INDEPENDENT DATABASE SCHEMES UNDER FUNCTIONAL AND INCLUSION DEPENDENCIES

Paolo Atzeni
IASI-CNR
Viale Manzoni 30
00185 Roma, Italy

Edward P.F. Chan †
Department of Computing Science
The University of Alberta
Edmonton, Alberta T6G 2H1, Canada

Abstract

In a context considering in a unique framework all the relations in a database, by means of the notion of global consistency, independent database schemes allow enforcement of constraints to be performed locally, thus providing independent updatability of the various relations. Independent schemes have hitherto been studied in the presence of functional and join dependencies. In this paper we extend the definition and give some characterizations when the involved set of constraints is composed of functional and inclusion dependencies.

1. Introduction

In a relational database, the information about an enterprise is represented by a set of tables or relations. Although information is generated from one or more relations, it is highly desirable that relations in a database can be updated independently. Independent updatability of relations has long been recognized as an important goal in the database design process. Recently, a class of database schemes called *independent schemes* ([9], [11], [18]) was proposed to allow enforcement of constraints to be performed locally in each relation, thus capturing an essential part of independent updatability. Informally, a database scheme is independent if local satisfaction of constraints in each relation implies global consistency of data. This class of database schemes generated a great deal of interest and was widely accepted to be a desirable criterion in designing a relational database ([1], [2], [6], [11], [15], [20], [21]).

All published work involving independence considers, as possible constraints, functional dependencies and, in some cases, the full join dependency. Whereas functional and join dependencies are certainly important, there is a general consensus that functional and inclusion dependencies are probably the two most common kinds of constraints in a relational database ([3], [4], [8], [12], [13], [17]). This paper is an attempt to bridge this gap by extending the idea of independence to functional and inclusion dependencies. It is hoped that this work would give a contribution towards a more realistic theory for designing relational databases. It should be pointed out that inclusion dependencies are by nature inter-relational and therefore the meaning of global satisfaction in our case is slightly different from the one studied previously. In Section 2, we first briefly introduce the notation needed throughout this paper. In Section 3, we define the meaning of independence in the presence of inclusion dependencies and show how this new notion is related to the classical one. The interaction between functional and inclusion dependencies is inherently difficult to handle ([7], [17]); so, following other authors, we restrict our attention to two particular cases: key-based dependencies ([12]) and functional and unary inclusion dependencies ([13]). These two cases are investigated respectively in Sections 4 and 5 and a characterization of independence is obtained in each case. These results should provide insight into the design process of independent schemes under our assumptions. As a by-product, we derive an algorithm which converts a state that conforms to certain conditions to a state that satisfies both the functional and unary inclusion dependencies. This is a generalized

† Work supported by the Natural Sciences and Engineering Research Council of Canada. Present address: Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada

version of the algorithm presented by [13] and might be proven to be fundamental in query optimization and schema analysis when functional and unary inclusion dependencies are considered. Finally, we give our conclusions in Section 6. Due to space limitations, we omit the standard definitions, and sketch most of the proofs.

2. Background Definitions and Notation

For the sake of brevity, we present only the nonstandard definitions: for example, we omit the definitions of attribute, relation scheme, tuple, relation, tableau (i.e. relation with variables), etc, which can be found in standard textbooks ([14], [22]).

2.1. Inclusion Dependencies

The constraints we are concerned with in this paper are functional dependencies and inclusion dependencies. Functional dependencies are well known, so we concentrate on inclusion dependencies. Given a database scheme $\mathbf{R}$, an *inclusion dependency* (IND) is a statement of the form $R_i[Y] \supseteq R_j[Z]$, where $R_i(X_i), R_j(X_j) \in \mathbf{R}$ and Y, Z are sequences of attributes, with the same length, respectively from X_i, X_j; within each sequence the attributes are distinct. Let $Y = B_1B_2...B_k$, $Z = C_1C_2...C_k$; a pair of relations, r_i, r_j satisfies this IND if, for every tuple $t_j \in r_j$, there is a tuple $t_i \in r_i$ such that for $t_i.B_h = t_j.C_h$, for $1 \leq h \leq k$ (or, if Y, Z are considered as sequences, $t_i.Y = t_j.Z$). In this paper, we associate with the database scheme a set of constraints $\Sigma = F \cup I$, where F is a set of FDs, each of which is *tagged* with a relation scheme (e.g., $R_i : Y \rightarrow Z$), to indicate the relation that has to satisfy it, and I is a set of INDs. A state of $\mathbf{R}$ is called *locally consistent* if its relations satisfy all the constraints in Σ.

Given a set Σ of constraints and a constraint σ, we say that Σ *implies* σ if every state that satisfies Σ also satisfies σ. If relations are required to be *finite* (i.e., to contain a finite number of tuples), then we use the term *finite implication*; for the other case we use the term *unrestricted implication*. Since we are interested in the practical applications of our results, in the rest of the paper we will always refer to finite implication. The set of constraints implied by Σ is called the *closure* of Σ and is indicated with Σ^+. Two sets of constraints are *equivalent* if their closures are equal.

The implication of FDs is widely known, so we do not discuss it. INDs were considered from a formal point of view much more recently than FDs. The implication of INDs and the interaction of FDs and INDs were first studied by [4]. Both the implication and the finite implication problem for the joint class of FDs and INDs were shown to be undecidable independently by [7] and [17]. For this reason, many authors (including [5], [12], [13]) studied restricted classes of INDs, and their interaction with restricted classes of FDs. It should be noted that, as opposed to what happens for FDs, the unrestricted implication problem and the finite implication problem for the joint class of FDs and INDs are not the same, as it can be shown by means of simple examples ([4]).

Among the restricted classes of INDs, some are particularly meaningful. The IND $R_i[Y] \supseteq R_j[Z]$ is *typed* if the two sequences Y and Z are equal. An IND is *unary* if the involved sequences are singletons: $R_i[A] \supseteq R_j[B]$, where A and B are single attributes. A set I of inclusion dependencies is *acyclic* if the directed graph $\mathcal{G}(I) = (N, E)$, with the relation names as nodes in N and an edge from R_i to R_j if there is an IND $R_i[X] \supseteq R_j[Y] \in I$, is acyclic.

2.2. The Weak Instance Model

The *weak instance model* is an approach to the relational model aimed at viewing the various relations within a database state in a unified framework. It is based on the notion of global satisfaction of constraints, where constraints (in our case FDs, but the extension to other kinds of embedded implicational dependencies is possible) are defined over attributes in the universe U, rather than over single relation schemes: the FDs in this case do not have tags associated with them, so we will sometimes use the term *untagged* FDs. Given a set F of untagged FDs, a relation $w(U)$ is a *weak instance* for a state $\mathbf{r}$ of $\mathbf{R}$ if w satisfies F and, for every $R_i(X_i) \in \mathbf{R}$, $r_i \subseteq \pi_{X_i}(w)$. The state $\mathbf{r}$ is said *(globally) consistent* with respect to (wrt) F if it has a weak instance wrt F. A practical method for testing for global consistency of FDs is based on the notions of state tableau and chase ([10]).

The concept of *independence* relates the notions of local and global satisfaction of dependencies. In the literature, it has been discussed with respect to FDs and sometimes with other dependencies, not including INDs. In the rest of the paper we will study independence when INDs are present, but now we give (a slight variation of) the standard definition of independence wrt FDs.

An untagged FD $X \rightarrow A$ is said to be *embedded* in a relation scheme $R(Y)$ if $XA \subseteq Y$. A set of untagged FDs is *embedded* in a database scheme $\mathbf{R}$ if each of them is embedded in some relation scheme of $\mathbf{R}$. (In the rest of this paragraph, we will use F_U and G_U to denote sets of untagged FDs). The projection of a set of FDs F_U onto R_i, denoted by $F_U^+|R_i$, is the set of the FDs in F_U^+ that are embedded in R_i. A database scheme $\mathbf{R}$ is said to be *cover embedding* for a set F_U of FDs if there exists a set G_U of FDs embedded in $\mathbf{R}$ such that $G_U^+ = F_U^+$. Given a set F_U, embedded in a database scheme $\mathbf{R}$, let F be the set of tagged FDs obtained from F_U by tagging each FD with the relation schemes it is embedded in (if an FD is embedded in more than one relation scheme, it appears as many times in F). The scheme $\mathbf{R}$ is *independent* with respect to F_U if each state that locally satisfies F is globally consistent wrt F_U. Similarly, given a set F of tagged FDs, we can define F_U by removing the tags and give an identical definition of independence wrt F. Independence was first proposed by [18], who gave necessary and sufficient conditions for it, when the only constraints are FDs represented by keys; [9] presented a number of results on independence, including a polynomial time test for independence with respect to a set of embedded FDs. More efficient algorithms were later presented by [11] and [20].

3. The Definition of Independence in Presence of INDs

In this section, we give a definition of independence wrt a set of FDs and INDs. Let $\mathbf{R} = \{R_1, R_2, ...R_n\}$ be a database scheme, $\Sigma = F \cup I$, where F is a set of tagged FDs, and I a set of INDs over $\mathbf{R}$. In this case the idea of defining independence on the basis of the untagged FDs corresponding to F does not quite work, since, as shown in the following example, there may be FDs implied due to the interaction with INDs, which cannot be taken into account in this way.

Example 3.1. Let the following attribute names be abbreviated by means of the respective initials: *Manager, Dept, Project, Employee.* Let $\mathbf{R} = \{R_1(MD), R_2(DP), R_3(MP), R_4(ED)\}$, $F = \{R_2 : D \rightarrow P, R_4 : E \rightarrow D\}$, $I = \{R_4[ED] \supseteq R_1[MD]\}$. In this case, the set F_U of untagged FDs would contain $D \rightarrow P$ and $E \rightarrow D$, and so it is easy to see that any state that is locally consistent wrt $\Sigma = F \cup I$ has a weak instance wrt F_U. However, Σ^+ contains also the FD $R_1 : M \rightarrow D$, and so the untagged FDs $M \rightarrow D$, $D \rightarrow P$, in the relation schemes $R_1(MD)$, $R_2(DP)$, $R_3(MP)$ cause a violation of the intuitive notion of independence. □

As a consequence, we have to consider the FDs implied by the interaction with INDs. For every i, $1 \leq i \leq n$, let G_i be a cover of the FDs with tag R_i in Σ^+; then let $G = G_1 \cup ... \cup G_n$, and G_U be the set of untagged FDs corresponding to G. Then, we say that a state $\mathbf{r}$ is *globally consistent* wrt Σ if it is globally consistent wrt G_U (i.e., it has a weak instance wrt G_U). Finally, a database scheme $\mathbf{R}$ is *independent wrt* Σ if every state $\mathbf{r}$ of $\mathbf{R}$ that is locally consistent wrt Σ is globally consistent wrt Σ.

(In the following, when no confusion can arise, we will omit the distinction between a set of tagged FDs and the corresponding set of untagged FDs).

The following two lemmas, whose easy proofs are omitted, describe some interesting properties about our definition of independence.

Lemma 3.1. *Let Σ and G be as above. If* $\mathbf{R}$ *is independent wrt G then* $\mathbf{R}$ *is independent wrt Σ.*

Lemma 3.2. *Let Σ and G be as above, and let $G^+ = F^+$. Then, for every* $\mathbf{R}$, *if* $\mathbf{R}$ *is independent wrt F then* $\mathbf{R}$ *is independent wrt Σ.*

Let us note that in general it is not the case that independence wrt F implies independence wrt Σ, because, as shown in example 3.1, there may be FDs in G that cause nonindependence.

The definition we have just given has one drawback: since in general the implication (as well as the finite implication) problem for INDs and FDs is undecidable, the set G of FDs need not be computable. As a consequence, following what has been done for other problems related to INDs, we will consider restricted classes of FDs and INDs, for which the implication problem is decidable.

4. Key-Based Dependencies.

The first kind of restriction we consider on FDs and INDs is related to the concept of key. [12] called a set Σ *key-based* if

(a) For every relation scheme $R_i(X_i) \in \mathbf{R}$, the FDs with tag R_i in F all have the same left-hand side K_i, and every attribute $A \in X_i - K_i$ is in the right-hand side Z of some FD in $R_i : K_i \rightarrow Z \in F$. (This means that, for every i, K_i is the (only) key for R_i).

(b) For each IND $R_j[Y] \supseteq R_i[X] \in I$, the set Y is contained in the key K_j of R_j, and the set X is disjoint from the key K_i of R_i.

Under these assumptions, [12] obtained a number of results on the containment of conjunctive queries. We will study a variant of the class of key-based dependencies, obtained by replacing assumption (b) with the following (while keeping assumption (a)):

(c) The set of INDs I is acyclic, and, for each IND $\sigma : R_j[Y] \supseteq R_i[X] \in I$, at least one of the following holds: (i) σ is typed; or (ii) it is not the case that Y properly contains the key K_j of R_j.

Since in both cases the restrictions are based on keys, and on the relationship between the sequences of attributes in the INDs and the keys, the term *key-based* would be suitable for both. In order to avoid confusion, we will refer to the assumptions that are made on the constraints with the names we introduced above: (a), (b), (c). In the rest of the section we study independence under assumptions (a), (c), but similar results can be obtained if (a) and (b) are assumed.

Since we deal with the weak instance model, and so want to give a unique meaning to each set of attributes, we assume that if the key K_i and a non-key attribute A of some relation scheme R_i are both contained in some other relation scheme $R_j(X_j)$, then the FD $R_j : K_i \to A$ is also in F^+, and so R_i and R_j have the same key. (Note that this does not mean that if the key of a relation scheme is contained in another relation scheme, then it is also a key for it: it is a key if also a non-key attribute is contained in both relation schemes).

Example 4.1. Let the following attribute names be abbreviated by means of the capitalized letters: *EmployeeNo, Name, Address, Salary, CourseTutored, Rank, CourseNo, Description, Instructor*; and the following relation names be abbreviated by means of the respective initials: *EMPLOYEES, TUTORS, PROFESSORS, COURSES*. Let us consider the database scheme $\mathbf{R} = E(EN, N, A, S)$, $T(EN, N, CT)$, $P(EN, N, R)$, $C(CN, D, I)$; let the FDs be defined in such a way that EN is the key for the first three schemes and CN the key for the last one; let the following INDs be in Σ: $E[EN, A] \supseteq T[EN, A]$, $E[EN, A] \supseteq P[EN, A]$, $P[N] \supseteq C[I]$, $C[CN] \supseteq T[CT]$. Here, the FD $EN \to N$ is embedded in three relation schemes, and so $\mathbf{R}$ is not independent wrt the FDs defined on it. However, the INDs "force" the FD to express the same relationship in all the three relation schemes, and so it would be reasonable for this scheme to be independent. The results in this section show that $\mathbf{R}$ is in fact independent wrt Σ. □

As a first step towards studying independence in this context, we consider the interaction between FDs and INDs.

Lemma 4.1. *Let* $\mathbf{R}$ *and* $\Sigma = F \cup I$ *satisfy assumptions (a), (c). Then, for any tagged FD* f, $f \in F^+$ *if and only if* $f \in \Sigma^+$.

Proof. The *only if* part is immediate, so let us concentrate on the *if* part. Let f be $R_i : Z \to A$, an FD not in F^+. We show that it is not in Σ^+ either, by showing a state that satisfies Σ and does not satisfy f. It is immediate to see that Z is not the key (nor a superkey) of R_i. The counterexample state is obtaind by modifying, by means of a variation of the chase procedure, a state with all relations empty except r_i, which contains exactly two tuples, which agree on Z and disagree on the other attributes.

We chase the state wrt Σ by applying the following rule as long as there is an IND $R_T[X] \supseteq R_S[Y]$ violated. For each tuple $t_S \in r_S$ such that there is no tuple $t_T \in r_T$ with $t_S.Y = t_T.X$, a tuple t is added to r_T, with $t.X = t_S.Y$, and the other attributes defined as follows: for every attribute $C \in X_T - X$, if X is a key or a superkey for R_T, and there is an FD $R_T : K_T \to C$ such that K_TC is contained in some other relation scheme, say R_Q, and there is a tuple $t_Q \in r_Q$ with $t_Q.K_T = t.K_T$, then $t.C$ is defined with the same value as $t_Q.C$; otherwise, $t.C$ is defined as a new constant.

It is immediate to show that, since the set I of INDs is acyclic, the process does not add any tuple to R_i and eventually terminates, producing a state that satisfies all INDs in I. In order to complete the proof it is sufficient to show that the final state satisfies the FDs in F, and so satisfies Σ. The proof is by induction on the number of steps in the chase process, i.e., the number of tuples being added. We omit the details.

Summarizing, the final state satisfies both F and I, and so Σ, whereas it violates $R_i : Z \to A$, because of the two initial tuples in r_i: this means that this FD is not implied by Σ. □

Lemma 4.1 has the important consequence that the two sets of untagged FDs F_U and G_U mentioned in Section 3 are equivalent and so the definition of independence can be given as follows: $\mathbf{R}$ is independent if every state $\mathbf{r}$ of $\mathbf{R}$ that is locally consistent wrt Σ is globally consistent wrt F_U.

Suppose $X \to A$ is a nontrivial FD embedded in $\{S_1, ..., S_k\} \subseteq \mathbf{R}$. A tagged FD $S_j : X \to A$ is said to be a *maximal inclusion* for $X \to A$ if for all S_p, $1 \le p \le k$, $S_j[XA] \supseteq S_p[XA]$ is in Σ^+. Notice that in general there may be more than one maximal inclusion for an FD. In this case the maximal inclusions for the FD include each other. However if the INDs are acyclic and if a maximal inclusion exists, then it is unique. $\mathbf{R}$ is said to satisfy the *maximal inclusion condition* if for all nontrivial FDs $X \to A$ in Σ embedded in one or more relation schemes, then there is a unique maximal inclusion for $X \to A$.

Let $\mathbf{R} = \{R_1(X_1), ..., R_n(X_n)\}$ be a database scheme and F be a set of tagged FDs on $\mathbf{R}$, satisfying condition (a). For every j, $1 \le j \le n$, F_j is the subset of F containing the FDs with tag R_j. $\mathbf{R}$ is said to satisfy the *uniqueness condition* wrt F ([19]) if for all R_i and R_j there does not exist $A \in X_i^+ - X_i$ (X_i^+ indicates the closure of X_i wrt the untagged FDs in F) such that there exists a (nontrivial) FD $R_j : K_j \to A$ in F_j with the closure of X_i wrt the untagged FDs in $F - F_j$ contains KA. [19] showed that the uniqueness condition is a necessary and sufficient condition for independence wrt to a set of key-based FDs.

Let now $\mathbf{R}$ and $\Sigma = F \cup I$ satisfy conditions (a), (c), and the maximal inclusion condition. Then, $\mathbf{R}$ is said to satisfy the *maximality and uniqueness condition wrt* Σ if $\mathbf{R}$ satisfies the uniqueness condition wrt F', where F' is the set of unique maximal inclusions for the set of the key dependencies in F. In the rest of this section we show that the maximality and uniqueness condition is necessary and sufficient for independence in this framework.

Lemma 4.2. *Let* $\mathbf{R}$ *and* Σ *satisfy conditions (a)* and (c). If $\mathbf{R}$ does not satisfy the maximal inclusion condition, then it is not independent wrt Σ.

Proof. Since $\mathbf{R}$ does not satisfy the maximal inclusion condition, there are two distinct relation schemes R_i, R_j, such that both of them embed a key dependency $K \rightarrow A$ but neither $R_i[KA] \supseteq R_j[KA]$ nor $R_j[KA] \supseteq R_i[KA]$ is in Σ^+. Without loss of generality, we assume there is no R_p, $p \neq i$ and $p \neq j$, in $\mathbf{R}$, such that $R_p[KA] \supseteq R_i[KA]$ or $R_p[KA] \supseteq R_j[KA]$ is in Σ^+.

In order to show that $\mathbf{R}$ is not independent wrt Σ the proof shows a state that satisfies Σ and is not globally consistent. This counterexample state is defined by chasing, in a way similar to the one in lemma 4.1, a simple state, with a tuple t_i in R_i, a tuple t_j in R_j, and the other relations empty. The details, similar to those in the proof of lemma 4.1, are omitted. □

Lemma 4.3. *Let $\mathbf{R}$ and Σ satisfy conditions (a), (c), and the maximal inclusion condition. If $\mathbf{R}$ does not satisfy the uniqueness condition wrt F, then it is not independent wrt Σ.*

Proof. By lemma 4.1, the FDs implied by F are the same as those implied by Σ. Let $S_0(Y_0)$ and $S_m(Y_m)$ be two relation schemes that violate the uniqueness condition. Let G_0, G_m be the sets of FDs in F with tag S_0, S_m, respectively. Then, the closure of Y_0 wrt $F - G_m$ contains KA, where $S_m : K \rightarrow A$ is the maximal inclusion for $K \rightarrow A$. Let the sequence of maximal inclusions used in computing the closure be $S_1 : Y_1 \rightarrow A_1, ..., S_{m-1} : Y_{m-1} \rightarrow A_{m-1}$, with $A_{m-1} = A$. Again, the proof shows a counterexample state, which satisfies Σ and is not globally consistent. Such a state is obtained by chasing, in the same way as in lemma 4.2, an initial state, with one tuple t_i in each relation $r_i(X_i)$: if X_i contains KA then define $t_i.(X_i - A) =$ 1's and $t_i.A = 2$; otherwise, $t_i =$ 1's. The details, again similar to those in the proof of lemma 4.1, are omitted. □

Lemma 4.4. *Let $\mathbf{R}$ and Σ satisfy conditions (a), (c). If $\mathbf{R}$ satisfies the maximality and uniqueness condition, then it is independent wrt Σ.*

Proof. Since $\mathbf{R}$ satisfies the maximality and uniqueness condition, there exists a set G of unique maximal inclusions for the set of key dependencies in $\mathbf{R}$. Let us define a database scheme S from G: for each $R_i \in \mathbf{R}$, if there is an FD in G with tag R_i, then there is a relation scheme $S_i(Y_i)$ in S, where Y_i is the union of all the attributes involved in FDs in G with tag R_i. The difference between S_i and its corresponding R_i in $\mathbf{R}$ is that R_i may contain some nonprime attributes which S_i does not have. Also, note that the untagged sets of FDs corresponding to F and G, respectively, are the same.

Claim 1. *S satisfies the uniqueness condition wrt G.*Easy.

Claim 2. For each state $\mathbf{r}$ of $\mathbf{R}$ locally consistent wrt Σ, there is a state $\mathbf{s}$ of S satisfying G such that $\mathbf{r}$ is globally consistent wrt F (and so wrt Σ) if and only if $\mathbf{s}$ is globally consistent wrt G. The state $\mathbf{s}$ is built as follows: for each relation r_i in $\mathbf{r}$, $s_i = \pi_{Y_i}(r_i)$. Clearly the state $\mathbf{s}$ is locally consistent wrt G. Also, since $\mathbf{R}$ satisfies the maximality and uniqueness condition, and the untagged sets corresponding to F and G are equal, it is not difficult to see that $\mathbf{s}$ is globally consistent wrt G if and only if $\mathbf{r}$ is globally consistent wrt F: in fact, every relation $w(U)$ is a weak instance for $\mathbf{r}$ wrt F if and only if it is a weak instance for $\mathbf{s}$ wrt G.

Let $\mathbf{r}$ be a locally consistent state of $\mathbf{R}$ wrt Σ. By claim 2, there is a state $\mathbf{s}$ of S, satisfying G, such that $\mathbf{r}$ is globally consistent wrt Σ if and only if $\mathbf{s}$ is consistent wrt G. By claim 1, S is independent wrt G, and so $\mathbf{s}$ is globally consistent wrt G and so $\mathbf{r}$ is globally consistent wrt Σ. Therefore $\mathbf{R}$ is independent wrt Σ. □

Theorem 4.1. *Let Σ and $\mathbf{R}$ satisfy conditions (a), (c). Then $\mathbf{R}$ is independent wrt Σ if and only if $\mathbf{R}$ satisfies the maximality and uniqueness condition wrt Σ.*

Proof. Follows from lemmata 4.2, 4.3, 4.4. □

5. Independence with respect to FDs and Unary INDs

In this section we make different assumptions on Σ: we put no restrictions on F, while requiring the INDs in I to be unary. In this case ([13]) the implication problem is decidable, and so the set of FDs implied by Σ can be computed. Using the same notations as in Section 3, given a database scheme $\mathbf{R} = \{R_1(X_1), ..., R_m(X_m)\}$ and the set of FDs and unary INDs $\Sigma = F \cup I$, for every $R_i \in \mathbf{R}$, G_i is a cover of the FDs in Σ^+ with tag R_i, and $G = \bigcup_i G_i$.

In order to prove our main results, we introduce an algorithm, essentially a modification of an algorithm presented by [13], and prove some properties about it. The algorithm operates on a (multi-)graph $\mathcal{G}(\Sigma)$ with colored edges, defined from $\mathbf{R}$ and Σ as follows. The nodes of $\mathcal{G}(\Sigma)$ correspond to the attributes in Σ tagged with relation names (e.g., $R_i.A_j$). $\mathcal{G}(\Sigma)$ has *red* edges, corresponding to FDs, and *black* edges, corresponding to INDs. There is a red edge from $R_i.A$ to $R_i.B$ if and only if $R_i : A \rightarrow B \in \Sigma$ and a black edge from $R_i.A$ to $R_j.B$ if and only if $R_i[A] \supseteq R_j[B] \in \Sigma$. If, for some pair of nodes n_1, n_2, the graph contains both the red (black) edge (n_1, n_2) and the red (black) edge (n_2, n_1), then we replace them with an undirected edge $\{n_1, n_2\}$ of the same color.

If we assume, without loss of generality, that Σ is closed wrt implication, the graph has the properties in lemma 2 of [13]:

(a) Nodes have red (black) self-loops, and the red (black) subgraph of $\mathcal{G}(\Sigma)$ is transitively closed.
(b) The subgraphs induced by the strongly connected components of $\mathcal{G}(\Sigma)$ contain only undirected edges.
(c) In each strongly connected component, the red (black) subset of edges forms a collection of node disjoint cliques (the red and black partitions of nodes may be different).
(d) If $A_1A_2...A_m \rightarrow B$ is an FD in Σ, and $A_1A_2...A_m$ have the common ancestor A in the red subgraph of $\mathcal{G}(\Sigma)$, then $\mathcal{G}(\Sigma)$ contains a red arc from A to B.

It is possible to consider a topological sort of the directed acyclic graph (*dag*) of the strongly connected components (*scc*'s) of $\mathcal{G}(\Sigma)$ and to assign to each scc a unique *scc-number*, smaller than the scc-number of all its descendant components in the dag. Let s be the number of scc's in $\mathcal{G}(\Sigma)$; for every $j \in \{1, ..., s\}$, scc(j) indicates the scc whose scc-number is j.

The algorithm we define takes as input a database state $\mathbf{r} = \{r_1, ..., r_n\}$ and adds to the various relations some tuples, in order to obtain another state that satisfies some interesting conditions. Assuming the domains contain the non-negative integers and the special symbol x (any other countable set could be used as well), the algorithm uses the following conventions:

- For each attribute $R.A$, amax($R.A$) indicates the maximum integer value in $\pi_A(r)$.
- For each red clique $\mathcal{C}$, cmax($\mathcal{C}$) indicates the maximum integer value among those assumed by the attributes in the clique; so,

$$\mathrm{cmax}(\mathcal{C}) = \max_{R.A \in \mathcal{C}} \{\mathrm{amax}(R.A)\}$$

- For each $j \in \{1, ..., s\}$, smax(j) indicates the maximum value among those assumed by the attributes in scc(j); so,

$$\mathrm{smax}(j) = \max_{R.A \in \mathrm{scc}(j)} \{\mathrm{amax}(R.A)\}$$

- For each red clique $\mathcal{C}$, desc($\mathcal{C}$) indicates the set of the attributes that are in $\mathcal{C}$ or in cliques that are red descendant of $\mathcal{C}$ in the dag.
- The algorithm adds tuples to the relations in the input state. Whenever it is said that the tuple t under consideration is padded with "new values" for the attributes in a set X, this means that for each attribute $A \in X$, $t.A = \mathrm{amax}(A) + 1$.

The database state input to the algorithm is required to satisfy the following conditions:

(i1) For every relation r_i, for every attribute A_j in its scheme, the only repeated value (if any) is x, and the other values form an initial subset of the positive integers (i.e., $\{1, 2, ..., \mathrm{amax}(R_i.A_j)\}$). Also, no violation of any IND is caused by x (i.e., if $R_i[A_j] \supseteq R_h[A_k]$ is in Σ and $x \in \pi_{A_k}(r_h)$, then $x \in \pi_{A_j}(r_i)$).
(i2) For every red clique $\mathcal{C}$ in $\mathcal{G}(\Sigma)$ (which is embedded in some relation scheme, say R_j), and for every pair $R_j.A_h, R_j.A_k \in \mathcal{C}$, we have $\mathrm{amax}(R_j.A_h) = \mathrm{amax}(R_j.A_k)$.
(i3) The state satisfies the FDs in G.

Given an input state that satisfies the above conditions, the algorithm in Figure 5.1 will produce a state that satisfies $\Sigma = F \cup I$. The correctness of this claim follows from the lemma below.

input: *a database state* $\mathbf{r}$ *over a scheme* $\mathbf{R}$ *with dependencies* Σ; $\mathbf{r}$ *satisfies conditions (i1), (i2), (i3);* s *is the number of scc's in* $G(\Sigma)$.
output: *a database state containing the input state and locally consistent wrt* Σ.

```
begin
  add a row of 0's to each relation
  for j := 1 to s do begin
    let C be the set of red cliques in scc(j)
    for each C ∈ C do begin
      let n_C = cmax(C) − min_{C'∈C} cmax(C') + 1
    end
    for each C ∈ C do
      for each r_l ∈ r do begin
        add n_C tuples to r_l
        with 0's for the attributes in desc(C)
        and new values elsewhere.
      end
    if j > 1 and smax(j) ≥ smax(j − 1) then begin
      let p = smax(j) − smax(j − 1) + 1
      for each C ∈ C do begin
        add p tuples to each relation
        with 0's for the attributes in desc(C)
        and new values elsewhere
      end
    end
  end
end
```

Figure 5.1.

Lemma 5.1. *After* j *executions of the outer loop, the following properties hold, for every* $j \geq 0$ ($j = 0$ *means before the first execution).*

(m1) *For every relation r_i, for every attribute A_l in its scheme, the only repeated values (if any) are $0, x$, and the other values form an initial subset of the positive integers $(1, 2, ..., \text{amax}(R_i.A_l))$. Also, $0 \in \pi_{A_l}(r_i)$ (and so no violation of any IND is caused by 0), and no violation of any IND is caused by x (i.e., if $R_h[A_k] \supseteq R_i[A_l] \in \Sigma$ and $x \in \pi_{A_l}(r_i)$, then $x \in \pi_{A_k}(r_h)$).*

(m2) *For every red clique $\mathcal{C}$ in $\mathcal{G}(\Sigma)$ (embedded in some relation scheme, say R_j), and for every pair $R_j.A_h, R_j.A_k \in \mathcal{C}$, we have $\text{amax}(R_j.A_h) = \text{amax}(R_j.A_k)$.*

(m3) *The state satisfies the FDs in G.*

(m4) *For every $i \leq j$, if $R.A, S.B \in \text{scc}(i)$, then $\text{amax}(R.A) = \text{amax}(S.B)$*

(m5) *For every $k < i \leq j$, if $R.A \in \text{scc}(k)$, $S.B \in \text{scc}(i)$, then $\text{amax}(R.A) > \text{amax}(S.B)$*

Proof. We omit this proof, which is carried out by carefully considering the various steps in the algorithm. □

Corollary 5.1. *The state produced by the algorithm satisfies the dependencies in $F \cup I$, provided that the input state satisfies the required conditions.*

Lemma 5.2. *Given a database scheme* **R**, *with the constraints $\Sigma = F \cup I$, and G defined as usual, if* **R** *is not independent wrt G, then there is a database state* **r** *such that*

(a) **r** *locally satisfies G,*

(b) **r** *is not globally consistent wrt G, and*

(c) **r** *satisfies the conditions required to be a legal input to the algorithm.*

Proof. Since **R** is not independent wrt G we know that there are states satisfying conditions (a), (b). One of them is the counterexample state shown by [20]. This state satisfies condition (i3) for being an input to algorithm (since it locally satisfies G), but need not satisfy conditions (i1) and (i2). However, this state can be transformed, by adding a suitable set of tuples to each relation, into a state that satisfies the required conditions. Again, we omit the details. □

Theorem 5.1. **R** *is independent wrt G if and only if* **R** *is independent wrt $\Sigma = F \cup I$.*

Proof. The *only if* part holds by lemma 3.1, so we concentrate on the *if* part. We proceed by showing that if **R** is not independent wrt G, then **R** is not independent wrt Σ either.

If **R** is not independent wrt G, then we know, by lemma 5.2, that there is a state **r** that locally satisfies G, is globally inconsistent wrt G, and can be input to our algorithm. Then, by corollary 5.1, the state output by the algorithm satisfies Σ. Since the algorithm only adds tuples, without modifying the existing ones, the output state is still globally inconsistent wrt G, and so wrt Σ: therefore it is a counterexample state that demonstrates that **R** is not independent wrt Σ. □

6. Conclusions

Functional and inclusion dependencies have been regarded as the two most common kinds of constraints in a relational database. In view of the desirability of independent schemes, we proposed and defined the concept of independent schemes in the presence of functional and inclusion dependencies. Since the interaction between functional and inclusion dependencies is difficult to understand in general ([7], [17]), we restricted our attention to key-based dependencies ([12]) and to functional and unary inclusion dependencies ([13]). In each case, a characterization of independence was obtained. This work brought insight into the design of independent schemes under these assumptions. For the case of functional and unary inclusion dependencies, we also derived an algorithm which converts a state that satisfies certain conditions to a state that satisfies both the functional and inclusion dependencies. This algorithm might be proven to be useful in query optimization and schema analysis when functional and unary inclusion dependencies are assumed. In this paper, we only considered certain restricted classes of functional and inclusion dependencies. There are other classes of functional and inclusion dependencies which have decidable solution for the implication problem. One such example is the class of functional and acyclic inclusion dependencies ([4], [8]). It will be interesting to investigate schema properties under such an assumption.

Acknowledgement

The authors would like to thank P. Kanellakis for providing them with an advance copy of the full version of [13].

7. References

1. Atzeni, P., Chan, E.P.F., "Efficient Query Answering in the Representative Instance Approach," *Proc. Fourth ACM SIGACT SIGMOD Symp. on Principles of Database Systems, 1985*, pp. 181-188.
2. Atzeni, P., De Bernardis, M.C., "A New Basis for the Weak Instance Model," *Proc. Sixth ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems, 1987*, pp. 79-86.
3. Beeri, C., Korth, H.F., "Compatible Attributes in a Universal Relation," *Proc. ACM SIGACT SIGMOD Symp. on Principles of Database Systems, 1982*, pp. 55-62.

4. Casanova, M.A., Fagin, R., Papadimitriou, C.H., "Inclusion Dependencies and their interaction with Functional Dependencies," *Journal of Comp. and System Sc.* **28**, 1984, pp. 29-59.
5. Casanova, M.A., Vidal, V.M.P., "Towards a Sound Integration Methodology," *Proc. Second ACM SIGACT SIGMOD Symp. on Principles of Database Systems, 1983*, pp. 39-48.
6. Chan, E.P.F., Mendelzon, A.O., "Answering Queries on Embedded-complete Database Schemes," *Journal of the ACM*, 1987, to appear.
7. Chandra A., Vardi, M.Y., "The Implication Problem for Functional and Inclusion Dependencies is Undecidable," *SIAM J. on Computing* **14**, 1985, pp. 671-677.
8. Cosmadakis, S.S., Kanellakis, P.C., "Functional and Inclusion Dependencies - A Graph Theoretical Approach," *Advances in Computing Research* **3**, 1986, JAI Press,pp. 163-184.
9. Graham, M.H., Yannakakis, M., "Independent Database Schemas," *Journal of Comp. and System Sc.* **28**, 1984, pp. 121-141.
10. Honeyman, P., "Testing Satisfaction of Functional Dependencies," *Journal of the ACM* **29**, 1982, pp. 668-677.
11. Ito, M., Iwasaki, M., Kasami, T., "Some Results on the Representative Instance in Relational Databases," *SIAM J. on Computing* **14**, 1985, pp. 334-354.
12. Johnson, D.S., Klug, A., "Testing Containment of Conjunctive Queries under Functional and Inclusion Dependencies," *Journal of Comp. and System Sc.* **28**, 1984, pp. 167-189.
13. Kanellakis, P.C., Cosmadakis, S.S., Vardi, M., "Unary Inclusion Dependencies have Polynomial time Inference Problems," *Proc. Seventeenth ACM SIGACT Symp. on Theory of Computing, 1983*, pp. 264-277.
14. Maier, D., *The Theory of Relational Databases*, Computer Science Press, 1983.
15. Maier, D., Rozenshtein, D., Warren, D.S., "Windows Functions," *Advances in Computing Research* **3**, 1986, JAI Press,pp. 213-246.
16. Maier, D., Ullman, J.D., Vardi, M., "On the Foundations of the Universal Relation Model," *ACM Trans. on Database Syst.* **9**, 1984, pp. 283-308.
17. Mitchell, J.C., "The Implication Problem for Functional and Inclusion Dependencies," *Information and Control*, **56**, 1983, pp. 154-173.
18. Sagiv, Y., "Can We Use the Universal Instance Assumption without Using Nulls?", *Proc. ACM SIGMOD International Conf. on Management of Data, 1981*, pp. 108-120.
19. Sagiv, Y., "A Characterization of Globally Consistent Database and Their Correct Access Paths," *ACM Trans. on Database Syst.* **8**, 1983, pp. 266-286.
20. Sagiv, Y., "Evaluation of Queries in Independent Database Schemes," unpublished manuscript, 1984.
21. Sciore, E., "Improving Database Schemes by Adding Attributes," *Proc. Second ACM SIGACT SIGMOD Symp. on Principles of Database Systems, 1983*, pp. 379-382.
22. Ullman, J.D., *Principles of Database Systems (2nd Ed.)*, Computer Science Press, 1982.