Maintaining Consistency of Imprecise Relations *

Mark Levene
Department of Computer Science
University College London
Gower Street
London WC1E 6BT, U.K.
Email: mlevene@cs.ucl.ac.uk

Abstract

We extend functional dependencies (FDs), which are the most fundamental integrity constraints that arise in practice in relational databases, to be satisfied in an imprecise relation. The problem we tackle is the following: given an imprecise relation r over a relation schema R and a set of FDs F over R, what is the most precise approximation of r, which is also consistent with respect to F.

We formalise the notion of an imprecise relation by defining tuple values to be sets of values rather than just single values as is the case when the information is precise. We interpret each value in such a set as being equally likely to be the true value. This gives rise to equivalence classes of equally likely values thus allowing us to define the *merge* of an imprecise relation r which replaces values in r by their equivalence class. We also define a partial order on merged imprecise relations leading to the notion of an imprecise relation being less precise than another imprecise relation. This partial order induces a lattice on the set of merged imprecise relations.

An imprecise relation is consistent with respect to a set of FDs F if it satisfies F. Satisfaction of an FD in an imprecise relation is defined in terms values being equally likely rather than equal. We show that Armstrong's axiom system is sound and complete for FDs being satisfied in imprecise relations. We redefine the chase procedure for an imprecise relation r over R as a means of maintaining consistency of r with respect to F. Our main result is that the output $chase(r, \mathrm{F})$ of the chase procedure is the most precise approximation of r with respect to F in the following sense. It is shown to be the join of all consistent imprecise relations s such that s is a merged imprecise relation that is

* Appeared in *Computer Journal* **39**, (1996), 114-123

less precise than r. It is also shown that $chase(r, \mathrm{F})$ can be computed in polynomial time in the sizes of r and F.

Keywords: Information storage and retrieval; imprecise relation; functional dependency.

1 Introduction

Functional dependencies (FDs) are the most fundamental integrity constraints that arise in practice in relational databases [26, 3]. Moreover, the problem of maintaining the consistency of a relational database is well-known [14, 11]. We introduce the notion of *imprecise relations* and FDs being satisfied in imprecise relations in order to cater for the situation when the information may be obtained from different sources and therefore may be imprecise. Components of tuples in imprecise relations are nonempty sets of values rather than just single values as is the case when the information is precise. In the absence of further information about the true value, we interpret each value in a set of values as being equally likely to be the true value. Thus each value in an imprecise relation induces an equivalence class of equally likely values, which we call an *imprecise set.*

Our model for imprecise relations builds upon the equivalence classes model of Shenoi and Melton [23, 24] for fuzzy relational databases, which extended the earlier fuzzy relational model of [6]. Both of these models use the notion of fuzzy ordering [28], where two values are considered to be equivalent if they are indistinguishable at some specified level of precision $\alpha \in [0, 1]$. The main difference between our model and those of [6, 23, 24] is that in our model an imprecise set is dynamic and determined solely by the current state of an imprecise relation, rather than being a subset of a fuzzy set at some level of precision.

We formalise the notion of an FD being satisfied in an imprecise relation (cf. [23, 24]; see [5] for a discussion of various approaches to defining FDs in the context of fuzzy relations). Our notion of FD reduces to the standard notion of FD [26, 3] when relations are precise. We are able to show that Armstrong's axiom system [2, 26, 3] is sound and complete for FDs being satisfied in imprecise relations. An imprecise relation is said to be *consistent* with respect to a set of FDs F if it satisfies F. Given an imprecise relation r over a relation schema R and a set of FDs F over R, we call the problem of finding the most precise approximation of r, which is also consistent with respect to F, the *consistency problem.* This is the main problem which we tackle herein.

As a motivating example consider a relation schema R $= \{N, P\}$, where N stands for a share name and P stands for the price of the share. In addition, suppose that the FD $N \rightarrow P$ is specified as a constraint, implying that each share has a unique price. We assume an imprecise relation over R, where the current price information about shares may be obtained from different sources. Thus at any given time the information we may have at our disposal may be inconsistent. Let r_1 be the imprecise relation over R, shown in Table 1. It can easily be verified that r_1 satisfies $N \rightarrow P$ and is thus consistent. Suppose that at some later stage a tuple was inserted into r_1, obtaining the imprecise relation r_2 shown in Table 2. It can easily be verified that r_2 does not satisfy $N \rightarrow P$ and is thus inconsistent. The imprecise relation r_2 can be approximated by the less precise relation r_3, shown in Table 3. It can easily be verified that r_3 satisfies $N \rightarrow P$ and is thus consistent. The imprecise relation r_3 is in fact the most precise approximation of r_2. The scenario of our example also fits in well with that of a *mobile* environment, where data is replicated across several mobile units and each mobile unit may acquire up to date information from different sources [4].

N	P
0	0

Table 1: The relation r_1

N	P
0	0
0	1

Table 2: The relation r_2

N	P
0	$\{0, 1\}$

Table 3: The relation r_3

We now outline the solution to the consistency problem. We define the *merge* of an imprecise relation r as the operation which replaces values in r by their equivalence class of equally likely values. This leads us to define a partial order on merged imprecise relations and the notion of an imprecise relation being less precise than another imprecise relation. This partial order induces a lattice on the set of merged imprecise relations, which we denote by MERGE(R),

Assuming that an imprecise relation r is updated with information obtained from several sources, it may be the case that at any given time the relation r is inconsistent (i.e. it is *not* consistent) with respect to a set of FDs F. Thus we need an algorithmic procedure which transforms an inconsistent imprecise relation into its most precise consistent approximation. The chase procedure [21, 14, 10, 18, 19] is a fundamental theorem proving tool in relational database theory. In the context of imprecise relations we redefine the chase procedure for an imprecise relation r over R as a means of maintaining consistency of r with respect to F. We investigate the properties of the chase procedure showing amongst other results that it outputs a consistent imprecise relation, it is unique and can be computed in polynomial time in the sizes of r and F, and it commutes with the *merge* operation.

In order to show that the chase procedure solves the consistency problem we give a semantic characterisation of the chase procedure by showing that in some accurate sense $chase(r, \mathrm{F})$ is the most precise approximation of r with respect to F. We first define the *join* of imprecise relations, which corresponds to the *least upper bound* [12] of these relations in the lattice MERGE(R). The join operation can also be viewed as a generalisation of the *intersection* operation of the relational algebra (cf. [7]), which is a special case of the natural join operation [26, 3]. We then define the *most precise approximation* of r with respect to F to be the join of all the consistent and merged imprecise relations which are less precise than r. Our main result is that $chase(r, \mathrm{F})$ is exactly the most precise approximation of r with respect to F.

We mention several other approaches in dealing with incomplete and imprecise information in relational databases. The standard way of dealing with incomplete information is by using the so called *null values*. The most common interpretation of a null value is "value at present unknown" [8, 10, 19] but other interpretations are "value does not exist", "no information" [29] and that of a *generic null value* [18]. When the null value of type "value is unknown" ranges over a finite set of values, one of which is the true value, we have an *or-set* [15] (cf. [20]). As was shown in [27] it is, in general, NP-complete to check whether a relation containing or-sets is consistent.

We have already mentioned that fuzzy set theory [28] can be used to to deal with uncertain and imprecise information [6, 23, 24]. A different approach is that of using rough sets [22] in order to approximate information in relational databases [16]. Finally, we mention that probability theory can also be used as a foundation for dealing with uncertain and imprecise knowledge in relational databases [17].

Definition 1.1 (Notation) We denote the cardinality of a set S by $|\mathrm{S}|$. The *size* of a set S is defined to be the cardinality of a standard encoding [9] of S.

If S is a subset of T we write $\mathrm{S} \subseteq \mathrm{T}$ and if S is a proper subset of T we write $\mathrm{S} \subset \mathrm{T}$. We often denote the singleton $\{\mathrm{A}\}$ simply by A. The nonempty powerset of a set S is denoted by $\mathcal{P}(\mathrm{S})$. We will make use of the index set $I = \{1, 2, \ldots, n\}$.

The rest of the paper is organised as follows. In Section 2 we we formalise the notion of imprecise relations. In Section 3 we define the notion of a functional dependency being satisfied in an imprecise relation. In Section 4 we redefine the chase procedure in the context of imprecise relations as a means for maintaining consistency of imprecise relations. In Section 5 we give a semantic characterisation of the chase procedure by showing that in some accurate sense it is the most precise approximation of an imprecise relation, which is also consistent with respect to a set of FDs. Finally, in Section 6 we give our concluding remarks.

2 Imprecise Relations

Herein we formalise the notion of an imprecise relation, which allows components of tuples to be nonempty sets of values rather than just a single value as is the case when the information is precise. We interpret each value in a set of values as being equally likely to be the true value. Each value in an imprecise relation induces an equivalence class of equally likely values, which we call an *imprecise set*. The motivation for our interpretation of an imprecise set is that when the information in an imprecise relation is obtained from several different sources, in the absence of further information about the true value, it is reasonable to assume that each value is equally likely.

Our model for imprecise relations builds upon the equivalence classes model for fuzzy relations of [23, 24], which extended the earlier fuzzy relational model of [6]. The main difference between our model and those of [6, 23, 24] is that our model an imprecise set is dynamic and determined solely by the current state of an imprecise relation, rather than being a fuzzy set at some level of precision.

Definition 2.1 (Relation schema and imprecise relation) Let $\mathcal{U}$ is a countable set of attributes and $\mathcal{D}$ be a finite set of domain values.

A *relation schema* R is a finite set of attributes in $\mathcal{U}$. A *imprecise tuple* t over R with respect to $\mathcal{D}$ (or simply a tuple t when R and $\mathcal{D}$ are understood from context) is a total mapping from R into $\mathcal{P}(\mathcal{D})$ such that $\forall A \in \mathrm{R}$, $t(A) \in \mathcal{P}(\mathcal{D})$.

An *imprecise relation* over R with respect to $\mathcal{D}$ (or simply a relation when R and $\mathcal{D}$ are understood from context) is a finite (possibly empty) set of imprecise tuples over R. We let $\mathrm{REL}_{\mathcal{D}}(\mathrm{R})$ (or simply REL(R) whenever $\mathcal{D}$ is understood from context) denote the finite set of all relations over R with respect to $\mathcal{D}$.

Example 2.1 Let $\mathrm{R} = \{N, P, B\}$ be a relation schema, where the semantics of N and P are as in the motivating example which was given in the introduction and B stands for the name of the broker dealing with shares that have name P. An imprecise relation r over R is shown in Table 4.

N	P	B
{0}	{0,1}	{0,1}
{0}	{1,2}	{0,2}
{1,2}	{2}	{0}
{3,4}	{3}	{3}

Table 4: The relation r

Definition 2.2 (Projection) The *projection* of a tuple t over R onto a set of attributes $Y \subseteq \mathrm{R}$, denoted by $t[Y]$, is the restriction of t to Y; we also call $t[Y]$ the Y-value of t.

The projection of a relation r over R onto Y, denoted $\pi_Y(r)$, is defined by $\pi_Y(r) = \{t[Y] \mid t \in r\}$.

The semantics of a set $t[A]$, where t is a tuple over R and $A \in \mathrm{R}$, are that the values $v \in t[A]$ are all equally likely to be the true value. Thus when a set $t[A]$ has more than one value this implies that information is imprecise. Observe that $t[A]$ cannot be empty, since we excluded the empty set from the powerset operation.

Definition 2.3 (Precise and imprecise relations) A tuple t over R is *precise* if $\forall A \in \mathrm{R}$, $|t[A]| = 1$, otherwise (when $|t[A]| > 1$) t is said to be *imprecise*.

A relation r over R is said to be *precise* if $\forall t \in r, t$ is precise, otherwise r is said to be *imprecise*.

From now on we let R *be a relation schema and* r *be a relation over* R. As usual uppercase letters from the end of the alphabet such as X, Y will be used to denote sets of attributes in R, while those from the beginning of the alphabet such as A, B will be used to denote single attributes in R.

The *active domain* of r with respect to A is the set of all A-values of tuples in r.

Definition 2.4 (Active domain) The *active domain* of r with respect to A, denoted by $adom(r, A)$ is defined by

$$adom(r, A) = \bigcup \{t[A] \mid t \in r\}.$$

The *basic information* of a relation r is the precise relation which is equivalent to the Cartesian product of all the active domains of r with respect to the attributes $A \in$ R.

Definition 2.5 (Basic information) The *basic information* of r, denoted by $inf(r)$, is the relation over R defined by

$$inf(r) = \{t \mid \forall A \in \mathrm{R}, t[A] = \{v\} \text{ for some } v \in adom(r, A)\}.$$

We next define when values are equally likely and introduce the notion of an *imprecise set* which is an equivalence class of equally likely values.

Definition 2.6 (Equally likely values) Two values $v_1, v_2 \in adom(r, A)$ are *equally likely* with respect to r and A, written $v_1 \approx_{(r,A)} v_2$ (or simply equally likely, written $v_1 \approx v_2$, whenever r and A are understood from context) if

either $\exists t \in r$ such that $v_1, v_2 \in t[A]$ or $\exists t_1, t_2, \ldots, t_n \in r, n \geq 2,$ such that $\forall i \in I - \{n\}, t_i[A] \cap t_{i+1}[A] \neq \emptyset$ and $v_1, v_2 \in \bigcup_{i \in I} t_i[A]$.

We call an equivalence class of a value $v \in adom(r, A)$ with respect to $\approx$ the *imprecise set* over A with respect to r that is induced by v (or simply the imprecise set induced by v whenever A and r are understood from context).

In general, we refer to an imprecise set over A with respect to r that is induced by some v, as an imprecise set over A with respect to r (or simply an imprecise set).

Example 2.2 With respect to the relation r over R shown in Table 4 we have the imprecise sets {0}, {1,2} and {3,4} over N, {0,1,2} and {3} over P, and {0,1,2} and {3} over B.

The next proposition follows from the fact that the set of equivalence classes is a partition of $adom(r, A)$ (see [13] for more detail on equivalence classes).

Proposition 2.1 The family of imprecise sets over A with respect to r are pairwise disjoint. $\square$

Theorem 2.2 The family of imprecise sets over A with respect to r can be computed in polynomial time in the size of r.

Proof. Construct a directed graph G = (V, E), where the vertices $w \in$ V correspond to the sets in $\pi_A(r)$ and an arc (w_1, w_2) is in E if and only if $w_1 \cap w_2 \neq \emptyset$. Let $G^+ = (V^+, E^+)$ be the reflexive and transitive closure of G, which can be computed in polynomial time in the cardinality of V [1]. It follows that G^+ can also be computed in polynomial time in the size of r, since both $\pi_A(r)$ and intersection of sets are polynomial time operations.

The result now follows, since c is an imprecise set over A with respect to r that is induced by some $v \in adom(r, A)$ if and only if

$$\text{for some } w \in V^+, v \in w \text{ and } c = \bigcup\{w' \mid (w, w') \in E^+\}. \quad \square$$

We next define the *merge* operation which replaces each A-value of a tuple in a relation by the imprecise set induced by its component values (cf. [23, 24]).

Definition 2.7 (Merge operation) The *merge* of r, denoted by $merge(r)$, is the relation resulting from replacing each A-value $t[A]$ of t, where $A \in$ R, $t \in r$ and $v \in t[A]$ by the imprecise set over A with respect to r that is induced by v.

We let $\text{MERGE}_{\mathcal{D}}(\text{R})$ (or simply MERGE(R) whenever $\mathcal{D}$ is understood from context) denote the finite set, $\{r \in \text{REL}_{\mathcal{D}}(\text{R}) \mid r = merge(r)\}$. If $r \in$ MERGE(R) then we say that r is a *merged* relation.

N	P	B
{0}	{0,1,2}	{0,1,2}
{1,2}	{0,1,2}	{0,1,2}
{3,4}	{3}	{3}

Table 5: The relation $merge(r)$

Example 2.3 The imprecise relation $merge(r)$, where r is the relation over R shown in Table 4, is shown in Table 5. We note that the first two tuples in r have been merged into a single tuple in $merge(r)$.

It can be verified that *merge* preserves the basic information of r, i.e. $inf(merge(r)) = inf(r)$. The next proposition follows directly from Definitions 2.6 and 2.7.

Proposition 2.3 For each $A \in$ R, the family of imprecise sets over R with respect to r is equal to the family of sets $\pi_A(merge(r))$. $\Box$

The next corollary follows from Definition 2.7 and Theorem 2.2.

Corollary 2.4 The relation $merge(r)$ can be computed in polynomial time in the size of r. $\Box$

We next define a partial order on sets of values and extend this partial order to tuples and relations in MERGE(R). Intuitively, the smaller the cardinality of an A-value of a tuple, the *finer* (or *more precise*) that A-value is, i.e. there are less equally likely A-values to choose from. Correspondingly, the larger the cardinality of an A-value of a tuple, the *coarser* (or *less precise*) that A-value is, i.e. there are more equally likely A-values to choose from. For example, if we have two possible prices for a given share then we have more precise information than if we had three possible prices for that share.

Definition 2.8 (Less precise sets of values, tuples and relations) We define a partial order in $\mathcal{P}(\mathcal{D})$, denoted by $\sqsubseteq$, as follows, where $c_1, c_2 \in \mathcal{P}(\mathcal{D})$:

$$c_1 \sqsubseteq c_2 \text{ if } c_2 \subseteq c_1.$$

If $c_1 \sqsubseteq c_2$ we say that c_1 is *coarser* (or *less precise*) than c_2; alternatively we say that c_2 is *finer* (or *more precise*) than c_1.

We extend $\sqsubseteq$ to be a partial order in the set of tuples over R as follows, where t_1 and t_2 are tuples over R:

$$t_1 \sqsubseteq t_2 \text{ if } \forall A \in \mathrm{R}, t_1[A] \sqsubseteq t_2[A];$$

we say that t_1 is *less precise* than t_2, or alternatively we say that t_2 is *more precise* than t_1.

Finally, we extend $\sqsubseteq$ to be a partial order in MERGE(R) as follows, where $r_1, r_2 \in$ MERGE(R):

$$r_1 \sqsubseteq r_2 \text{ if } \forall t_2 \in r_2, \exists t_1 \in r_1 \text{ such that } t_1 \sqsubseteq t_2;$$

we say that r_1 is *less precise* then r_2, or alternatively r_2 is *more precise* than r_1.

We observe that when extending $\sqsubseteq$ to relations, if we relax the condition $r \in$ MERGE(R) to $r \in$ REL(R), then $\sqsubseteq$ is a preorder and not a partial order, since the antisymmetry property of partial orders is lost. For example, let $r_1 = \{\langle\{0,1\}\rangle\}$ and let $r_2 = \{\langle\{0\}\rangle, \langle\{0,1\}\rangle\}$. It can easily be verified that $r_1 \sqsubseteq r_2$ and that $r_2 \sqsubseteq r_1$ but $r_1 \neq r_2$. It can also be verified that $merge(r_2) = r_1$.

The reason $\sqsubseteq$ is a partial order in MERGE(R) is that due to Propositions 2.1 and 2.3 the tuples in each relation $r \in$ MERGE(R) form an *antichain* [12] with respect to $\sqsubseteq$. That is, for all distinct tuples $t_1, t_2 \in r$, $t_1 \not\sqsubseteq t_2$ and $t_2 \not\sqsubseteq t_1$.

We further observe that the partial order $\sqsubseteq$ in MERGE(R) is the well-known *Smyth* power domain ordering [25]. (For a thorough investigation of the application of power domain orderings in database theory see [7].)

From Definition 2.8 we can deduce that MERGE(R) is a *lattice* with respect to $\sqsubseteq$ [12], whose top element is $\emptyset$, i.e. the relation with an empty set of tuples, and whose bottom element is the singleton $\{\langle\mathcal{D}, \ldots, \mathcal{D}\rangle\}$ conveying *no information*.

We next extend $\approx_{(r,A)}$ to sets of values.

Definition 2.9 (Equally likely sets of values) Two sets of values $c_1, c_2 \in \mathcal{P}(adom(r, A))$ are *equally likely* written $c_1 \approx_{(r,A)} c_2$ (or simply $c_1 \approx c_2$, whenever r and A are understood from context) if

$$\exists v_1 \in c_1, \exists v_2 \in c_2 \text{ such that } v_1 \approx v_2.$$

We further extend $\approx_{(r,A)}$ to tuples $t_1, t_2 \in r$, as follows: $t_1 \approx_r t_2$ (or simply $t_1 \approx t_2$, whenever r is understood from context) if $\forall A \in \mathrm{R}$, $t_1[A] \approx_{(r,A)} t_2[A]$.

It follows that if $t_1 \approx_r t_2$ then for each $A \in \mathrm{R}$, the A-values of t_1 and t_2 belong to the same equivalence class with respect to $\approx_{(r,A)}$ and are thus equally likely.

Example 2.4 With respect to the relation r over R shown in Table 4 the reader can verify that $t_1 \approx_r t_2$, where t_1 and t_2 are the first and second tuples in r.

The next proposition is a direct consequence of Definitions 2.7 and 2.9.

Proposition 2.5 $t_1 \approx_r t_2$ if and only if there exists a unique tuple $t \in merge(r)$ such that $t_1 \approx_s t$ and $t_2 \approx_s t$, where $s = \{t, t_1, t_2\}$. $\square$

In terms of [23, 24] t_1 and t_2 above are redundant tuples that can be replaced by the nonredundant tuple t. Thus if two tuples are indistinguishable at some level of precision, or using our terminology are equally likely, they can be merged together.

3 Functional Dependencies in Imprecise Relations

Herein we formalise the notion of an FD being satisfied in an imprecise relation (cf. [23, 24]; see [5] for a discussion of various approaches to defining FDs in the context of fuzzy relations). Our notion of FD reduces to the standard notion of FD [26, 3] when relations are precise. We are able to show that Armstrong's axiom system [2, 26, 3] is sound and complete for FDs being satisfied in imprecise relations.

Definition 3.1 (Functional dependency) A *functional dependency* over R (or simply an FD) is a statement of the form $X \rightarrow Y$, where $X, Y \subseteq \mathrm{R}$.

From now on we let F *be a set of FDs over* R and $X \rightarrow Y$ *be a single FD over* R. We define the *size* of F to be the number of attributes appearing in F including repetitions.

Definition 3.2 (Satisfaction of an FD) An FD $X \to Y$ is satisfied in a relation r, denoted by r $\models_\approx X \to Y$, if $\forall t_1, t_2 \in r$, if $t_1[X] \approx t_2[X]$ then $t_1[Y] \approx t_2[Y]$.

A set of FDs F over R is satisfied in r, denoted by $r \models_\approx$ F, if $\forall X \to Y \in$ F, $r \models_\approx X \to Y$.

If $r \models_\approx$ F we say that r is *consistent* with respect to F (or simply r is consistent if F is understood from context); otherwise if $r \not\models_\approx$ F then we say that r is *inconsistent* with respect to F (or simply r is inconsistent).

We let $\text{SAT}_{\mathcal{D}}$(F) (or simply SAT(F) whenever $\mathcal{D}$ is understood from context) denote the finite set $\{r \in \text{MERGE}_{\mathcal{D}}(\text{R}) \mid r \models_\approx \text{F}\}$.

Example 3.1 Let F $= \{N \to PB, B \to N\}$ be a set of FDs over R, where R is the relation schema whose semantics where given in Example 2.1. The reader can verify that $r \models_\approx N \to PB$ but that $r \not\models_\approx B \to N$, where r is the relation over R shown in Table 4. Thus $r \in \text{SAT}(\{N \to PB\})$ but $r \notin$ SAT(F). Equivalently, if we replace r above by $merge(r)$, which is shown in Table 5, we have $merge(r) \in \text{SAT}(\{N \to PB\})$ but $merge(r) \notin$ SAT(F).

We note that the definition of satisfaction of an FD in a relation reduces to the standard definition of the satisfaction of an FD when the relation is precise [26, 3]. The next proposition is immediate from Definition 3.2 and Propositions 2.1 and 2.3.

Proposition 3.1 The following two statements are true:

1. If $r \in$ MERGE(R), then $r \models_\approx X \to Y$ if and only if $\forall t_1, t_2 \in r$, if $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$.
2. $r \models_\approx X \to Y$ if and only if $merge(r) \models_\approx X \to Y$. $\square$

Definition 3.3 (Logical implication) We say that F *logically implies* an FD $X \to Y$ over R written F $\models_\approx X \to Y$, whenever for any domain $\mathcal{D}$, $\forall r \in \text{REL}_{\mathcal{D}}(\text{R})$, the following condition is true:

$$\text{if } r \models_\approx \text{F holds then } r \models_\approx X \to Y \text{ also holds.}$$

For a set of FDs G over R, we say that F logically implies G if written F $\models_\approx$ G, if $\forall X \to Y \in$ G, F $\models_\approx X \to Y$. We say that G is a *cover* of F if F $\models_\approx$ G and G $\models_\approx$ F.

We assume that the reader is familiar with Armstrong's axiom system [2, 26, 3] comprising the three inference rules reflexivity, augmentation and transitivity.

Definition 3.4 (Axiom system) Whenever an FD $X \rightarrow Y$ can be proven from F using a finite number of inference rules from Armstrong's axiom system, we write $\text{F} \vdash X \rightarrow Y$.

Armstrong's axiom system is *sound* if $\text{F} \vdash X \rightarrow Y$ implies $\text{F} \models_{\approx} X \rightarrow Y$. Correspondingly, Armstrong's axiom system is *complete* if $\text{F} \models_{\approx} X \rightarrow Y$ implies $\text{F} \vdash X \rightarrow Y$.

It is fundamental result in relational database theory that Armstrong's axiom system is sound and complete for FDs being satisfied in precise relation. Our next result shows that this result carries over to imprecise relations.

Theorem 3.2 Armstrong's axiom system is sound and complete for FDs being satisfied in imprecise relations.

Proof. The reader can verify that Armstrong's axiom system is sound; see also [24, Lemma 3.1]. For completeness we refer the reader to [26, Theorem 7.1]; this proof is still valid in our case, since a precise relation is just a special case of an imprecise relation. □

4 The Chase Procedure

The chase procedure is a fundamental theorem proving tool in relational database theory. The main uses of the chase procedure have being testing implication of data dependencies [21] and testing consistency of a relational database with respect to a set of data dependencies [14]. The chase procedure has also been extended to incomplete relational databases for the purpose of testing consistency [10, 18, 19].

In the context of imprecise relations we redefine the chase procedure for maintaining consistency rather than testing consistency. Assuming that an imprecise relation r is updated with information obtained from several different sources, it may be the case that at any given time the relation r is inconsistent (i.e. it is *not* consistent) with respect to a set of FDs F. For example, if we obtain two different prices for a given share. Thus we input r and F into the chase procedure and its output, denoted by $chase(r, \text{F})$, is a consistent relation over R with respect to F. In Section 5 we will show that in some accurate sense the chase procedure outputs the most precise approximation of r, which is also consistent with respect to F.

We investigate the properties of the chase procedure showing amongst other results that it outputs a consistent relation, it is unique and can be computed in polynomial time in the sizes of r and F, and it commutes with the *merge* operation.

The pseudo-code for the algorithm *chase*(r, F) (more generally called the *chase procedure*), which given the inputs r and F returns a relation over R, is presented in Algorithm 1.

Algorithm 1 (***chase(r,*** **F))**

1. **begin**
2.   Result := r;
3.   Tmp := $\emptyset$;
4.   **while** Tmp $\neq$ Result **do**
5.   Tmp := Result;
6.   **if** $\exists X \rightarrow Y \in$ F, $\exists t_1, t_2 \in$ Result such that $t_1[X] \approx t_2[X]$ but $t_1[Y] \not\approx t_2[Y]$ **then**
7. $\forall A \in Y - X,\ t_1[A], t_2[A] := t_1[A] \cup t_2[A]$;
8.   **end if**
9.   **end while**
10.   **return** *merge*(Result);
11. **end.**

We call an execution of line 7 in Algorithm 1 a *chase step*, and say that the chase step *applies* the FD $X \rightarrow Y$ to the *current state* of *chase*(r, F). Intuitively, at each chase step the chase procedure assigns the A-values of two tuples to coarser imprecise sets.

Example 4.1 The imprecise relation *chase*(r, F), where r is the relation over R shown in Table 4 and F is the set of FDs over R specified in Example 3.1 is shown in Table 6. The reader can verify that *chase*(r, F) $\models_{\approx}$ F, i.e. *chase*(r, F) is consistent, and that *chase*(r, F) = *chase*(*merge*(r), F), where *merge*(r) is shown in Table 5.

The next lemma shows that *chase*(r, F) preserves the basic information of r and is less precise than *merge*(r).

Lemma 4.1 The following two statements are true:

N	P	B
{0,1,2}	{0,1,2}	{0,1,2}
{3,4}	{3}	{3}

Table 6: The relation $chase(r, \mathrm{F})$

1. $inf(chase(r, \mathrm{F})) = inf(r)$.
2. $chase(r, \mathrm{F}) \sqsubseteq merge(r)$.

Proof. The first part can be verified by inspecting Algorithm 1. To prove the second part we use an induction on the minimal number, k, of chase steps needed to compute a current state s of $chase(r,\mathrm{F})$, asserting that $merge(s) \sqsubseteq merge(r)$ after k chase steps.

(*Basis*): If $k = 0$, then the result is immediate by line 10 in Algorithm 1, since $merge(s) = chase(r, \mathrm{F}) = merge(r)$.

(*Induction*): Assume the result holds when the minimal number of chase steps needed to compute s is k, with $k \geq 0$; we then need to prove that the result holds when the minimal number of chase steps to compute s is $k+1$. Let t_1, t_2 be the tuples in line 7 of Algorithm 1 just after the application of the $k+1$th chase step. Also, let r' be the current state of $chase(r, \mathrm{F})$ and $t'_1, t'_2 \in r'$ be the current states of t_1, t_2, respectively, prior to the application of the $k+1$th chase step.

On inspection of line 7 it follows that for $i = 1, 2, t_i \sqsubseteq t'_i$ and thus also $merge(\{t_i\}) \sqsubseteq merge(\{t'_i\})$. Therefore, $chase(r, \mathrm{F}) \sqsubseteq merge(r')$ holds. Furthermore, by inductive hypothesis $merge(r') \sqsubseteq merge(r)$. The result that $chase(r, \mathrm{F}) \sqsubseteq merge(r)$ follows as required by the transitivity of $\sqsubseteq$. $\square$

Lemma 4.2 The following two statements are true:

1. $chase(r, \mathrm{F})$ terminates in polynomial time in the sizes of r and F.
2. $chase(r, \mathrm{F})$ is unique.

Proof. $chase(r, \mathrm{F})$ terminates due to the following argument. Prior to each chase step we have that $\forall A \in Y - X,\ t_1[A], t_2[A] \subset t_1[A] \cup t_2[A]$, since by Proposition 2.1 $t_1[A]$ and $t_2[A]$ are disjoint. It follows that the A-values of tuples in r are strictly monotonically increasing with respect to $\subseteq$. It follows that $chase(r, \mathrm{F})$ terminates, since $\forall A \in \mathrm{R}$, $adom(r, A)$ is finite.

We now show that $chase(r, \mathrm{F})$ can be computed in polynomial time in the sizes of r and F. By Theorem 2.2 testing whether $t_1[X] \approx t_2[X]$ and whether $t_1[Y] \not\approx t_2[Y]$ can be done in polynomial time in the size of r. So, each execution of the while loop beginning at line 4 and ending at line 9 can be computed in polynomial time in the size of r and F. It remains to show that this while loop is executed a polynomial number of times in the sizes of r and F, recalling that by Corollary 2.4 $merge(\mathrm{Result})$ can be computed in polynomial time in the size of r.

Let m be the maximum cardinality, over all $A \in \mathrm{R}$, of an imprecise set over A with respect to r; m is less than the size of r, since $m \leq adom(r, A)$. The result of the first part now follows, since the while loop cannot be executed more than $m|r||\mathrm{R}|$ times due to the fact that the A-values of r are strictly monotonically increasing after each chase step.

In order to show that $chase(r, \mathrm{F})$ is unique we adopt the technique used in [3, Theorem 4.1]. Define two A-values $t_1[A]$ and $t_2[A]$, with $t_1, t_2 \in r$, to be *A-equatable* in r (or simply equatable if A and r are understood from context) if one of the following conditions holds:

1. $t_1[A] \approx t_2[A]$, or
2. for some FD $X \rightarrow Y \in \mathrm{F}$, with $A \in Y - X$, $t_1[X] \approx t_2[X]$, or
3. for some A-value, $t_3[A]$, with $t_3 \in r$, both $t_1[A]$ and $t_3[A]$ are equatable and also $t_2[A]$ and $t_3[A]$ are equatable, or
4. for some FD $X \rightarrow Y \in \mathrm{F}$, we have that $\forall B \in X$, $t_1[B]$ and $t_2[B]$ are equatable.

From the above definition it follows that equatability is an equivalence relation on the A-values of the tuples in r. Moreover, it can be verified that equatability is invariant with respect to chase steps. Therefore, $t_1[A]$ and $t_2[A]$ are equatable in r if and only if either $t_1[A] \approx t_2[A]$ or after some chase step $u_1[A] \approx u_2[A]$, where u_1 and u_2 are the current states of t_1 and t_2, respectively. Let $\{c_1, c_2, \ldots, c_n\}$ be an equivalence class with respect to A-equatability in r. It follows that $\bigcup_{i \in I} c_i$ is an imprecise set over A with respect to $chase(r, \mathrm{F})$. Let $t \in r$ and assume that $t[A] \approx c_i$, for some $i \in I$. The result follows, since by line 10 in Algorithm 1 which merges the result, we can deduce that, irrespective of the order of chase steps, $u[A] = \bigcup_{i \in I} c_i$, where u is the current state of t in $chase(r, \mathrm{F})$. □

The next theorem shows that the chase procedure outputs a consistent relation and that it is cover insensitive.

Theorem 4.3 The following two statements are true:

1. $chase(r, \mathrm{F}) \models_{\approx} \mathrm{F}$, i.e. $chase(r, \mathrm{F})$ is consistent.
2. If F and G are sets of FDs over R such that G is a cover of F, then $chase(r, \mathrm{F}) = chase(r, \mathrm{G})$,

Proof. Let $s = chase(r, \mathrm{F})$. For the first part assume that for some $X \rightarrow Y \in \mathrm{F}$, $s \not\models_{\approx} X \rightarrow Y$. Then for some $t_1, t_2 \in s, t_1[X] \approx t_2[X]$ but $t_1[Y] \not\approx t_2[Y]$. This leads to a contradiction, since we can deduce that $s \neq chase(r, \mathrm{F})$, due to the fact that the condition of the if statement of line 6 in Algorithm 1 is true.

For the second part, in order to show that $chase(r, \mathrm{F}) = chase(r, \mathrm{G})$, we adopt the technique used in [21, Theorem 3]. Let $\mathrm{H} = \mathrm{F} \cup \mathrm{G}$. By the first part $chase(r, \mathrm{F}) \models_{\approx} \mathrm{F}$, and thus by the definition of logical implication $chase(r, \mathrm{F}) \models_{\approx} \mathrm{H}$. Thus $chase(r, \mathrm{F}) = chase(r, \mathrm{H})$, since $\mathrm{F} \subseteq \mathrm{H}$, and thus chase steps need not apply any FD in H$-$F. The result follows, since on using the same argument it is also the case that $chase(r, \mathrm{G}) = chase(r, \mathrm{H})$. □

The next theorem shows that the chase procedure commutes with *merge*.

Theorem 4.4 $chase(r, \mathrm{F}) = chase(merge(r), \mathrm{F})$.

Proof. By Proposition 2.5 the merge of r induces a mapping ρ from r to $merge(r)$ such that $\rho(t) = t'$, with $t \in r$ and $t' \in merge(r)$, if and only if $t \approx_s t'$, with $s = \{t, t'\}$. The result now follows by the proof of Lemma 4.2 on using the notion of equatability of A-values, since two A-values $t_1[A]$ and $t_2[A]$, with $t_1, t_2 \in r$, are A-equatable in r if and only if $(\rho(t_1))[A]$ and $(\rho(t_2))[A]$ are A-equatable in $merge(r)$. □

5 The Most Precise Approximation of an Imprecise Relation

Herein we give a semantic characterisation of the chase procedure by showing that in some accurate sense it is the most precise approximation of an imprecise relation, which is also consistent with respect to a set of FDs. We first define the *join* of imprecise relations, which corresponds to the *least upper bound* [12] of these relations in the lattice MERGE(R).

We then define the *most precise approximation* of r with respect to F to be the *join* of all the consistent and merged relations which are less precise than r. Our main result is that *chase*(r, F) is exactly the most precise approximation of r with respect to F. Thus the chase procedure solves the consistency problem in polynomial time in the size of r and F.

Recall from Section 2 that MERGE(R) is a lattice with respect to $\sqsubseteq$. We will now define the *join* operation on relations in this lattice.

Definition 5.1 (Join operation) The *join* of two relations, $r_1, r_2 \in$ MERGE(R), denoted by $r_1 \sqcup r_2$, is given by

$$r_1 \sqcup r_2 = \{t \mid \exists t_1 \in r_1, \exists t_2 \in r_2 \text{ such that } \forall A \in \mathrm{R}, t[A] = t_1[A] \cap t_2[A] \text{ and } t[A] \neq \emptyset\}.$$

It can be verified that the join of two relations is their *least upper bound* in MERGE(R). (We note that the terms *least upper bound* and *join* are used synonymously in the lattice-theoretic literature [12].) It is interesting to note that the join operation generalises the *intersection* operation of the relational algebra (cf. [7]), which is a special case of the natural join operation [26, 3].

From now on we will assume that $r_1, r_2 \in$ MERGE(R). The next proposition shows that the join of two merged relations is also a merged relation.

Proposition 5.1 Let $r = r_1 \sqcup r_1$. Then $r = merge(r)$.

Proof. Suppose that $r \neq merge(r)$ and thus for some $A \in$ R, $\exists u_1, u_2 \in r$ such that $u_1[A] \cap u_2[A] \neq \emptyset$ and $u_1[A] \neq u_2[A]$. Let S $= u_1[A] \cap u_2[A]$.

It follows that $\exists t_1, t_1' \in r_1, t_2, t_2' \in r_2$ such that $u_1[A] = t_1[A] \cap t_2[A]$ and $u_2[A] = t_1'[A] \cap t_2'[A]$. Thus, S $\subseteq t_1[A], t_2[A]$ and also S $\subseteq t_1'[A], t_2'[A]$. It follows that $t_1[A] = t_1'[A]$ and that $t_2[A] = t_2'[A]$, since $r_1, r_2 \in$ MERGE(R) and respectively, S $\subseteq t_1[A] \cap t_2[A] \neq \emptyset$ and S $\subseteq t_1'[A] \cap t_2'[A] \neq \emptyset$. The result follows, since we can deduce that $u_1[A] = u_2[A]$ contrary to our assumption. $\Box$

The next theorem shows that if two relations are consistent then their join is also consistent. That is, the join operation preserves satisfaction of FDs. In order to understand this result let us assume that s_1 and s_2 are precise relations over R. Then, in this special case, the join $s_1 \sqcup s_2$ reduces to the intersection $s_1 \cap s_2$. Now, it is easy to verify that for precise relations intersection preserves FD satisfaction. Thus the next theorem generalises this result to imprecise relations.

Theorem 5.2 Let $r_1, r_2 \in$ SAT(F). Then $r_1 \sqcup r_2 \in$ SAT(F).

Proof. Let $r = r_1 \sqcup r_2$ and suppose that $r \notin$ SAT(F). It follows that for some FD, $X \to Y \in$ F and for some attribute $A \in Y$, $r \not\models_{\approx} X \to A$. Thus $\exists u_1, u_2 \in r$ such that $u_1[X] \approx u_2[X]$ but $u_1[A] \not\approx u_2[A]$. Moreover, by Propositions 3.1 and 5.1 $u_1[X] = u_2[X]$ and $u_1[A] \cap u_2[A] = \emptyset$.

It follows by Definition 5.1 that $\exists t_1, t_1' \in r_1$, $\exists t_2, t_2' \in r_2$ such that $\{u_1\} = \{t_1\} \sqcup \{t_2\}$ and $\{u_2\} = \{t_1'\} \sqcup \{t_2'\}$. Thus $u_1[X] \subseteq t_1[X], t_2[X]$ and $u_2[X] \subseteq t_1'[X], t_2'[X]$. It follows that $t_1[X] \cap t_1'[X] \neq \emptyset$ and also $t_2[X] \cap t_2'[X] \neq \emptyset$. Therefore, since $r_1, r_2 \in$ MERGE(R) we have that $t_1[X] = t_1'[X]$ and $t_2[X] = t_2'[X]$. Thus by Proposition 3.1 $t_1[A] = t_1'[A]$ and $t_2[A] = t_2'[A]$, since $r_1, r_2 \in$ SAT(F). It follows that $(t_1[A] \cap t_2[A]) = (t_1'[A] \cap t_2'[A])$ and therefore that $u_1[A] = u_2[A]$ contradicting our assumption. $\square$

The converse of Theorem 5.2 is false as the next example shows. Let r_1 and r_2 be the relations over R $= \{N, P\}$, shown in Tables 1 and 2, respectively. Then, it can be verified that $r_2 \not\models_{\approx} N \to P$ but $(r_1 \sqcup r_2 = r_1) \models_{\approx} N \to P$.

The most precise approximation of an imprecise relation r over R with respect to F is the join of all consistent relations s such that s is a merged relation that is less precise than r.

Definition 5.2 (Most precise approximation of an imprecise relation) The *most precise approximation* of an imprecise relation r over R with respect to F (or simply the approximation of r if R and F are understood from context), denoted by $approx(r, \mathrm{F})$, is given by

$$\bigsqcup \{s \mid s \sqsubseteq merge(r) \text{ and } s \in \mathrm{SAT(F)}\}.$$

We note that the join of a set of relations is well-defined, since the join operator is commutative and associative [12]. The next lemma shows some properties of approximations.

Lemma 5.3 The following four statements are true:

1. $inf(approx(r, \mathrm{F})) = inf(r)$.
2. $approx(r, \mathrm{F})$ is consistent.
3. $approx(r, \mathrm{F}) \sqsubseteq merge(r)$.
4. $approx(r, \mathrm{F}) = merge(r)$ if and only if r is consistent.

Proof. The first part follows by Definitions 5.1 and 5.2. The second part follows by Theorem 5.2, since by Definition 5.2 $approx(r, \mathrm{F})$ is the join of consistent and merged relations. The third part follows, since the join of several relations corresponds to their least upper bound due to the commutativity and associativity of the join. The *if part* of the fourth part follows by Definition 5.2, since if r is consistent then $r \in \mathrm{SAT(F)}$. The *only if* part of the fourth part follows by Theorem 5.2 on using Proposition 3.1 part (2), since by Definition 5.2 $approx(r, \mathrm{F})$ is the join of consistent relations. □

The next theorem, which is the main result of this section, shows that output of the chase procedure is equal to the corresponding most precise approximation. Thus the imprecise relation $chase(r, \mathrm{F})$, which is shown in Table 6, is the most precise approximation of r with respect to F, where r is the relation over R shown in Table 4 and F is the set of FDs over R specified in Example 3.1.

Theorem 5.4 $chase(r, \mathrm{F}) = approx(r, \mathrm{F})$.

Proof. We need to show that $chase(r, \mathrm{F}) \sqsubseteq approx(r, \mathrm{F})$ and that $approx(r, \mathrm{F}) \sqsubseteq chase(r, \mathrm{F})$.

($chase(r, \mathrm{F}) \sqsubseteq approx(r, \mathrm{F})$): By Lemma 4.1 part (2) $chase(r, \mathrm{F}) \sqsubseteq merge(r)$. Furthermore, by Theorem 4.3 part (1) and line 10 in Algorithm 1 $chase(r, \mathrm{F}) \in \mathrm{SAT(F)}$. Thus by Definition 5.2 $chase(r, \mathrm{F}) \sqsubseteq approx(r, \mathrm{F})$, since $\bigsqcup$ corresponds to the least upper bound operation in MERGE(R).

($approx(r, \mathrm{F}) \sqsubseteq chase(r, \mathrm{F})$): We use an induction on the minimal number, k, of chase steps needed to compute a current state s of $chase(r, \mathrm{F})$, asserting that $approx(r, \mathrm{F}) \sqsubseteq merge(s)$ after k chase steps.

(*Basis*): If $k = 0$, then the result is immediate by part (3) of Lemma 5.3, since $s = r$ and by line 10 in Algorithm 1 $chase(r, \mathrm{F}) = merge(r)$.

(*Induction*): Assume the result holds when the minimal number of chase steps needed to compute s is k, with $k \geq 0$; we then need to prove that the result holds when the minimal number of chase steps to compute s is $k+1$. Let r' be the current state of $chase(r, \mathrm{F})$ prior to the application of the $k+1$th chase step.

Thus, by inductive hypothesis $approx(r, \mathrm{F}) \sqsubseteq merge(r')$. By inspection of line 7 in Algorithm 1 we can deduce that $chase(r, \mathrm{F})$ is the maximal consistent and merged relation with respect to $\sqsubseteq$, which is less precise than $merge(r')$. That is, $chase(r, \mathrm{F}) \sqsubseteq merge(r')$ and if for some $s' \in \mathrm{SAT(F)}$ we have that $s' \sqsubseteq merge(r')$ then it is also true that $s' \sqsubseteq chase(r, \mathrm{F})$. The result follows, since by Proposition 5.1 and Lemma 5.3 part (2) $approx(r, \mathrm{F}) \in \mathrm{SAT(F)}$. □

6 Concluding Remarks

We have defined the notion of an imprecise relation which builds upon the fuzzy relational model [6, 23, 24]. An imprecise set is an equivalence class of equally likely values in an imprecise relation. The *merge* operation was then defined in order to replace each tuple value $t[A]$ by the imprecise set induced by the values $v \in t[A]$. We defined the partial order $\sqsubseteq$ on sets of values in $\mathcal{P}(\mathcal{D})$, which when extended to imprecise relations yields a lattice on the set of merged relations in MERGE(R).

We extended FDs to imprecise relations, using equally likely sets of values rather than equality, and in Theorem 3.2 showed that Armstrong's axiom system is sound and complete for such FDs. We then defined the chase procedure over imprecise relations as a means of maintaining consistency of an imprecise relation with respect to a set of FDs. It was shown that the fundamental properties of the chase procedure over precise relations [21, 14] also hold for the chase procedure over imprecise relations. In particular, in Lemma 4.2 we showed that $chase(r, \mathrm{F})$ is unique and can be computed in polynomial time in the size of r and F. Furthermore, in Theorem 4.3 we showed that $chase(r, \mathrm{F})$ is consistent and cover invariant. In Theorem 4.4 we showed that the chase procedure commutes with *merge*.

In order to give a semantic characterisation of the chase procedure we defined the concept of the most precise approximation of an imprecise relation with respect to a set of FDs. We defined the join operation on imprecise relations in MERGE(R) and were able to show in Theorem 5.2 that if two relations are consistent then their join is also consistent. The most precise approximation of an imprecise relation r over R with respect to F is the join of all consistent relations s such that s is a merged relation that is less precise than r. Our final and main result is that of Theorem 5.4 which states that $chase(r, \mathrm{F}) = approx(r, \mathrm{F})$ implying that the chase procedure outputs the most precise approximation of r with respect to F. We conclude positively that given an imprecise relation r, which may not be consistent, the chase procedure solves the consistency problem in polynomial time in the size of r and F.

It may be the case that we have at our disposal some knowledge about the distribution of values in an imprecise set. In this case we can relax our assumption that the values in the imprecise set are equally likely. We are currently investigating a generalisation of our model that takes this additional information into account.

Throughout the paper we have assumed that we have a single imprecise relation r over R and a set of FDs F over R. It would interesting to extend the results of this paper to deal with the case of an imprecise database. An imprecise database is a family of imprecise relations each over a relation schema R_i together with an associated local set of FDs F_i over R_i. In this case we could distinguish between the notions of *local consistency* and *global consistency*. Local consistency is the satisfaction of each local set of FDs F_i by the relation in the database over R_i. On the other hand, global consistency is the satisfaction of the the global set of FDs $\bigcup_i F_i$ in some appropriate *representative instance* of the imprecise database [14, 18]. We then say that the database is *independent* with respect its global set of FDs if local satisfaction implies global satisfaction [11].

Acknowledgements. The author would like to thank the referees for their constructive criticism and helpful remarks.

References

[1] Aho, A.V., Hopcroft, J.E., Ullman, J.D. (1983) *Data Structures and Algorithms*. Addison-Wesley, Reading Ma.

[2] Armstrong, W.W. (1974) Dependency structures of data base relationships. In *Proceedings of the IFIP Congress*, Stockholm, pp. 580-583.

[3] Atzeni, P. and De Antonellis, V. (1993) *Relational Database Theory.* Benjamin/Cummings, Redwood City, Ca.

[4] Badrinath, B.R. and Imielinski, T. (1992) Replication and mobility. In *Proceedings of the 2nd IEEE Workshop on Management of Replicated Data.*

[5] Bosc, P., Dubois, D. and Prade, H. (1994) Fuzzy functional dependencies - An overview and a critical discussion. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, Orlando, Florida, pp. 325-330.

[6] Buckles, B.P. and Petry, F.E. (1982) A fuzzy representation of data for relational databases. *Fuzzy Sets and Systems*, **7**, 213-226.

[7] Buneman, P., Jung, A. and Ohori, A. (1991) Using powerdomains to generalize relational databases. *Theoretical Computer Science*, **91**, 23-55.

[8] Codd, E.F. (1979) Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems*, **4**, 397-434.

[9] Garey, M.R. and Johnson, D.S. (1979) *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman, New York.

[10] Grahne, G. (1984) Dependency satisfaction in databases with incomplete information. In *Proceedings of the International Conference on Very Large Data Bases*, Singapore, pp. 37-45.

[11] Graham, M.H. and Yannakakis, M. (1984) Independent database schemas. *Journal of Computer and System Sciences*, **28**, 121-141.

[12] Grätzer, G. (1978) *General Lattice Theory*. Academic Press, New York.

[13] Halmos, P.R. (1974) *Naive Set Theory*. Springer-Verlag, New York.

[14] Honeyman, P. (1982) Testing satisfaction of functional dependencies. *Journal of the ACM*, **29**, 668-677.

[15] Imielinski, T. (1989) Incomplete information in logical databases. *IEEE Data Engineering*, **12** 29-40.

[16] Keen, D. and Rajasekar, A. (1993) Rough sets and data dependencies. In *Proceedings of the Workshop on Incompleteness and Uncertainty in Information Systems*, Montreal, pp. 87-101.

[17] Lakhamanan, L.V.S. and Johnstone, H.L. (1993) A relational model for manipulating probabilistic knowledge. In *Proceedings of the Workshop on Incompleteness and Uncertainty in Information Systems*, Montreal, pp. 138-152.

[18] Levene, M. and Loizou, G. (1994) The nested universal relation model. *Journal of Computer and System Sciences*, **49**, 683-717.

[19] Levene, M. and Loizou, G. (1996) The additivity problem for functional dependencies in incomplete relations. To appear in *Acta Informatica*.

[20] Lipski Jr., W. (1979) Semantic issues connected with incomplete information databases. *ACM Transaction on Database Systems*, **4**, 262-296.

[21] Maier, D., Mendelzon, A.O. and Sagiv, Y. (1979) Testing implications of data dependencies, *ACM Transaction on Database Systems*, **4**, 455-469.

[22] Pawlak, Z. (1982) Rough sets, *International Journal of Computer and Information Sciences* **11**, 341-356.

[23] Shenoi, S., Melton, A. and Fan, L.T. (1990) An equivalence classes model of fuzzy relational databases. *Fuzzy Sets and Systems*, **38**, 153-170.

[24] Shenoi, S. and Melton, A. (1992) Functional dependencies and normal forms in the fuzzy relational database model. *Information Sciences*, **60**, 1-28.

[25] Smyth, M.B. (1978) Power domains. *Journal of Computer and System Sciences*, **16**, 23-36.

[26] Ullman, J.D. (1988) *Principles of Database and Knowledge-Base Systems, Vol. I.*. Computer Science Press, Rockville, Md.

[27] Vadaparty, K. and Naqvi, S. (1995) Using constraints for efficient query processing in nondeterministic databases. *IEEE Transaction on Knowledge and Data Engineering*, **7**, 850-864.

[28] Zadeh, L.A. (1971) Similarity relations and fuzzy orderings, *Information Sciences*, **3**, 177-200.

[29] Zaniolo, C. (1984) Database relations with null values. *Journal of Computer and System Sciences*, **28**, 142-166.