanings of X,Y in $\mathfrak{I}$ then:

1. $\forall x \in \pi, \exists y \in \pi' . x \subseteq y$

2. $p \subseteq p'$. ∎

Because of the duality of $\cdot, +$ in the lattice of partitions (see Proposition 1) the FPD $X = X \cdot Y$ is equivalent to the PD $Y = X + Y$. Proposition 2 is illustrated in Figure 1.

Example 1: Assume we have two attributes EMPLOYEE-NUMBER and MANAGER-NUMBER and we wish to express the fact that: *each employee can be associated with only one manager.* This is analogous to the familiar functional dependency for relations

[16, 12] EMPLOYEE-NUMBER → MANAGER-NUMBER. We express this constraint as the FPD EMPLOYEE-NUMBER = EMPLOYEE-NUMBER • MANAGER-NUMBER The rigorous justification of the correspondence of the functional dependency X→Y to the FPD X=X•Y is contained in Section 3.2. Intuitively the above FPD guarantees that two individuals of the population of EMPLOYEE-NUMBER in the same block of $\pi_{\text{EMPLOYEE-NUMBER}}$ (e.g., all unlucky individuals whose employee number is 13) are also in the same block of $\pi_{\text{MANAGER-NUMBER}}$ (e.g., the set of individuals whose manager has manager number 7). A fine point is that in each model of this FPD p(EMPLOYEE-NUMBER)⊆p(MANAGER-NUMBER). Therefore, by Proposition 2, MANAGER-NUMBER is uniquely determined for any possible EMPLOYEE-NUMBER. Note that the semantics in this case allows models in which a manager manages individuals who do not have employee numbers, although every individual with an employee number must be managed by a manager. An equivalent way of expressing this functional determination is using the PD EMPLOYEE-NUMBER + MANAGER-NUMBER = MANAGER-NUMBER.

Example 2: Another class of useful facts are ISA relationships such as: *every car is a vehicle.* A surprising fact about partition interpretations is that there seems to be no distinction between expressing functional determination and ISA relationships. This is because ISA relationships implicitly define a function from subset to superset. In this example the natural FPD constraint is: CAR = CAR • VEHICLE. In any model of this FPD p(CAR)⊆p(VEHICLE) and a CAR block functionally determines a VEHICLE block.

Example 3: Let p be a population of cars and p′ a population of bicycles, moreover let p∩p′=∅. We wish to express that: *every vehicle is either a car or a bicycle.* Let CAR-RGS be interpreted as a partition π of p, and BICYCLE-RGS as a partition π' of p′ (the database cannot distinguish between cars with the same car registration or between bicycles with the same bicycle registration). Since the two populations are distinct $\pi+\pi'=\pi\cup\pi'$, that is + produces the union of two families of blocks. In this case one may use the PD: VEHICLE-RGS = CAR-RGS + BICYCLE-RGS.

Example 4: Consider a database d with only one relation r representing an undirected graph. This relation has three attributes: HEAD, TAIL and COMPONENT. For every edge {a,b} in the graph we have two tuples abc and bac in the relation, where c is a number which could vary with a and b. These are the only tuples in r. Note that by the way r was constructed from the graph we may, without loss of generality, restrict our attention to models of d with p(HEAD)=p(TAIL). We would like to express that: *component is the connected component in which the arc (head, tail) belongs.* We can do this by restricting partition interpretations to: $\mathfrak{I}\models$(d and COMPONENT = HEAD + TAIL). This last example illustrates how by regarding both d and a PD as syntactic objects we can express the meaning of symmetric transitive closure.

Example 5: Suppose we wish to keep a relation of cars, which are complex objects with characteristics registration number and factory serial number. We can express it as a composite partition using the semantics of relation schemes: CAR = RGS-NUMBER • SERIAL-NUMBER.

There are some other natural restrictions we might impose on our models. Assume $\mathfrak{I}$ satisfies database d and a set of PD's, we say that:

1. $\mathfrak{I}$ satisfies the *Complete Atomic Data Assumption* (CAD) if:
   $\forall A\in\mathfrak{U}, \forall a\in\mathfrak{D}$.
   $a\in d[A]$ if and only if $f_A(a)\neq\emptyset$.

2. $\mathfrak{I}$ satisfies the *Common Atomic Populations Assumption* (CAP) if:
   $\forall A,B\in\mathfrak{U}$. p(A)=p(B).

3. In $\mathfrak{I}$ A,B have *disjoint populations* if $p(A) \cap p(B) = \emptyset$

The CAD assumption produces a closed world [13]. It says that the only true atomic facts about attribute A are the ones in database d in the columns headed by A. As in Definition 1 true is equivalent to having nonempty meaning. An interesting fact is that, in contrast to CAD, which turns out to be quite restrictive (from a complexity point of view), CAP has no significant effect on our analysis. Disjoint populations allow us to use + in order to express ∪ as in Example 3 above.

Finally we would like to mention that the lattice generated by the atomic partitions of $\mathfrak{I}$ is not necessarily a *distributive lattice* [3]. We illustrate this in Figure 2 together with many of the preceding definitions.

3. Expressive Power of Partitions

We are interested in the following question: Given a database d and a finite set E of PD's, is there a partition interpretation $\mathfrak{I}$ such that $\mathfrak{I} \models d$ and $\mathfrak{I} \models E$ (i.e., $\mathfrak{I} \models e$ for all e in E)? We first examine this question in the case where E consists of FPD's; we discover a close connection between existence of a partition interpretation and existence of a weak instance [8, 18] for d satisfying E_F, the set of FD's corresponding to the FPD's in E. The ideas introduced motivate a definition of satisfaction of a PD by a database, which enables us to study the expressive power of PD's viz. EID's.

3.1. Partitions and Weak Instances

Given a database d and a set E of FPD's, let $E_F = \{X \rightarrow Y \mid X = X \cdot Y$ is in E$\}$. We ask the following two questions:

a. Is there a partition interpretation $\mathfrak{I}$ such that $\mathfrak{I} \models d,E$?

b. Is there a partition interpretation $\mathfrak{I}$ such that $\mathfrak{I} \models d,E$ and $\mathfrak{I}$ satisfies CAD and CAP?

Recall that a relation w is a *weak instance* for a database d iff every tuple of relation R[U] of d appears in the projection of w on U [8, 18]. The answer to both questions above is given in the following

Theorem 1:

a. There is an $\mathfrak{I}$ such that $\mathfrak{I} \models d,E$ iff there is a weak instance for d satisfying E_F.

b. There is an $\mathfrak{I}$ such that $\mathfrak{I} \models d,E$ and $\mathfrak{I} \models CAD,CAP$ iff there is a weak instance w for d which satisfies E_F and $w[A] = d[A]$, for A in $\mathfrak{U}$.

Proof:

a. ($\Rightarrow$): Say $\mathfrak{I} \models d,E$ and define a relation w over $\mathfrak{U}$ as follows: for each i in $\cup_{A \in \mathfrak{U}} p(A)$, w contains a tuple t_i, where for each attribute A, $t_i[A]$ is a if $i \in a \in \pi_A$, and i otherwise. It is easy to see that w is a weak instance for d: if abc is a tuple appearing in relation R[ABC] of d, then $a \cap b \cap c \neq \emptyset$, so there is an i such that $i \in a \in \pi_A$, $i \in b \in \pi_B$, $i \in c \in \pi_C$, and therefore $t_i[ABC] = abc$. To see that $w \models E_F$, let $X \rightarrow Y$ be an FD in E_F, and suppose there are tuples t_i, t_j in w such that $t_i[X] = t_j[X]$. Then for each A in X $t_i[A] = t_j[A]$, which means $i,j \in a$ for some $a \in \pi_A$. But $\mathfrak{I} \models X = X \cdot Y$, and thus for each B in Y $i,j \in b$ for some $b \in \pi_B$, i.e. $t_i[B] = t_j[B]$. Thus, $w \models X \rightarrow Y$.

($\Leftarrow$): Let w be a weak instance for d, $w \models E_F$. Define $\mathfrak{I}$ as follows: for each A in $\mathfrak{U}$, $p(A) = \{i_t \mid t$ is a tuple of w$\}$, and for each a in $\mathfrak{D}$, $f_A(a) = \{i_t \mid t[A] = a\}$. This f_A induces partition π_A. Note that $\mathfrak{I}$ satisfies CAP. To see that $\mathfrak{I} \models d$, let abc be a tuple in relation R[ABC] of d: there is a tuple t in w with $t[ABC] = abc$, and thus $i_t \in f_A(a) \cap f_B(b) \cap f_C(c) = a \cap b \cap c$, i.e. $a \cap b \cap c \neq \emptyset$. To see that $\mathfrak{I} \models E$, let $X = X \cdot Y$ be an FPD in E, and let i_t, i_s be elements such that $i_t, i_s \in a \in \pi_A$ for each A in X. This means $t[X] = s[X]$, and since $w \models E_F$, $t[B] = s[B]$ for each B in Y. But then $i_t, i_s \in b \in \pi_B$ for each B in Y, and thus $\mathfrak{I} \models X = X \cdot Y$.

b. Same as (a): all we need to observe is that, if $\mathfrak{I}$ satisfies

CAD and CAP, the relation w defined from $\mathfrak{I}$ has $w[A]=d[A]$ for each A in $\mathfrak{U}$. Conversely, if $w[A]=d[A]$ for each A in $\mathfrak{U}$, then the partition interpretation $\mathfrak{I}$ defined from w satisfies CAD (it always satisfies CAP). ∎

Thus, given d,E we can test in polynomial time whether there is an $\mathfrak{I}$ such that $\mathfrak{I}\models d,E$, since we can test whether d has a weak instance satisfying E_F [8]. However, introducing CAD complicates things:

Theorem 2: Given d,E it is NP-complete to test whether there is an $\mathfrak{I}$ such that $\mathfrak{I}\models d,E$ and $\mathfrak{I}\models CAD,CAP$.

Proof (Sketch): Membership in NP follows from Theorem 1 (b): just guess an appropriate weak instance w (w need only contain one tuple for each tuple of d).

NP-hardness is shown by a reduction from NOT-ALL-EQUAL-3SAT [6]: given a Boolean formula φ over variables $x_1,...,x_n$ with clauses $c_1,...,c_m$, test whether there is a truth assignment under which each clause c_i has one true and one false literal.

From φ construct d,E as follows: d has a relation $R[TA_1...A_n]$ with two tuples $tu_1...u_n$, $tv_1...v_n$, and for each clause of φ, say $c_1=x_1\vee x_2\vee(\neg x_3)$, d has a relation $R[TA_4...A_nX_1...X_n]$, with a tuple $fw_4...w_na_1a_2\ b_3z_4...z_n$. E contains $X_i=X_i{\cdot}A_i$, i=1,...,n, and for clause c_1 it contains $X_1{\cdot}X_2{\cdot}X_3=X_1{\cdot}\ X_2{\cdot}X_3{\cdot}T$. Thus, E_F consists of $X_i\rightarrow A_i$, i=1,...,n, and $X_1X_2X_3\rightarrow T$ for clause c_1. Figure 3 shows an example for n=4.

We now show that φ is satisfiable iff relation $R[TA_1...A_nX_1...\ X_n]$ (see Figure 3) can be filled in so that no new symbols are introduced in any column, and the FD's in E_F are satisfied (by Theorem 2 (b), this is equivalent to existence of an $\mathfrak{I}$ such that $\mathfrak{I}\models d,E,CAD,CAP$). Observe that for each i, $s_1[X_i]\neq s_2[X_i]$, $s_1[X_i]\neq z_i$, $s_2[X_i]\neq z_i$ (because of the FD $X_i\rightarrow A_i$), and thus $\{s_1[X_i], s_2[X_i]\}=\{a_i, b_i\}$. Make x_i *true* if $s_1[X_i]=a_i$, *false* if $s_1[X_i]=b_i$. It is easy to see that, because of the FD $X_1X_2X_3\rightarrow T$ and the tuple s_3, we will only be able to fill in the rest of the values iff the above truth assignment makes one literal of c_1 true and one false. ∎

We remark that, since the partition interpretation $\mathfrak{I}$ defined from a weak instance w satisfies CAP, introducing CAP as a requirement does not change anything; i.e., there is an $\mathfrak{I}$ such that $\mathfrak{I}\models d,E,CAP$ iff there is a weak instance for d satisfying E_F. However, having CAD alone results in a somewhat weaker condition than having CAD and CAP:

Theorem 1':
b.' There is an $\mathfrak{I}$ such that $\mathfrak{I}\models d,E,CAD$ iff there is a weak instance w for d which satisfies E_F and for each tuple t of w and each FD $X\rightarrow Y$ in E_F, if $t[A]\in d[A]$ for each A in X, then $t[B]\in d[B]$ for each B in Y.

Proof: By the same construction as for Theorem 1 (a). ∎

On the other hand, this new condition is no easier to test:

Theorem 2': Given d,E it is NP-complete to test if there is an $\mathfrak{I}$ such that $\mathfrak{I}\models d,E,CAD$.

Proof: By a slight modification of the reduction in the Proof of Theorem 2. ∎

Finally observe that, if d consists of a single relation, all conditions collapse to one: $d\models E_F$.

3.2. Expressive Power of PD's

We now restrict attention to databases consisting of a single relation r: We want to study what kinds of things can be said about r by sets of PD's. To do this, however, we first have to define what it *means* for r to satisfy a PD. We make use of a "standard" partition interpretation of r:

Definition 3: Let r be a relation over $\mathfrak{U}$; I(r) is the following partition interpretation:

1. for each A in $\mathfrak{U}$, $p(A)=\{i_t \mid t \text{ is a tuple of } r\}$.

2. for each a in $\mathfrak{D}$, $f_A(a)=\{i_t \mid t[A]=a\}$.

Partition π_A is induced by f_A. Note that I(r) satisfies CAP. We also define, given a partition interpretation $\mathfrak{I}$, a "standard" relation that corresponds to it:

Definition 4: Let $\mathfrak{I}$ be a partition interpretation over $\mathfrak{U}$; *R($\mathfrak{I}$) is a relation containing a tuple* t_i *for each i in* $\cup_{A\in\mathfrak{U}}p(A)$, where $t_i[A]=a$ if $i\in f_A(a)\in\pi_A$, and $t_i[A]=i$ otherwise.

Observe that $R(I(r))=r$, but $I(R(\mathfrak{I}))$ is not necessarily $\mathfrak{I}$. Also $I(r)\models r$ for any relation r. These definitions are motivated by the Proof of Theorem 1. By the argument there, $r\models X\rightarrow Y$ implies $I(r)\models X=X\cdot Y$, and $\mathfrak{I}\models X=X\cdot Y$ implies $R(\mathfrak{I})\models X\rightarrow Y$. Thus, if $I(r)\models X=X\cdot Y$ then $R(I(r))\models X\rightarrow Y$, i.e. $r\models X\rightarrow Y$. This shows that $r\models X\rightarrow Y$ *iff* $I(r)\models X=X\cdot Y$. This supports our claim that the FPD $X=X\cdot Y$ is the correct counterpart of the FD $X\rightarrow Y$. Taking this observation one step further, we introduce the following

Definition 5: A relation r satisfies a PD e (notation: $r\models_{rel}e$) iff $I(r)\models e$.

Given this definition, one easily sees the following:

1. $r\models_{rel}C=A\cdot B$ iff for any tuples t,s∈r, ($t[C]=s[C]$ iff $t[A]=s[A]$ *and* $t[B]=s[B]$).

2. $r\models_{rel}C=A+B$ iff for any tuples t,s∈r, ($t[C]=s[C]$ iff there is a sequence $s_0,\ldots,s_n$ of tuples of r with $t=s_0$, $s_n=s$, and for $i=0,\ldots,n-1$ $s_i[A]=s_{i+1}[A]$ *or* $s_i[B]=s_{i+1}[B]$).

Having reduced our models to relations, it becomes evident from (1) that $r\models_{rel}X=X\cdot Y$ iff $r\models X\rightarrow Y$, and from (2) that the relation of Example 4 correctly represents the transitive closure of an undirected graph iff it satisfies COMPONENT = HEAD + TAIL.

We now want to compare the expressive power of PD's to that of previously studied database constraints, namely EID's [5]. Let us say that an EID σ *is expressed* by a set E of PD's iff for any relation r, $r\models\sigma$ iff $r\models_{rel}E$. From the algebraic properties of $\cdot$, the PD $X=Y\cdot Z$ is equivalent to $X=X\cdot Y\cdot Z \wedge Y\cdot Z=X\cdot Y\cdot Z$, and therefore it is expressed by the set $\{X\rightarrow YZ, YZ\rightarrow X\}$. However, because of Example 4 it should come as no surprise [1] that the PD $C=A+B$ cannot be expressed by any set of EID's:

Theorem 3: Let $\mathfrak{U}=ABC$; the PD $C=A+B$ cannot be expressed by any set of first-order sentences.

Proof: Let Σ be a set of first-order sentences (over a single ternary relation symbol R) which expresses $C=A+B$. For $k\geq 1$, let φ_k be the following first-order formula:

"$t[C]=s[C]$ and there is *no* sequence $s_0,\ldots,s_k$ such that $t=s_0$, $s_k=s$, and for $i=0,\ldots,k-1$, $s_i[A]=s_{i+1}[A]$ or $s_i[B]=s_{i+1}[B]$"

(one can easily write φ_k). Observe that the relation r in Figure 4 is a *model* for $\Sigma\cup\{\varphi_k\}$: $r\models_{rel}C=A+B$ so $r\models\Sigma$, and clearly $r\models\varphi_k$. Thus, any finite subset of $\Sigma'=\Sigma\cup\{\varphi_k: k\geq 1\}$ has a model, and thus by the Compactness Theorem [4] Σ' has a model, say r'. But this is a contradiction, since r' $\models\Sigma$ and thus r' satisfies $C=A+B$, and on the other hand r' $\models\varphi_k$ *for all* $k\geq 1$ and therefore it does not satisfy $C=A+B$. ∎

On the other hand, an EID as simple as an MVD cannot be expressed by PD's:

Theorem 4: Let $\mathfrak{U}=ABC$; the MVD $A\rightarrow\rightarrow B$ cannot by expressed by any set of PD's.

Proof: Let E be a set of PD's which expresses $A\rightarrow\rightarrow B$. Referring to Figure 5, relation r_1 satisfies $A\rightarrow\rightarrow B$, so $I(r_1)\models E$, and $L(I(r_1))$ (the lattice obtained from $I(r_1)$ by

closing under sums and products) satisfies E. On the other hand, r_2 does not satisfy $A \twoheadrightarrow B$ so $I(r_2)$ does not satisfy E, and $L(I(r_2))$ does not satisfy E. But this is a contradiction, because $L(I(r_1))$, $L(I(r_2))$ are *isomorphic*, and thus they satisfy exactly the same PD's. ∎

4. Dependency Inference

Given a set E of PD's and a PD e, we want to know if $E \models_{rel} e$, i.e. if e holds in every relation that satisfies E. We first observe that this question can be approached as a (uniform) word problem for a certain class of algebraic structures, namely *lattices.*

Lemma 1: $E \models_{rel} e$ iff $E \models_{lat} e$, i.e. iff e holds in every *lattice* that satisfies E.

Proof: ($\Leftarrow$): Suppose $E \models_{lat} e$, and let r be a relation that satisfies E. Then $I(r) \models E$, and thus the lattice L(I(r)) obtained by I(r) by closing under sums and products satisfies E. But then e holds in I(r), and thus r satisfies e.

($\Rightarrow$): Suppose $E \models_{rel} e$, and let L be a lattice satisfying E. L is isomorphic to a sublattice of the lattice of partitions of some set X [3]. Translated into our terminology, this means that we can find a partition interpretation $\mathfrak{I}$, *satisfying* CAP, such that L is isomorphic to $L(\mathfrak{I})$. Thus, $\mathfrak{I} \models E$. Now consider the relation $R(\mathfrak{I}) = r$: since $\mathfrak{I} \models$ CAP, $I(R(\mathfrak{I})) = \mathfrak{I}$, therefore $r \models_{rel} E$. By the hypothesis $r \models_{rel} e$, which means $I(r) = \mathfrak{I} \models e$. Thus $L(\mathfrak{I}) \models e$, and e holds in L. ∎

Thus, $E \models_{lat} e$ can be viewed as a *word problem*, since a set with two binary operations $+, \cdot$ is a lattice iff the following set of axioms (LA) is satisfied:

1. $x+x=x$, $x \cdot x = x$ (idempotency)
2. $x+y=y+x$, $x \cdot y = y \cdot x$ (commutativity)
3. $x+(y+z)=(x+y)+z$, $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ (associativity)
4. $x+(x \cdot y)=x$, $x \cdot (x+y) = x$ (absorption)

I.e., $E \models_{lat} e$ iff e is implied from $E \cup LA$.

In particular, let e_σ be the FPD corresponding to an FD σ (e_σ is $X = X \cdot Y$ if σ is $X \rightarrow Y$), and let E_Σ be the set of FPD's corresponding to a set of FD's Σ. Since $r \models \sigma$ iff $r \models_{rel} e_\sigma$, $\Sigma \models \sigma$ iff $E_\Sigma \models_{rel} e_\sigma$. Thus, the implication problem for FD's can be reduced, in a straightforward way, to the (uniform) *word problem for idempotent commutative semigroups* (structures with a single associative, commutative and idempotent operator). On the other hand, since $X=Y$ is equivalent to $X = X \cdot Y$ *and* $Y = Y \cdot X$, we can also reduce the above word problem to the implication problem for FD's.

We now present a polynomial-time algorithm for the implication problem for PD's. Suppose we are given a set E of PD's, and a PD $p=q$: we want to test if $E \models_{lat} p=q$.

Consider the set $W(\mathcal{U})$ of partition expressions over $\mathcal{U}$, $+$, $\cdot$: we define several binary relations on $W(\mathcal{U})$. First, define $\leq_{id}$ (*identically* less-than-or-equal) inductively as follows:

1. $A \leq_{id} A$, A in $\mathcal{U}$.
2. if $p \leq_{id} r$, $q \leq_{id} r$ then $p+q \leq_{id} r$.
3. if $p \leq_{id} r$ then $p \cdot q \leq_{id} r$.
4. if $r \leq_{id} p$, $r \leq_{id} q$ then $r \leq_{id} p \cdot q$.
5. if $r \leq_{id} p$ then $r \leq_{id} p+q$.

($+, \cdot$ are meant here as uninterpreted operations on partition expressions, which return another expression).

Now define $=_{id}$ as follows: $p =_{id} q$ iff ($p \leq_{id} q$ *and* $q \leq_{id} p$).

(Intuitively, $p =_{id} q$ iff the PD $p=q$ holds *in all relations*).

The relation $=_{id}$ is an *equivalence relation*, and in particular it is a *congruence* (see [3] for a long and tedious proof): i.e., if $p_1 =_{id} q_1$, $p_2 =_{id} q_2$ then $p_1+p_2 =_{id} q_1+q_2$,

and $p_1 \cdot p_2 =_{id} q_1 \cdot q_2$. Thus, one can define $+, \cdot$ on the set of *equivalence classes* of $=_{id}$. The structure obtained this way is a *lattice* [3].

We now capture the effect of E: define the following relation $=_{sub}$ on $W(\mathfrak{U})$: $p =_{sub} q$ iff q can be obtained from p by substituting w_i for one or more occurences of z_i, where $z_i = w_i$ ($w_i = z_i$) is in E, $i=0,...,n$ for some n. It is easily verified that $=_{sub}$ is a congruence.

Now define $=_E$ as the *sum* of $=_{id}$, $=_{sub}$: this means that the equivalence classes of $=_E$ form a partition of $W(\mathfrak{U})$ which is the sum of the partitions induced by $=_{id}$, $=_{sub}$. In other words, $p =_E q$ iff there is a sequence $q_0,...,q_n$ such that $p = q_0$, $q_n = q$, and for $i=0,...,n-1$ $q_i =_{id} q_{i+1}$ *or* $q_i =_{sub} q_{i+1}$.

As a sum of two congruences, $=_E$ is also a congruence [7]. One can also observe that the equivalence classes of $=_E$ form a lattice L_E under the induced $+, \cdot$: just check LA, e.g. $p+p =_E p$ because $p+p =_{id} p$, and in general if $p =_{id} q$ then $p =_E q$. Note that L_E satisfies the PD $p=q$ iff $p =_E q$.

We now show that the relation $=_E$ captures the PD's implied by E, and give a proof system for implication of PD's:

Lemma 2: The following statements are equivalent:

a. $E \models_{lat} p=q$

b. $p =_E q$

c. $p \leq_E q$ and $q \leq_E p$ can be proved using the following rules:

1. $A \leq_E A$, A in $\mathfrak{U}$.

2. $z \leq_E w$, $w \leq_E z$ for $z=w$ in E.

3. *from* $p \leq_E q$, $q \leq_E r$ *derive* $p \leq_E r$.

4. *from* $p \leq_E r$, $q \leq_E r$ *derive* $p+q \leq_E r$.

5. *from* $p \leq_E r$ *derive* $p \cdot q \leq_E r$.

6. *from* $r \leq_E p$, $r \leq_E q$ *derive* $r \leq_E p \cdot q$.

7. *from* $r \leq_E p$ *derive* $r \leq_E p+q$.

Proof: (a)⇒(b): if $p \neq_E q$, then L_E does not satisfy $p=q$, whereas it satisfies E. Thus, L_E is a *counterexample* to $E \models_{lat} p=q$.

(b)⇒(c): by an easy induction on the definition of $=_E$.

(c)⇒(a): it is easy to see (by induction) that if $p \leq_E q$ can be proved, then $p \leq q$ in every lattice satisfying E. Thus, $p=q$ in every lattice satisfying E, i.e. $E \models_{lat} p=q$. ∎

We can now prove our main result:

Theorem 5: There is a polynomial-time algorithm for inference of PD's.

Proof: Observe that, if there is a proof that $p \leq_E q$, then this proof need only mention subexpressions of p, q, and of the expressions appearing in E. Thus, we can just write down these expressions (say, as in [11]) and repeatedly apply the rules, until no new inference can be made. ∎

Since inference of FD's can be seen as a special case of inference of PD's, the problem is actually *polynomial-time complete* [17]. However, in the special case where E is empty it can be solved in *logarithmic space* [9] as follows: we first rewrite $p=q$ as a Boolean tree with leaves of the form $A \leq B$, A,B in $\mathfrak{U}$. We then replace $A \leq A$ by *true* and $A \leq B$ by *false* if $A \neq B$, and evaluate the resulting tree.

Example: $A+B=C \cdot D$ is (recursively) rewritten as

$$A+B \leq C \cdot D \wedge C \cdot D \leq A+B$$

$$(A+B \leq C \wedge A+B \leq D) \wedge (C \cdot D \leq A \vee C \cdot D \leq B)$$

$$((A \leq C \wedge B \leq C) \wedge (A \leq D \wedge B \leq D)) \wedge ((C \leq A \vee D \leq A) \vee (C \leq B \vee D \leq B))$$

5. Testing Satisfaction of PD's

Given a database d and a set of PD's E, we want to test if there is a partition interpretation $\mathfrak{I}$ such that $\mathfrak{I} \models d,E$. We first give a characterization in terms of (general) weak instances:

Lemma 3: There is an $\mathfrak{I}$ such that $\mathfrak{I} \models d,E$ iff there is a weak instance for d satisfying E.

Proof: Let w be a weak instance for d satisfying E. It is then easy to see (as in the Proof of Theorem 1) that $I(w) \models d$, and of course $I(w) \models E$.

Conversely, let $\mathfrak{I}$ be a partition interpretation satisfying d,E. Consider the relation $R(\mathfrak{I})$: As in the Proof of Theorem 1, $R(\mathfrak{I})$ is a weak instance for d, and thus it remains to show that it satisfies E, i.e. that $I(R(\mathfrak{I})) \models E$. Now this partition interpretation may be different from $\mathfrak{I}$, since it satisfies CAP while $\mathfrak{I}$ in general does not; however, one can observe that $L(I(R(\mathfrak{I})))$ is *isomorphic* to $L(\mathfrak{I})$, and therefore $I(R(\mathfrak{I})) \models E$ since $\mathfrak{I} \models E$. ∎

We now outline how to test whether there exists a weak instance for d satisfying E.

First, we replace E by a set E'of PD's of the form $C=A \cdot B$ or $C=A+B$, where A,B,C are attributes from a universe $\mathfrak{U}'$ containing $\mathfrak{U}$: this is done by (recursively) replacing $X=Y \cdot Z$ by the PD's $X=C$, $Y=A$, $Z=B$, $C=A \cdot B$, where A,B,C are *new* attribute names. It is easy to check that there is a weak instance for d satisfying E iff there is a weak instance for d satisfying E'.

Let us denote by $p \rightarrow q$, where p,q are partition expressions, the PD $p=p \cdot q$. This slight abuse of notation is consistent, since the FPD $X \rightarrow Y$ is actually *equivalent* to the FD $X \rightarrow Y$. Now a PD $C=A \cdot B$ in E'can be replaced by the FPD's $C \rightarrow AB$, $AB \rightarrow C$, and a PD $C=A+B$ in E'can be replaced by the PD's $A+B \rightarrow C$, $C \rightarrow A+B$; furthermore, the PD $A+B \rightarrow C$ can be replaced by the FPD's $A \rightarrow C$, $B \rightarrow C$. We now have a set F consisting of FPD's and of PD's of the form $C \rightarrow A+B$, and it is obvious that there is a weak instance for d satisfying E'iff there is a weak instance for d satisfying F.

Now compute (using the algorithm of the previous Section) all *consequences* of F of the form $A \rightarrow B$, A,B in $\mathfrak{U}'$, and add them to F. Furthermore, if now F contains $A \rightarrow B$ and $C \rightarrow A+B$, replace $C \rightarrow A+B$ by $C \rightarrow B$. Let F'be the set of FPD's in F. The crucial fact is given in the following

Lemma 4: There is a weak instance for d satisfying E'iff there is a weak instance for d satisfying F'.

Proof (Sketch): The "only if" direction is obvious. For the converse, let w be a weak instance for d satisfying F', and assume that w does not satisfy some PD $C \rightarrow A+B$ in E'. We can pick two tuples t,s such that $t[ABC]=a_1b_1c$, $s[ABC]=a_2b_2c$, and add to w a new tuple s such that $s[ABC]=a_1b_2c$ and the relation w_1 obtained satisfies F'. But then we can repeate the argument to obtain relations w_2, w_3 and so on. The relation w_ω obtained after an infinite number of steps is a weak instance for d satisfying E', because any violation of some PD $C \rightarrow A+B$ appearing at any stage has been taken care of at some later stage. ∎

Theorem 6: There is a polynomial-time algorithm to test whether a given database d is consistent with a set E of PD's.

Proof: Using the polynomial-time algorithm for inference of PD's given in Section 4, we can construct the set F'. By Lemma 4, we can then use the chase algorithm of [8] to test if d is consistent with F'. ∎

6. Conclusions

We have shown that: (1) the inference problem for PD's and (2) the problem of testing consistency of a set of relations with a set of PD's are in polynomial time. Both proofs use algebraic techniques and make use of finite and

infinite relations. If we restrict ourselves to finite relations we are faced with hard questions about finite realizations of lattices [3].

We would like to point out that the FD inference problem can be formulated, in a straightforward fashion, as a special case of the generator problem for finitely presented algebras [11]. In our analysis FD inference is a word problem in a lattice, which reveals much more of its algebraic structure.

References

1. Aho, A.V. and Ullman, J.D. "Universality of Data Retrieval Languages". *Proceedings of the Sixth ACM Symposium on Principles of Programming Languages, ACM* (1979).

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

3. Crawley, P. and Dilworth, R.P.. *Algebraic Theory of Lattices.* Prentice-Hall, Inc., , 1973.

4. Enderton, H.B. *A Mathematical Introduction to Logic.* Academic Press, Inc., , 1972.

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

6. Garey, M.R. and Johnson, D.S.. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, , 1979.

7. Grätzer, G.. *Universal Algebra.* Springer-Verlag New York Inc., , 1979.

8. Honeyman, P. "Testing Satisfaction of Functional Dependencies". *Journal of the ACM 29*, 3 (July 1982), 668-677..

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

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

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

12. Maier, D.. *The Theory of Relational Databases.* Computer Science Press, Inc., , 1983.

13. Reiter R. On Closed World Databases. In *Logic and Databases*, Plenum Press, , 1978..

14. Sadri, F. and Ullman, J.D., "Template Dependencies: A Large Class of Dependencies in Relational Databases and its Complete Axiomatization". *JACM* (1982).

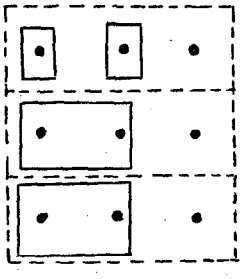
15. Spyratos, N. "The Partition Model: A Deductive Database Model". *INRIA Research Report No. 286* (April 1984), .

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

17. Vardi, M.Y. "Personal Communication". ().

18. Vassiliou, Y. *A Formal Treatment of Imperfect Information in Database Management.* Ph.D. Th., University of Toronto, 1980.

19. Yannakakis, M. and Papadimitriou, C.H. "Algebraic Dependencies". *Journal of Computer and System Sciences 25*, 1 (August 1982), 2-41..



—— partition π

----- partition π'

Figure 1

A	B	C
a	b	c
a_2	b_1	c
a_2	b_1	c_1
a_1	b	c_1

$A = A \cdot B$

(2a)

Database d and E

$p(A) = p(B) = p(C) = \{1,2,3,4\}$

$\pi_A = \{\{1\},\{4\},\{2,3\}\}$

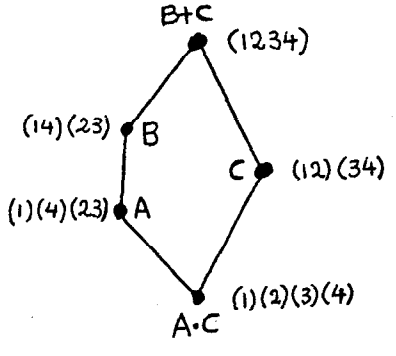
$\pi_B = \{\{1,4\},\{2,3\}\}$

$\pi_C = \{\{1,2\},\{3,4\}\}$

f_A		f_B		f_C	
a	$\{1\}$	b	$\{1,4\}$	c	$\{1,2\}$
a_1	$\{4\}$	b_1	$\{2,3\}$	c_1	$\{3,4\}$
a_2	$\{2,3\}$	else	$\emptyset$	else	$\emptyset$
else	$\emptyset$				

(2b)

$\mathcal{J} \models d, E, CAD, CAP$



(2c)

The lattice $L(\mathcal{J})$ is not distributive

$B \cdot (A+C) \neq (B \cdot A) + (B \cdot C)$

Figure 2

T	A_1	A_2	A_3	A_4
t	u_1	u_2	u_3	u_4
t	v_1	v_2	v_3	v_4

T	A_4	X_1	X_2	X_3	X_4
f	w_4	a_1	a_2	b_3	z_4

E_F: $X_i \to A_i$, $i = 1, \ldots, 4$

$X_1 X_2 X_3 \to T$

$c_1 = x_1 \vee x_2 \vee (\neg x_3)$

	T	A_1	A_2	A_3	A_4	X_1	X_2	X_3	X_4
s_1:	t	u_1	u_2	u_3	u_4				
s_2:	t	v_1	v_2	v_3	v_4				
s_3:	f				w_4	a_1	a_2	b_3	z_4

Figure 3

r:

	A	B	C
t:	1	2	0
	3	2	0
	3	4	0
	5	4	0
	⋮		
	K-1	K	0
	K+1	K	0
s:	K+1	K+2	0

$r \models \Sigma, \varphi_K$

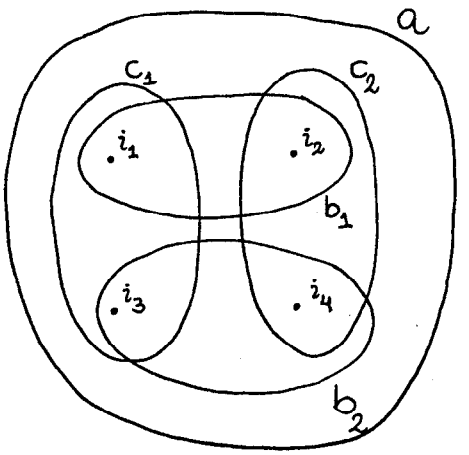
(K even)

Figure 4

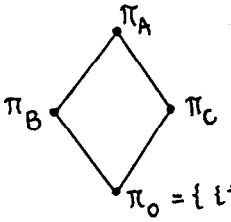
r_1 :

	A	B	C
1:	a	b_1	c_1
2:	a	b_1	c_2
3:	a	b_2	c_1
4:	a	b_2	c_2

$I(r_1)$:



$L(I(r_1))$:

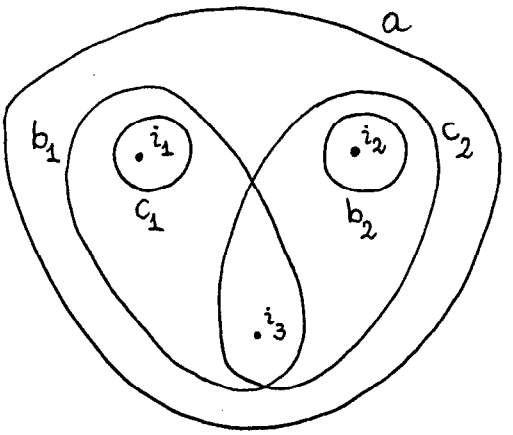


$\pi_0 = \{ \{i_1\}, \{i_2\}, \{i_3\}, \{i_4\} \}$

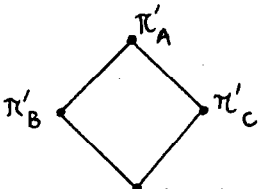
r_2 :

	A	B	C
1:	a	b_1	c_1
2:	a	b_2	c_2
3:	a	b_1	c_2

$I(r_2)$:



$L(I(r_2))$:



$\pi'_0 = \{ \{i_1\}, \{i_2\}, \{i_3\} \}$

Figure 5