Imperfect knowledge in databases

Esteban Zimányi Alain Pirotte

Abstract

The field of imperfect knowledge (i.e., uncertain, imprecise, incomplete) is characterized by: (1) an active research area largely within relational theory; (2) a great variety of approaches that are not easily comparable, and (3) few practically usable results because of the inherent complexity of representing and manipulating imperfect knowledge.

This paper draws together and analyzes a number of proposals from the literature, and it identifies several research issues. Our aim was to give a unifying overview by formulating the database semantics both with a model-theoretic flavor (sets of possible worlds) and with a proof-theoretic flavor (sets of first-order formulas). Besides the relational model, we briefly describe some works on non-first normal form relations and the universal relation model.

1 Introduction

A database is an abstraction of a piece of the real world. In such a process, both the extension (the relevant entities) and the level of detail are restricted according to the needs of a number of applications that will use the data in the database. Thus a database is always an imperfect picture of the real world.

In database management, data is modeled on two levels: the database schema specifies the structure of the database while the database extension represents a set of facts or events of the real world. The *Appropriate Scheme Assumption* [Bis83] makes explicit the assumption that this two-level description is a valid view of the world. Such a distinction between schema and extension is usually blurred in artificial intelligence.

The information contained in a database is *complete* and *certain* if it represents an accurate and adequate picture of the corresponding application domain in the real world. However, in real life, information is often *imperfect* in several ways. Giving a precise characterization of imperfectness is not easy. Possible definitions follow. A piece of information in a database is *uncertain* if the corresponding piece of information in the real world is imperfectly known. A piece of information in a database is *imprecise* if it approximates the corresponding piece of information in the real world. With these definitions, vagueness is similar to imprecision, while incompleteness is an extreme case of imprecision, where a piece of information is missing.

Very often, uncertainty and imprecision go together as two facets of the same phenomenon. Also a characterization of what is uncertain and what is imprecise is relative to the pragmatics of database usage or simply to the way of phrasing the imperfect piece of information.

For example, the fact "John is between 34 and 40 years of age" may be seen as a certain but imprecise piece of information. For the same reality, the fact "John is between 35 and 37 years of age" could be less certain but more precise.

Data manipulation, that is, query answering, integrity constraints checking, and database updates, must be redefined to take the imperfect knowledge into account. The choice of which incompleteness and uncertainty to introduce in the database depends on the availability of mechanisms and algorithms for manipulating such information in a semantically correct way. Another concern is the complexity of manipulating incompleteness or uncertainty, which should not prevent its application to practical situations. Thus, the treatment of incompleteness and uncertainty is a trade-off between richness of representation and (in)efficiency of manipulation.

Null values have been most widely used to model incomplete information. There are many possible nuances of nulls (the ANSI/X3/SPARC study group [ANS75] listed 14 of them). This paper reviews the most important ones in addition to other approaches to incomplete and uncertain information listed below.

Existential null values express that the existence of an attribute value is known but the specific value is not. If the attribute is single-valued, the null value is called *atomic* existential value; an example is when the age of a person is unknown. When the attribute is multi-valued, the null value is called *set* existential value; an example is when it is known that a student is registered in at least one course without knowing which one(s). Existential null values are presented in Section 4.

Or-sets are particular kinds of existential null values where the unknown attribute takes its value from a specified subset of a database domain. Different types of or-sets are presented in Section 4.3.

Inexistent or *inapplicable null values* are used when an attribute cannot have a value. For example, for a single person, the name of the spouse is inapplicable. Inexistent null values are presented in Section 5.

Universal null values represent that a property is satisfied for all the possible values of an attribute. For example, a professor may teach the same unknown course in all departments.

Default null values allow to represent that an object has a predetermined value, unless explicitly specified otherwise. For example, all employees will have a 5% salary raise, except those working in a specific department.

No-information or open null values correspond to the case where it is not known whether a value exists. For example, if the marital status of a person is unknown, the name of the spouse is a no-information null value. As for existential values, there are two possible interpretations: *atomic* and *set*. Universal, default, and open null values are presented in Section 6.

Incomplete information in a database may also take the form of general *disjunctive information*, as in "John teaches Physics or Mary teaches Algebra". Disjunctive databases are presented in Section 11.

Maybe information is uncertain information, as in "Jean possibly teaches Physics". As shown in Section 10, maybe information is related to disjunctive information.

Uncertainty can also be take the form of *probabilistic databases*, presented in Section 12. An example is "John teaches Physics with probability 0.8".

The rest of this paper is organized as follows. Section 2 introduces the important semantics of possible worlds. Section 3 is a motivating section presenting the basic manipulation mechanisms for an imperfect database and illustrating them with or-sets and null values. This section can be omitted on a first reading as it uses concepts presented in more detail in subsequent sections. Section 4 is devoted to a systematic presentation of existential null values and their generalizations as or-sets. Section 5 treats inexistent values and Section 6 the open databases. Section 7 briefly considers the combination of null values, an area where many questions remain open. Section 8 considers the universal relation model. Section 9 addresses nested relational databases. Section 10 addresses maybe information. Section 11 is devoted to general disjunctive databases. Finally, Section 12 is devoted to probabilistic databases.

2 Possible worlds

A database represents information about a part of the real world. If the information available is complete and certain, then there is a clear correspondence between the database and the real world. On the contrary, when knowledge about the world is incomplete or uncertain, several scenarios or states with complete and certain knowledge are possible, but it is not known which one represents the real state of the world.

Thus, a database containing incomplete or uncertain information implicitly represents a set of possible states or *possible worlds.* A *possible world* is a hypothetical state of the world that may be represented by an ordinary database with complete and certain information. If the knowledge about the world is represented with a logical theory, each possible world corresponds to a model of the theory.

Consider for example relations *teaches*(*professor*,*course*) and *takes*(*student*,*course*) shown in Figure 1, where variables x and y represent unknown values or, more precisely, *null values* of the

teaches

professor	*course*
Martin	Physics
x	Algebra

student	*course*
Paul	Physics
Peter	y

takes

Figure 1: Incomplete relations *teaches* and *takes.*

type "value exists, but is unknown". As we will see in next sections, there are different ways of interpreting these relations depending on how the null values and the relations are interpreted.

The semantics of a database with incomplete or uncertain information assigns, to each relation R, the value of a representation function $REP(R)$, which is a set of possible worlds. More generally, the representation function associates to a database (i.e., a set of relations) a set of classical relational databases which are its possible worlds.

For example, suppose that for relations *teaches* and *takes* above, the null values x and y represent, respectively, the or-sets $\{\text{Martin}, \text{Anne}\}$ and $\{\text{Algebra}, \text{Calculus}\}$ and that the or-sets are exclusive (i.e., either Martin or Anne, but not both, teaches Algebra). Suppose in addition that the relations are interpreted under the *Closed World Assumption* [Rei78], stating that all facts not explicitly represented in the relations are false. Thereby, the possible worlds $REP(\langle teaches, takes\rangle)$ of the relations are as in Figure 2.

$$\begin{array}{lcl}
M_1 & : & teaches = \{(\text{Martin,Physics}),(\text{Martin,Algebra})\}, \\
 & & takes = \{(\text{Paul,Physics}),(\text{Peter,Algebra})\} \\
M_2 & : & teaches = \{(\text{Martin,Physics}),(\text{Martin,Algebra})\}, \\
 & & takes = \{(\text{Paul,Physics}),(\text{Peter,Calculus})\} \\
M_3 & : & teaches = \{(\text{Martin,Physics}),(\text{Anne,Algebra})\}, \\
 & & takes = \{(\text{Paul,Physics}),(\text{Peter,Algebra})\} \\
M_4 & : & teaches = \{(\text{Martin,Physics}),(\text{Anne,Algebra})\}, \\
 & & takes = \{(\text{Paul,Physics}),(\text{Peter,Calculus})\}
\end{array}$$

Figure 2: $REP(\langle teaches, takes\rangle)$.

If the relations are considered with other interpretation of nulls, or under another closure assumption, $REP(\langle teaches, takes\rangle)$ will yield another set of possible worlds.

The notion of possible worlds provides a convenient tool that we use throughout this paper to investigate the semantics of both the representation and the manipulation of imperfect information.

3 Manipulating an imperfect database

This section illustrates the basic mechanisms for defining, with the help of possible worlds, the manipulation of an imperfect database. Two different points of view must be considered. Under a *semantical* point of view, the transformation of a database represented by a set of possible worlds $\mathcal{X}$ produces another set $\mathcal{Y}$ of possible worlds representing the intuitive meaning of the operation. Under a *syntactical* point of view, a set of extended relations $\mathbf{R}$ yields another set of extended relations $\mathbf{R}'$. However, in order that the manipulation be "faithful", the set of relations $\mathbf{R}'$ must capture the meaning represented by the set of possible worlds $\mathcal{Y}$.

3.1 Queries

With complete and sure information, every assertion is either true or false in the model of the world that is described by the database and queries have a certain result made of a list of sure assertions. On the contrary, when only partial or uncertain information is available, an assertion has a truth value in each possible world: either it is true in all the possible worlds, or it is false, or it is true in some worlds and false in others. Thus, the mechanisms for database manipulation must be adapted accordingly.

Suppose that a database is composed of a set of "extended" relations in which imperfect information is represented by null values, disjunctions, probabilities, etc. In order to process queries applied to such extended relations, the relational algebra operators have to be generalized while remaining "faithful" to the underlying semantics defined by the representation function *REP* based on possible worlds.

Consider again relations *teaches* and *takes* of Figure 1 and the associated set of possible worlds $\mathit{REP}(\langle \mathit{teaches}, \mathit{takes}\rangle)$ given in Figure 2. Suppose we want to obtain, through a generalized join of *teaches* and *takes*, a relation representing which professors teach which courses to which students. The possible worlds $\mathit{REP}(\langle \mathit{teaches}, \mathit{takes}\rangle)$ are the basis for the meaning of the generalized join. Since all the possible worlds are equally likely, the possible results for the join are obtained by performing a classical join in each possible world of $\mathit{REP}(\langle \mathit{teaches}, \mathit{takes}\rangle)$, as given in Figure 3. The relation obtained as result of the generalized join of *teaches* and *takes* should have these relations as its set of possible worlds.

$$\begin{aligned}
M_1 &= \{(\text{Martin,Physics,Paul}),(\text{Martin,Algebra,Peter})\}\\
M_2 &= \{(\text{Martin,Physics,Paul})\}\\
M_3 &= \{(\text{Martin,Physics,Paul}),(\text{Anne,Algebra,Peter})\}
\end{aligned}$$

Figure 3: *teaches* $\bowtie$ *teaches* in the possible worlds.

Formally, let f be an arbitrary relational expression over a multirelation $\mathbf{R}$, i.e., over a collection $\mathbf{R} = \langle R_1, \ldots, R_k\rangle$ of extended relations. The best way to define the extension $\bar{f}(\mathbf{R})$ of f over $\mathbf{R}$ is [IV89] such that:

$$\mathit{REP}(\bar{f}(\mathbf{R})) = f(\mathit{REP}(\mathbf{R})) = \{f(\mathbf{r}) \mid \mathbf{r} \in \mathit{REP}(\mathbf{R})\}. \tag{3.1}$$

This definition is the strongest requirement that can be made, since it states that the expressive power of a given type of extended relation $\mathbf{R}$ is sufficient to exactly represent the results of all relational algebra expressions as extended relations $\bar{f}(\mathbf{R})$ of the same kind as $\mathbf{R}$.

Condition (3.1), referred to as *strong correctness criterion*, is schematized in Figure 4: both paths from the upper left corner to the lower right corner must produce the same result.

This requirement however is too strong for several extensions [IL84, Lip84], even if expression $\bar{f}$ is restricted to a subset of relational algebra. This means that the desired result $f(\mathit{REP}(\mathbf{R}))$

$$\begin{tikzcd}
\mathbf{R} \arrow[r, "REP"] \arrow[d, "\bar{f}"'] & PW \arrow[d, "f"] \\
\mathbf{R}' \arrow[r, "REP"'] & PW'
\end{tikzcd}$$

Figure 4: Strong correctness of f.

of some expressions is not representable by any extended relation of the same kind as $\mathbf{R}$. There are two ways to try to cope with the problem: further generalize the relations to enhance their expressive power, or give up strong correctness and define weaker criteria.

For example, it can be shown that the possible worlds of Figure 3 cannot be represented by a relation with existential null values. They are represented by the following relation:

professor	*course*	*student*
Martin	Physics	Paul
z	Algebra	Peter

where the null value z represents the or-set $\{\text{Anne}, \text{Martin}, \emptyset\}$ (see Section 4.3).

When a particular kind of relations is too weak to represent results of arbitrary expressions f, an alternative to further generalization is to accept approximations. The semantics of possible worlds may be used to specify *weaker correctness criteria* and to measure the information loss. One such weak correctness criterion could be that $\bar{f}$ capture everything in $f(REP(\mathbf{R}))$ and as little extra as possible [Mai83]. This can be stated as:

$$REP(\bar{f}(\mathbf{R})) \supseteq f(REP(\mathbf{R}))$$

with the additional condition that there is no multirelation $\mathbf{R}'$ such that

$$REP(\bar{f}(\mathbf{R})) \supset REP(\mathbf{R}') \supseteq f(REP(\mathbf{R})).$$

Clearly, if an operator satisfies the strong correctness criterion, it also satisfies this weak criterion.

As another approximation, suppose that, instead of defining the semantics of a query f as $\{f(\mathbf{r}) \mid \mathbf{r} \in REP(\mathbf{R})\}$, we are interested in $\bigcap_{\mathbf{r} \in REP(\mathbf{R})} f(\mathbf{r})$, i.e., the information that is true in all possible worlds. For example, in Figure 3, Martin teaches Physics to Paul in all possible worlds. This motivates the notion of *true sets*.

Given a set of possible worlds $\mathcal{X}$, the true set $\cap\mathcal{X}$ is the set of tuples belonging to all worlds in $\mathcal{X}$. The requirements for a faithful extension of relational operations on a multirelation $\mathbf{R}$ then reduce to [Imi89]:

(1) *Preservation of true sets*: any relational algebra expression f must satisfy

$$\cap\, REP(\bar{f}(\mathbf{R})) = \cap f(REP(\mathbf{R})),$$

i.e., the relation $\bar{f}(\mathbf{R})$ should preserve information about all tuples which for sure belong to $f(REP(\mathbf{R}))$.

(2) *Recursiveness*: $\bar{f}(\bar{g}(\mathbf{R})) = (\bar{f}\bar{g})(\mathbf{R})$. This states that, as in the usual relational algebra, intermediate results of subexpressions can be saved and used for computing the full expression.

Requirements (1) and (2) are truly minimal for any reasonable extension of the relational algebra. They are still restrictive, i.e., extensions of some operations operating on some type of relations will be impossible. A common mistake in extending the relational algebra is to consider the correctness of each operation separately while ignoring their interaction. These interactions are captured by the recursiveness requirement.

3.2 Integrity constraints

Integrity constraints express properties that any database instance must verify in order to be a legal state of the database schema. We consider here two of the more common constraints: *functional* and *multivalued dependencies*. Given a relation scheme $\mathcal{R}(A, B, C)$, the functional dependency $A \to B$ can be expressed by the formula

$$\forall x\, \forall y\, \forall z\, \forall y_1\, \forall z_1\, r(x, y, z) \wedge r(x, y_1, z_1) \to y = y_1.$$

The multivalued dependency $A \to\to B$ can be expressed by the formula

$$\forall x\, \forall y\, \forall z\, \forall y_1\, \forall z_1\, r(x, y, z_1) \wedge r(x, y_1, z) \to r(x, y, z).$$

A relation r satisfies a dependency δ if r is a model of δ (in the sense of mathematical logic). A relation r satisfies a set of dependencies Σ if it satisfies all δ's in Σ. Given a set of dependencies Σ defined over a scheme $\mathcal{R}$, $Sat(\Sigma)$ is the set of relations on $\mathcal{R}$ that satisfy Σ.

The meaning of a database specified by a multirelation $\mathbf{r} = \langle r_1, \ldots, r_k\rangle$ and a set Σ of dependencies is called the *completion of* $\mathbf{r}$ *with respect to* Σ and is denoted by $comp_\Sigma(\mathbf{r})$. The completion $\mathbf{s} = \langle s_1, \ldots, s_k\rangle$ of $\mathbf{r}$ should satisfy [Mai83]:

(1) $r_i \subseteq s_i$ for all i,
(2) $\mathbf{s} \in Sat(\Sigma)$,
(3) for any $\mathbf{s}'$ that satisfy (1) and (2), $s_i \subset s'_i$ for at least one i.

The intuition is that (1) the original information should not decrease; (2) the completion should satisfy the dependencies; and (3) the increase should be minimal. If the completion exists, it is unique, and if it does not exist, then the database is inconsistent.

In the case of an imperfect database, a meaning must be assigned to a set of possible worlds $\mathcal{X}$ in the presence of a set of dependencies Σ. Since each possible world represents a conventional relational database, the notion of satisfaction of a set of dependencies generalizes to $\mathcal{X} \subseteq Sat(\Sigma)$. Let $\mathcal{X}$ be the following set of possible worlds of a relation *teaches*(*professor*,*course*,*department*)

$$\begin{aligned} M_1 &= \{(\text{Paul,Calculus,Zoology}),(\text{Paul,Algebra,Botany})\} \\ M_2 &= \{(\text{Thomas,Calculus,Zoology}),(\text{Anne,Calculus,Botany})\} \\ M_3 &= \{(\text{Anne,Calculus,Zoology}),(\text{Anne,Calculus,Botany}), \\ &\qquad (\text{Anne,Algebra,Zoology}),(\text{Anne,Algebra,Botany})\} \end{aligned}$$

and $\Sigma = \{\delta_1 = \mathit{prof} \to\to \mathit{course}, \delta_2 = \mathit{course} \to \mathit{prof}\}$. $\mathcal{X}$ should be changed in order to satisfy Σ. To satisfy δ_1, the tuples (Paul, Calculus, Botany) and (Paul, Algebra, Zoology) should be added to M_1. M_2 violates δ_2 and the only possibility is to drop it from the completion. M_3 satisfies both δ_1 and δ_2.

Thus, the completion of a set of possible worlds $\mathcal{X}$ with respect to a set Σ of dependencies is obtained by replacing each multirelation in $\mathcal{X}$ by its completion, provided that it exists [Gra91]:

$$Comp_\Sigma(\mathcal{X}) = \{comp_\Sigma(\mathbf{r}) \mid \mathbf{r} \in \mathcal{X}\}.$$

Consider now the syntactical point of view. Given a multirelation $\mathbf{R}$ containing imperfect knowledge, the natural interpretation that $\mathbf{R}$ satisfies Σ is:

$$REP(\mathbf{R}) \subseteq Sat(\Sigma).$$

Thus we need to define a function $\overline{Comp}$ for transforming $\mathbf{R}$ in order to cut down $REP(\mathbf{R})$ to $REP(\mathbf{R}) \cap Sat(\Sigma)$ for any multirelation $\mathbf{R}$ and set of dependencies Σ.

$$\begin{tikzcd}\mathbf{R} \arrow[r, "REP"] \arrow[d, "\overline{Comp}_{\Sigma}"'] & PW \arrow[d, "Comp_{\Sigma}"] \\ \mathbf{R}' \arrow[r, "REP"'] & PW'\end{tikzcd}$$

Figure 5: Completion with respect to dependencies Σ.

Furthermore, $\overline{Comp}$ should be defined so that the possible worlds of the transformed multirelation are the same as those obtained by completing the possible worlds individually, as schematized in Figure 5.

Consider a relation *teaches* as follows:

prof.	*course*	*dep.*
Anne	Calculus	Zoology
x	Algebra	Botany

and the functional dependency $\delta_2 = prof \rightarrow course$. To represent $\overline{Comp}_{\delta_2}(teaches)$ we must be able to express the fact that x cannot be equal to Anne. Now consider the same relation with the multivalued dependency $\delta_1 = prof \rightarrow\rightarrow course$. Any relation in $Comp_{\delta_1}(REP(teaches))$ should include two or four tuples depending on the value of x. Thus, in order to define $\overline{Comp}_{\delta_1}(teaches)$, we need a C-relation [IL84] as follows:

prof.	*course*	*dep.*	*Con*
Anne	Calculus	Zoology	
x	Algebra	Botany	
x	Calculus	Botany	$x =$ Anne
x	Algebra	Zoology	$x =$ Anne

stating that the last two tuples belong to the relation only in the case that condition $x =$ Anne is verified. C-relations are presented in Section 4.

3.3 Updates

An update to an imperfect database is correct if the possible worlds of the updated database are the same as those obtained by applying the update separately to each possible world of the original database. This is schematized in Figure 6. Thus, for example, the semantics for inserting a tuple t into a database $\mathcal{X}$ is given by inserting t into each possible world of $\mathcal{X}$.

$$\begin{tikzcd}\mathbf{R} \arrow[r, "REP"] \arrow[d, "\text{update}"'] & PW \arrow[d, "\text{update}"] \\ \mathbf{R}' \arrow[r, "REP"'] & PW'\end{tikzcd}$$

Figure 6: Update of an imperfect database.

Consider a *teaches* relation:

professor	*course*
Tom	Physics
Luc	x
y	Algebra

where the domains for attributes *professor* and *course* are, respectively, $\{\text{Tom}, \text{Luc}, \text{Anne}\}$ and $\{\text{Physics}, \text{Algebra}\}$. The possible worlds of the relation are given in Figure 7.

$$
\begin{aligned}
M_1 &= \{(\text{Tom,Physics}),(\text{Luc,Physics}),(\text{Tom,Algebra})\}\\
M_2 &= \{(\text{Tom,Physics}),(\text{Luc,Physics}),(\text{Luc,Algebra})\}\\
M_3 &= \{(\text{Tom,Physics}),(\text{Luc,Physics}),(\text{Anne,Algebra})\}\\
M_4 &= \{(\text{Tom,Physics}),(\text{Luc,Algebra}),(\text{Tom,Algebra})\}\\
M_5 &= \{(\text{Tom,Physics}),(\text{Luc,Algebra})\}\\
M_6 &= \{(\text{Tom,Physics}),(\text{Luc,Algebra}),(\text{Anne,Algebra})\}
\end{aligned}
$$

Figure 7: Possible worlds of *teaches*.

Updates themselves can be incompletely specified. For example, the information that "Anne teaches an unknown course" is represented by a pair of one-tuple possible worlds $\mathcal{Y} = \{\{(\text{Anne}, \text{Physics})\}, \{(\text{Anne},$
Thus, in general, update operations must be viewed as set-theoretical operations involving sets of possible worlds.

Ordinary relational updates are not sufficiently powerful to express all desirable transformations on a set of possible worlds. For example, with ordinary updates, there is no way to add new possible worlds or to eliminate possible worlds when they become inappropriate. We will now describe several types of updates to incomplete databases, as defined in [AG85].

Given two sets of possible worlds $\mathcal{X}$ and $\mathcal{Y}$, the *insertion* of $\mathcal{Y}$ into $\mathcal{X}$ is given by the pairwise union

$$\mathcal{X}(\cup)\mathcal{Y} = \{x \cup y \mid x \in \mathcal{X} \wedge y \in \mathcal{Y}\},$$

i.e., a possibility for the result is obtained by inserting a possibility of $\mathcal{Y}$ into a possibility of $\mathcal{X}$. For example, inserting into the database $\mathcal{X}$ of Figure 7 the set of possible words $\mathcal{Y}$ representing the fact "Anne teaches an unknown course" yields a set of 12 possible worlds.

The *deletion* of $\mathcal{Y}$ from $\mathcal{X}$ is accomplished by the pairwise difference

$$\mathcal{X}(-)\mathcal{Y} = \{x - y \mid x \in \mathcal{X} \wedge y \in \mathcal{Y}\}.$$

For example, the result of deleting all facts about Tom from database $\mathcal{X}$ of Figure 7 can be defined by $\mathcal{X}(-)\mathcal{Y}_2$ where $\mathcal{Y}_2 = \{\{(\text{Tom}, \text{Physics}), (\text{Tom}, \text{Algebra})\}\}$.

The knowledge contained in two databases $\mathcal{X}$ and $\mathcal{Y}$ can be integrated into one database by taking the tuples common to a possible state in $\mathcal{X}$ and a possible state in $\mathcal{Y}$ as one possible state of the new database. This *integration* update is defined by the pairwise intersection

$$\mathcal{X}(\cap)\mathcal{Y} = \{x \cap y \mid x \in \mathcal{X} \wedge y \in \mathcal{Y}\}.$$

The *subjection* update is used when our knowledge of the world has increased so that only some specific states of a database $\mathcal{X}$ remain possible. If, for instance, Tom is now known to teach only one course and if $\mathcal{Y}_3$ are all states satisfying this fact, the database must keep only those states in $\mathcal{X}$ that are also in $\mathcal{Y}_3$, i.e., $\mathcal{X} \cap \mathcal{Y}_3$ must replace $\mathcal{X}$.

Similarly, negative knowledge can be incorporated into a database. If we know that Luc does not teach Algebra and $\mathcal{Y}_4$ is the set of all states where Luc teaches Algebra, the database

$\mathcal{X}$ must be replaced by $\mathcal{X} - \mathcal{Y}_4$. Notice the implicit use of the Closed World Assumption, since the absence of a fact in a possible state means that the fact is false in that state.

Notice also that, unlike updates based on pairwise operations, the subjection updates only restrict the set of possible states of the database: the true state of the database is not modified, it is our knowledge of it that gets more precise.

The *augmentation* of $\mathcal{X}$ with $\mathcal{Y}$ is defined as $\mathcal{X} \cup \mathcal{Y}$. The intuitive meaning is that, in addition to the set $\mathcal{X}$ of possible worlds already known, other possible worlds $\mathcal{Y}$ are also plausible. Hence, the new set of possible worlds are those that are either in $\mathcal{X}$ or in $\mathcal{Y}$.

Finally, the *modification* update applies a transformation f to each possible world. For instance, f could modify Anne's course to Physics in the set of possible worlds $\mathcal{X}$ of Figure 7.

Notice that the modification can be incompletely specified. If we want to modify either Anne's course or Luc's course to Physics, this is accomplished by a modification F which is a set of two modification functions: one is like the previous one and the other modifies Luc's course to Physics. The result of the modification is thus defined as

$$F(\mathcal{X}) = \{f(x) \mid f \in F \wedge x \in \mathcal{X}\}.$$

So far, we have considered the desired semantics of updates in the form of general operations on sets of possible worlds. As for query evaluation and dependency checking, we need a syntactical way of updating an imperfect database so that the update captures the semantics of the operation in the possible worlds.

Consider the following relations *teaches*(*professor*,*course*) and *takes*(*student*,*course*,*dept.*)

professor	*course*
Anne	Calculus

student	*course*	*dept.*
John	Calculus	Chemistry
Martin	x	Botany
Peter	y	Physics

and suppose one wishes to update the database so that all students taking Calculus are in the Botany department. While (John, Calculus, Botany) surely belongs to the updated relation, the problem for Peter is that his department depends on the values of the unknown value y. The update is captured by the following C-relation

student	*course*	*dept.*	*Con*
John	Calculus	Chemistry	
Martin	x	Botany	
Peter	y	Physics	$y \neq$ Calculus
Peter	y	Botany	$y =$ Calculus

Notice that every possible world of this relation contains exactly three tuples.

Consider now the update that Peter's professor also teaches Algebra. Then, Anne's courses in relation *teaches* depend on an unknown null value in a different relation. The update is represented by the following C-relation:

professor	*course*	*Con*
Anne	Calculus	
Anne	Algebra	$y =$ Calculus

The two updates considered were completely specified. Consider now an update so that either Martin or Peter is deleted from the first relation *takes* above. Intuitively, this means that, in each possible world of *REP*(*takes*), one of the tuples can be deleted. Such an update can be represented with the following C-relation:

student	*course*	*dept.*	*Con*
John	Calculus	Chemistry	
Martin	x	Botany	$v \neq$ Martin
Peter	y	Botany	$v \neq$ Peter

in addition to a global condition stating that $v =$ Martin $\vee\, v =$ Peter. It can be verified that, in any possible world, either the second or the third tuple is discarded.

3.4 Summary

In this section, we have introduced mechanisms for defining the manipulation of imperfect databases in a semantically correct way. The different types of imperfect information are studied in detail in the next sections. However, as we will see, manipulating imperfect information is not an easy task. A negative result typical of this complex field is that the simplest ideas do not work very far. For example, the intuitively appealing existential null values do not support a satisfactory definition of the relational join. An existential null value on a finite domain can be represented as an or-set, but or-sets do not permit a correct join either; relations with or-sets represented by variables are the next step, but still unsatisfactory for the algebra. Only conditional relations and general disjunctive databases, fairly more complex than databases with existential null values, are flexible and powerful enough to support a definition of the algebra.

4 Existential values

4.1 Atomic existential values

Existential null values correspond to the situation where the value of an attribute exists, but the specific value is unknown. An existential value is *atomic* when the unknown attribute has exactly one possible value. *Set* existential values are studied in the next section.

We describe now several types of relations having null values. In the following relation *teaches*

professor	*course*	*classroom*
Marc	Databases	AX210
Pierre	x	H1309
Thomas	Mechanics	y
Thomas	Languages	z

variables represent null values, where two different variables may or may not represent the same constant. Thus, the course taught by Pierre in classroom H1309 and the rooms in which Thomas teaches his two courses are unknown.

Relations as the above can be supplemented with global conditions composed of equalities and/or inequalities, as in the following example:

professor	*course*	*classroom*
Marc	Databases	AX210
Pierre	x	H1309
Thomas	Mechanics	y
Thomas	Languages	y

$\boxed{x \neq y \wedge y \neq \text{H3322}}$

The fact that y appears twice means that Thomas teaches both his courses in the same room. The global condition gives additional information about the values of x and y. Notice that $x \neq y$ can be deduced from the database since the domains of *course* and *classroom* are disjoint.

Finally, *conditional relations* (or *C-relations*) are obtained by adding to the relations a column containing local conditions composed of equalities and inequalities. An example is as follows:

professor	*course*	*classroom*	*cond*
Marc	Databases	AX210	
Pierre	x	H1309	
Thomas	Mechanics	y	$x =$ Physics
Thomas	Languages	y	$x \neq$ Physics

$\boxed{x \neq y \wedge y \neq \text{H3322}}$

It represents that Thomas teaches Mechanics if Pierre teaches Physics, otherwise he teaches Languages. Notice that, unlike the previous example, Thomas teaches only one course. *Horn tables* are conditional tables where only restricted types of local and global conditions are allowed [Gra91].

Existential values in relational databases have been an active area of research after their introduction by Codd [Cod75, Cod79]. He studied the introduction of a unique null value and proposed a three-valued logic based on the truth values $\{true, false, unknown\}$ and a so-called null substitution principle. Later, Codd [Cod86, Cod87] generalized the approach to inapplicable null values in a four-valued logic with an additional truth value *inapplicable*. However, as pointed out by various researchers [Gra77, Vas79, Bis81, Bis83], Codd's approach suffers from several drawbacks, for example in the treatment of tautologies and since it does not provide for several unknown values.

An aspect of the existential null values was pointed out in [Lip79, Lip81], who defined information which *surely*, alternatively *possibly*, can be extracted from a database in the presence of unknown values, and proposed a query language containing these modal operators. Notice that such considerations apply in general to all types of incomplete information.

An important step in the understanding of the unknown null values was achieved by Imieliński and Lipsky [IL84]. Unlike Codd, they introduced several unknown values for obtaining a richer modeling power, and they generalized the relational algebra operators. Their most important contribution was the definition of the correctness criteria presented in Section 3 and the methodology for generalizing relational operators. These ideas can be applied to a much wider range of databases than was initially expected. Another major result was to prove that for obtaining all the answers for arbitrary relational queries, the expressive power of conditional relations was needed.

Grahne [Gra91], defining the semantics of incomplete databases as sets of possible worlds, showed that two different lattices can be defined in incomplete databases. He studied query evaluation, integrity constraint satisfaction, and database updates as operations on these lattices. Grahne also studied the manipulation of different types of relations containing unknown null values and, following [AKG91], established the complexity of query evaluation, integrity constraint satisfaction, and database updates in this context.

On the other hand, from a logical standpoint, the information content of a database can be formalized by means of *extended relational theories* [Rei84, Rei86, HPRV89]. In these theories an *extension axiom* is associated with each relation of the database to specify all and the only tuples of constants belonging to the relation.

Extended relational theories also contain a set of *unique name axioms*, specifying which distinct constants represent different objects (for example Marc $\neq$ Pierre). In extended theories, existential null values are represented by constants of the same type as ordinary constants. It is the absence of some unique name axioms that expresses their role to represent the incompleteness of knowledge.

For example, given the previous conditional relation, the extension axiom of predicate *teaches* is as follows:

$$\begin{aligned}\forall x\,\forall y\,\forall z\ \mathit{teaches}(x,y,z) \leftrightarrow\ & (x = \text{Marc} \wedge y = \text{Databases} \wedge z = \text{AX210})\vee\\ & (x = \text{Pierre} \wedge y = \omega_1 \wedge z = \text{H1309})\vee\\ & (\omega_1 = \text{Physics} \rightarrow x = \text{Thomas} \wedge y = \text{Mechanics} \wedge z = \omega_2)\vee\\ & (\omega_1 \neq \text{Physics} \rightarrow x = \text{Thomas} \wedge y = \text{Languages} \wedge z = \omega_2).\end{aligned}$$

Constants ω_1 and ω_2 represent existential values. In addition to unique name axioms like Marc $\neq$ Pierre or Marc $\neq$ Thomas for ordinary constants, the extended theory contains the following unique name axioms for the existential null values: $\omega_2 \neq$ H3322, and $\omega_1 \neq \omega_2$.

Reiter gave an algorithm to answer queries in extended relational theories and proved that his algorithm is *sound*, in the sense that all the answers retrieved are correct, but is *incomplete*, since some correct answers are not obtained by the algorithm. In [DF88] Reiter's approach is extended for theories containing constraints on the existential null values.

With respect to dependencies in relations containing existential values, after the early works of [Vas80, IL83], Grahne [Gra84, Gra91] introduced the notion of general conditions and proved that conditional relations with general conditions are powerful enough for enforcing a set of functional and full join dependencies. Query evaluation and dependency satisfaction were also studied in [Ler86a, Ler86b] with Reiter's logical framework.

Updates in relations containing existential values were studied by Abiteboul and Grahne [AG85, Gra91]. They defined the semantics of updates and the different set-theoretic operators that can be used for updating a database containing existential values, as presented in Section 3.3. In addition they highlighted the ability of different relations to support update operations. In particular they showed that conditional relations fully support all updates operations while other types of relations are well suited for only some types of update operations. Using a logical approach, in [Win86, Win88b, Win88a] are studied updates with null values and disjunctive information in restricted theories of first-order logic with equality.

Figure 8 summarizes the successive generalizations needed to support correct manipulations of null values, together with the problems that justify the next richer version.

extended relations	*problem*
relations with one null value	tautologies
relations with marked null values	complete algebra and integrity constraints
conditional relations with null values	updates
conditional relations with null values and global conditions	complexity of manipulation

Figure 8: Enrichment of expressive power for null values.

4.2 Set existential values

Consider a relation *optional_course*, stating the optional courses taken by students, given by

student	*optional_course*
Dominique	Languages
Anne–Marie	$\{x\}$
Martin	$\{y\}$
Jean–Pierre	z

Databases $\notin \{x\}$

Here, $\{x\}$ and $\{y\}$ denote set existential values, meaning that Anne–Marie and Martin take *at least one* course while Jean–Pierre takes *exactly* one. In addition, it is known that Databases is not among the courses of Anne–Marie.

Set existential values have been studied by Hulin [Hul90]. Representing them in first-order logic leads to *two-sorted extended theories*, with a sort $\mathcal{C}$ for ordinary constants and atomic existential values, and a sort $\mathcal{D}$ for sets. Variables are sorted and each n-ary relational predicate is defined as being of sort $(\mathcal{C}, \ldots, \mathcal{C})$. In addition, the binary predicate $\in$ (belongs), of sort $(\mathcal{C}, \mathcal{D})$, is included.

In the previous example, variables $\{x\}$ and $\{y\}$ are represented by constants ϵ_1 and ϵ_2 of sort $\mathcal{D}$. To specify that these constants do not represent the empty set, the *non-emptiness axioms* $\exists x\ x \in \epsilon_1$, and $\exists x\ x \in \epsilon_2$ are introduced in the theory, where variable x is of sort $\mathcal{C}$.

Another type of information that can be introduced in the theory is that two sets are disjoint. This is represented with the following *disjointness axiom*:

$$\neg(\exists x\ x \in \epsilon_1 \wedge x \in \epsilon_2) \ \text{ or, equivalently, } \ \forall x\ x \notin \epsilon_1 \vee x \notin \epsilon_2.$$

Finally, the extension axiom for predicate *optional_course* is as follows:

$$\begin{aligned} \forall x\ \forall y\ & \mathit{optional\_course}(x, y) \leftrightarrow (x = \text{Dominique} \wedge y = \text{Languages}) \vee \\ & (x = \text{Anne–Marie} \wedge y \in \epsilon_1) \vee (x = \text{Martin} \wedge y \in \epsilon_2) \vee \\ & (x = \text{Jean–Pierre} \wedge y = \omega). \end{aligned}$$

In addition to unique name axioms, the *non-membership axiom* Databases $\notin \epsilon_1$ is needed.

As remarked by Imieliński [Imi86], incomplete information is often introduced in relational databases by updates on views. Set existential values are important in this context since they allow to reflect more accurately our knowledge of the world.

For example, given predicates *teaches*(*professor*,*course*) and *takes*(*student*,*course*), a view v defined by

$$\forall x\ \forall y\ v(x, y) \leftrightarrow (\exists z\ \mathit{teaches}(x, z) \wedge \mathit{takes}(y, z))$$

may be used for obtaining the pairs (professor,student) for professors giving at least one course to a student. To add to view v the pair (Peter,Anne), stating that Peter teaches one (or several) courses to Anne, the classical approach is to use an existential value, or a Skolem constant ω, and assert the facts *teaches*(Peter, ω) and *takes*(Anne, ω).

Thus, existential values are often the result of Skolemizing existentially quantified conjunctive views. Yet Skolemization preserves satisfiability but not derivability, i.e., Skolemizing views may make true formulas that were false before. Hulin [Hul90] gives several examples of Skolemized theories that lead to anomalous answers when evaluating queries. Such anomalous answers, derived from Reiter's axiomatization of null values, are not discovered by his evaluation algorithm because of its incompleteness. However, other anomalous answers are derived by his algorithm which cannot be justified in the non-Skolemized theory. Hulin showed that the anomalous answers generated with the extended relational theories disappear with two-sorted relational theories.

4.3 Or-sets and restricted cardinality sets

Or-sets (or disjunctive sets) [Lip79, Sow84, Yeo87, IV89, Imi89] generalize existential null values. An existential value represents an attribute whose actual value is in a database domain. An or-set represents an attribute whose value is an explicit, smaller set.

When an or-set appears in a relation tuple, the actual value of the attribute is one element of the or-set. An ordinary atomic value can be viewed as an or-set with the value as single element. An existential value corresponds to an or-set containing the entire attribute domain. Or-sets can also contain a special value $\emptyset$ to represent the possibility of there being no value.

In the following relation *teaches*(*professor*,*course*), the second tuple represents that Thomas teaches one of the courses of the set $\{\text{Physics}, \text{Chemistry}, \text{Biology}\}$. The third tuple represents that either Susan teaches Algebra, or that Martha does, or that the course is not taught.

professor	*course*
Marc	Databases
Thomas	$\{\text{Physics}, \text{Chemistry}, \text{Biology}\}$
$\{\text{Susan}, \text{Martha}, \emptyset\}$	Algebra

Or-sets are not powerful enough to represent some useful varieties of incomplete information. For this reason, [MY87] proposed the *generalized or-sets*, in which the actual value is understood to be some non-empty subset of the stored set. If the actual value may be empty, the special value $\emptyset$ is included in the stored set. For example, interpreting in the relation above $\{\text{Susan}, \text{Martha}, \emptyset\}$ as a generalized or-set means that Algebra may be taught by either Susan or Martha, or both, or may not be offered.

[MY87] also studied the combination of or-sets, generalized or-sets, as well as and-sets in the relations. An *and-set* in an attribute A of a tuple t represents that the actual value of the attribute is precisely the specified set of values. And-sets are those used in the context of nested or non-first normal form relations, which are studied in Section 9.

In addition, another kind of sets are defined in [MG88], the *restricted cardinality sets* (*rc-sets*). An rc-set $S_{p,q}$ defines a set S of values from the attribute's domain such that $0 \leq p \leq q \leq n$ with n is the number of elements in S. An rc-set is interpreted as meaning that the actual value of the attribute is some subset s of the stored set of candidates S such that $p \leq \text{card}(s) \leq q$. If $q = 0$, the actual value is the empty set.

For example, a tuple (Thomas, $\{\text{Physics}, \text{Chemistry}, \text{Biology}\}_{1,2}$) in a relation *teaches* means that Thomas will teach between one and two of the courses represented in the set. Notice that rc-sets allow to represent ordinary atomic values, or-sets, generalized or-sets, and and-sets.

Generalized operators on relations containing rc-sets as defined in [MG88] suffer from one important drawback. As shown in [Imi89], it is not possible to define algebraic operators over relations having or-sets (and, consequently, having generalized or-sets and rc-sets) that satisfy the requirements stated in Section 3.1 for a correct generalization of the relational algebra.

For this reason, Imieliński and Vadaparty [IV89, Imi89] represented or-sets by variables, in what they called *or-relations*. These relations contain variables, with a set of values associated to each variable. An example of or-relation is as follows:

professor	*course*	*classroom*
Marc	Databases	AX210
Pierre	x	H1309
Thomas	Mechanics	y

where x and y represent or-sets with dom(x) = {Physics, Materials} and dom(y) = {H1309, H1310}. In this example, Pierre teaches either Physics or Materials, and Thomas teaches Mechanics either in room H1309 or in room H1310.

However, as shown by Imieliński, the expressive power of relations with or-sets (even represented with variables) is not sufficient to support both projection and join. In fact, to support both operators, or-sets must be extended to general disjunctions like $\mathit{teaches}(\text{Marc}, \text{Calculus}) \vee \mathit{teaches}(\text{Anne}, \text{Algebra})$. This leads to disjunctive databases presented in Section 11. Furthermore, since Imieliński and Vadaparty used the true sets for stating the semantics of queries, they did not considered "possible" or "maybe" information which can be derived from the database. Maybe information is studied in Section 10.

Similar to Figure 8, Figure 9 summarizes the successive generalizations of or-sets and the problem with each one justifying the definition of the next one.

extended relations	*problem*
relations with or-sets	unable to represent that the attribute takes no value
relations with extended or-sets having a special value $\emptyset$	unable to represent that two or-sets are the same
relations with extended or-sets represented by variables	unable to represent or-sets over several attributes
disjunctive databases	definition of full algebra

Figure 9: Enrichment of expressive power for or-sets.

5 Inexistent values

Inexistent null values, denoted by $\perp$, allow to represent that an attribute is inapplicable. For example, consider the following relation $\mathit{takes}(\mathit{student}, \mathit{department}, \mathit{course})$

student	*department*	*course*
Pierre	Physics	Calculus
Jean	$\perp$	Probability

The second tuple represents that Jean takes Probability without being registered in any department.

Representing inexistent null values in mathematical logic is not straightforward. In particular, the following are incorrect representations:

$$\neg(\exists x)\mathit{takes}(\text{Jean}, x, \text{Probability}) \text{ or } (\forall x)\neg\mathit{takes}(\text{Jean}, x, \text{Probability})$$

because they prevent obtaining Probability as an answer to the query "in which courses is Jean registered?".

The same problem arises when representing inexistent values as "empty" set existential values in a two-sorted theory. As stated by Hulin [Hul90], inapplicable null values probably require three-valued and two-sorted logic theories.

Another solution is to represent inexistent values with a special constant $\perp$ of the same type as other constants, leaving their interpretation to extra-logical notions. The extension axiom of predicate of predicate *takes* becomes

$$\begin{aligned} \forall x\ \forall y\ \forall z\ \mathit{takes}(x, y, z) \leftrightarrow {} & (x = \text{Pierre} \wedge y = \text{Physics} \wedge z = \text{Calculus}) \vee \\ & (x = \text{Jean} \wedge y = \perp \wedge z = \text{Probability}). \end{aligned}$$

To respect the meaning of the inexistent null value, the following constraint must be introduced in the theory

$$\forall x\ \textit{takes}(\text{Jean}, x, \text{Probability}) \leftrightarrow x = \perp .$$

As shown by this constraint, it is necessary to introduce in the database a different constant for each inexistent value. In fact, representing that "Marie is registered in Algebra without being registered in any department" by adding the tuple (Marie, $\perp$, Algebra) would result in considering both persons as belonging to the same department, which is not correct.

Therefore, the constants of the database must be divided in two subsets: a set C containing the ordinary constants and a set $\perp = \{\perp_1, \ldots, \perp_n\}$ that contains all the inexistent values appearing in the database. Of course, appropriate unique name axioms have to be introduced in the theory for stating that inexistent values are different from each other and from every other constant.

Inapplicable values were studied by various researchers. As already mentioned, Codd proposed a four-valued logic for the manipulation of existential as well as inapplicable null values. Vassiliou [Vas79] used a denotational semantics approach. In addition, inexistent null values were studied by Zaniolo [Zan84] and by Gottlob and Zicari [GZ88] in conjunction with other kinds of null values presented in later sections.

6 Open databases and null values

6.1 Closure assumptions

With the *Closed World Assumption* (CWA), all the information not explicitly represented in the database is supposed false. For a relation *teaches*(*professor*, *course*, *department*), the CWA amounts to saying that there are no other professors, no other courses, and no other departments than those appearing in the relation.

With the *Open World Assumption* (OWA), the above assumption does not hold since there may be other tuples in the relation. In addition, with the OWA, the negative information must be explicitly stated, contrary to closed databases. For instance, in the following relation

	professor	*course*	*department*
positive component	Pierre	Calculus	Physics
	Marc	Calculus	Chemistry
negative component	Marc	Algebra	Physics

it is represented that "Marc does not teach Algebra in the Physics department".

This relation is represented by the following formulas:

$$\begin{aligned}&\forall x\, \forall y\, \forall z\ \textit{teaches}(x, y, z) \leftarrow (x = \text{Pierre} \wedge y = \text{Calculus} \wedge z = \text{Physics}) \vee \\ &\qquad (x = \text{Marc} \wedge y = \text{Calculus} \wedge z = \text{Chemistry}),\\ &\neg \textit{teaches}(\text{Marc,Algebra,Physics}).\end{aligned}$$

Open and closed databases have some properties in common. For instance, a relational expression not containing the difference operator yields the same result for both open or closed databases. The difference operator implicitly assumes the Closed World Assumption. Thus, a difference operator taking into account the negative component of relations is needed.

Another difference between open and closed databases pointed out by Reiter [Rei90] deals with integrity constraint checking when constraints are expressed as formulas in epistemic logic. Much work remains to be done to establish relationships between open and closed databases, and to define query answering, integrity checking, and updates in open databases.

6.2 Universal and default values

Under the OWA, two other categories of null values naturally arise: the *universal* and the *default values*.

Suppose that "Jean teaches Biology in all departments", and "Brigitte teaches no course in the Chemistry department" must be represented in relation *teaches* of the previous section.

If the relation is interpreted under the CWA, the first fact can be represented by adding to the relation a set of tuples of the form (Jean, dep_i, Biology), one for each department of the database. In addition, if the information present in the relation is consistent with the second fact, the CWA implies that Brigitte teaches no course in the Chemistry department.

The situation is completely different under the OWA. Since all the departments are not necessarily known, the first fact cannot be represented by introducing tuples of the form (Jean, dep_i, Biology). The situation is still different when representing the second fact, because the negative information must be explicitly introduced in open databases. Therefore, the previous facts must be represented with an *universal null value*, denoted by $\forall$, as follows:

	professor	*course*	*department*
positive component	Pierre	Calculus	Physics
	Marc	Calculus	Chemistry
	Jean	Biology	$\forall$
negative component	Marc	Algebra	Physics
	Brigitte	$\forall$	Chemistry

Under the OWA, this relation is represented as follows:

$$\begin{aligned}\forall x\, \forall y\, \forall z\; teaches(x, y, z) \leftarrow\ & (x = \text{Pierre} \land y = \text{Calculus} \land z = \text{Physics}) \lor \\ & (x = \text{Marc} \land y = \text{Calculus} \land z = \text{Chemistry}) \lor \\ & (x = \text{Jean} \land y = \text{Biology}),\end{aligned}$$

in addition to the formulas

$$\neg teaches(\text{Marc,Algebra,Physics}), \qquad \forall x\ \neg teaches(\text{Brigitte}, x, \text{Chemistry}).$$

Although universal values were introduced by Biskup in [Bis83], to our knowledge, no work has formally studied the manipulation of such values in relational databases.

Default null values are another category of nulls needed when the OWA is applied to a database. Suppose that the income of university professors is represented by the following relation:

year	*professor*	*income*
1990	Jean	19,000
1990	Marc	20,000

Without more information, suppose that it is estimated by default that every professor not explicitly represented has an income of 22,000 for the year 1990.

Under the CWA, this default is represented by a tuple for each professor different from Marc. On the contrary, under the OWA, since not all professors are known, we need to introduce in the relation the tuple $\langle 1990, *, 22,000\rangle$, where $*$ denotes a default null value.

Thus, the relation can be represented in first-order logic by the following extension axiom:

$$\begin{aligned}\forall x, y, z\ income(x, y, z) \leftarrow\ & (x = 1990 \land y = \text{Jean} \land z = 19,000) \lor \\ & (x = 1990 \land y = \text{Marc} \land z = 20,000) \lor \\ & (x = 1990 \land y \neq \text{Jean} \land y \neq \text{Marc} \land z = 22,000),\end{aligned}$$

in addition to the constraint

$$(\forall x, y, z)(income(x, y, z) \wedge x = 1990 \wedge y \neq \text{Jean} \wedge y \neq \text{Marc} \rightarrow z = 22,000).$$

It is also possible to represent default values in first-order logic with a set-theoretical notation. The default information can be represented by defining a set $\epsilon_1 = \{\text{Jean,Marc}\}$, by replacing the last disjunct in the extension axiom of *income* by the formula

$$x = 1990 \wedge y \notin \epsilon_1 \wedge z = 22,000,$$

and by replacing the constraint with:

$$\forall x, y, z\ income(x, y, z) \wedge x = 1990 \wedge y \notin \epsilon_1 \rightarrow z = 22,000.$$

The introduction of default information in databases is very important. Because many applications have to deal with great number of unknown parameters, for example in economical environments, it is necessary to make estimations and decisions based on these estimations. Default information provides an extra capability for handling uncertainty, because it states "common" or "average" values that are likely to be true in the absence of evidence proving the contrary.

Nevertheless, very few work has been realized on defaults in databases. Some basic issues are discussed in [Dat86] for relational databases and in [KS91] for deductive databases. Notice that systems like SYNTEL [RRHD88], a programming system used for financial risk assessment, includes default information, as well as unknown and uncertain values. For manipulating such information, SYNTEL includes an ad-hoc *default join* operator.

Research in this context has to be based on the definition of precise semantics for default values. All the work on default logics (e.g. [Rei80]) is relevant.

6.3 Open values

Thus far, we have considered databases that are totally open or totally closed. Another possibility is to use a mixed approach, where each relation of the database is interpreted either under the Open or under the Closed World Assumption. This is the approach followed by Gelfond and Przymusinska for the definition of the Careful Closure Assumption [GPP86].

The Locally Open World Assumption [GZ88] enables to partially or entirely relax the closure assumption inside a relation. An *open null value* expresses that an attribute of a tuple must be interpreted under the Open World Assumption: the attribute value may not exist, it may take exactly one value, or it may take several values. In other words, no assumption is made about the attribute's value. In the following relation, the open null value Ω expresses that the relation is closed, except for attribute *student*:

student	*department*	*course*
Pierre	Biology	Calculus
Ω	Chemistry	Algebra

The following facts can be inferred: (1) the only departments are Biology and Chemistry; (2) the only courses are Calculus and Algebra; (3) the only student in Biology who takes Calculus is Pierre; and (4) there may be zero, one, or several students registered in the Chemistry department taking Algebra. No other fact is true in this relation.

Suppose now that $(\Omega_1, \Omega_2, \Omega_3)$ replaces the second tuple of the above relation. The open values express that the relation is interpreted under the Open World Assumption. All the attributes of the second tuple are open, meaning that that Pierre is registered in Biology and takes Calculus, but nothing else is known and, in particular, no negated facts.

The no-information nulls of Zaniolo [Zan84] can be seen as open null values taking at most one value from the attribute domain. As remarked by Zaniolo, open null values are less informative than existential (either atomic or set) and inexistent null values. They can approximate both when it is not known whether or not values exist for the attribute.

The semantics for relations containing open null values can be specified as follows. Consider the first relation *takes* above and suppose that the domain for *student* is {Pierre, Anne}. Then, the relation represents the following set of possible worlds

$M_1 = \{$(Pierre,Biology,Calculus),($\perp$,Chemistry,Algebra)$\}$
$M_2 = \{$(Pierre,Biology,Calculus),(Pierre,Chemistry,Algebra$\}$
$M_3 = \{$(Pierre,Biology,Calculus),(Anne,Chemistry,Algebra$\}$
$M_4 = \{$(Pierre,Biology,Calculus),(Pierre,Chemistry,Algebra), (Anne,Chemistry,Algebra)$\}$

Open null values can be modeled in first-order logic with two-sorted relational theories, where each open value is represented by a set. As noted by Hulin [Hul90], the only difference between the set existential values and the open values is that the open null values do not have non-emptiness axioms. For example, the above relation is represented in a two-sorted relational theory with the axiom

$$\begin{aligned}\forall x\, \forall y\, \forall z\; takes(x,y,z) \leftrightarrow (&x = \text{Pierre} \wedge y = \text{Biology} \wedge z = \text{Calculus}) \vee \\ &(x \in \epsilon_1 \wedge y = \text{Chemistry} \wedge z = \text{Algebra}).\end{aligned}$$

Gottlob and Zicari [GZ88] studied the manipulation of open values in conjunction with existential and inexistent values. This is presented in the next section.

7 Combination of null values

Consider the following relation *takes*:

	student	*department*	*course*
positive component	Pierre	Botany	Calculus
	Ω_1	Ω_2	Calculus
	Marie	Physics	$\perp$
	Jean	Physics	$\{x\}$
	Paul	Biology	Ω_3
negative component	$\forall$	Chemistry	Calculus
	Paul	Biology	Statistics
	$*$	Botany	Calculus

Interpreted under the Locally Open World Assumption, it models the following facts:

- Pierre is registered in Botany and takes Calculus;
- students, explicitly known or not, registered in departments, explicitly known or not, may take Calculus;
- Marie is registered in Physics and takes no course;
- Jean is registered in Physics and takes one or several courses, but it is not known which ones;
- Paul may take courses in Biology, but Statistics is not among them;

- no student registered in Chemistry takes Calculus;
- no student registered in Botany, except Pierre, takes Calculus.

The relation is represented in first-order logic by the following extension axiom:

$$\begin{aligned}\forall x\,\forall y\,\forall z\; \mathit{takes}(x,y,z) \leftrightarrow{}& (x = \text{Pierre} \wedge y = \text{Botany} \wedge z = \text{Calculus}) \vee \\ & (x \in \epsilon_1 \wedge y \in \epsilon_2 \wedge z = \text{Calculus}) \vee (x = \text{Marie} \wedge y = \text{Physics} \wedge z = \perp) \vee \\ & (x = \text{Jean} \wedge y = \text{Physics} \wedge z \in \epsilon_3) \vee (x = \text{Paul} \wedge y = \text{Biology} \wedge z \in \epsilon_4),\end{aligned}$$

and the following axioms:

$$\begin{array}{cc} \exists x\; x \in \epsilon_3, & \forall x\; \mathit{takes}(\text{Marie}, \text{Physics}, x) \leftrightarrow x = \perp, \\ \forall x\; x = \text{Statistics} \rightarrow x \notin \epsilon_4, & \forall x\; \neg \mathit{takes}(x, \text{Chemistry}, \text{Calculus}), \\ \multicolumn{2}{c}{\forall x\; x \notin \{\mathit{Pierre}\} \rightarrow \neg \mathit{takes}(x, \text{Botany,Calculus}).} \end{array}$$

The semantics of relations combining several types of nulls can be derived from the possible worlds semantics of the individual nulls. However, as pointed out by Gottlob and Zicari [GZ88], combining nulls raises the question of consistency and redundancy.

If $\mathbf{t} = (\text{Pierre,Botany,}\perp)$ is added to the relation *takes* above, it becomes inconsistent, since $\mathbf{t}$ contradicts the first tuple stating that Pierre is registered in Botany and takes Calculus.

Consider now redundancy. Intuitively, a tuple $\mathbf{t}$ is redundant in a relation R iff $\mathbf{t}$ can be removed without changing the semantics of the relation, that is, iff the possible worlds of R are the same as those of $R - \{\mathbf{t}\}$. For instance, $(\text{Jean}, \text{Physics}, y)$, where y is an existential value is redundant, since it does not add any information to $(\text{Jean}, \text{Physics}, \{x\})$. Similarly, $(\text{Marie}, \text{Physics}, \Omega)$ is redundant with respect to $(\text{Marie}, \text{Physics}, \perp)$, the second tuple constraining Marie to take no course.

The relational operators have to be generalized for manipulating the different types of null values. In addition, integrity checking and database updates must also take into account the semantics expressed by all types of nulls. Much work remains to be done in this context.

8 Universal relation databases and null values

The *Universal Relation model* (UR model) [Ull82, Ull83, MU83, MUV84] allows the user to view a relational database as if it were composed of a single relation. Thus the UR model provides logical data independence by freeing the user from navigating amongst the relations in a given database. The most fundamental assumption of the model is that there is an universal set of attributes $U = \{A_1, \ldots, A_k\}$ for the application being modelled, each attribute having an unique meaning. It is also assumed that every set of attributes $X \subseteq U$ has a basic relationship, that is, in the user's mind there is an unique semantic relationship amongst the attributes of X. The theory of the universal relation model was firmly established with the introduction of the *weak instance approach* [Hon82].

Given a database schema $\boldsymbol{\mathcal{R}} = \langle \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$, each relation scheme $\mathcal{R}_i$ is a subset of U. Constants and variables are allowed in tuples. A *tableau* on a relation scheme $\mathcal{R}$ is a set of tuples on $\mathcal{R}$, while a *relation* on $\mathcal{R}$ is a set of total tuples (i.e. tuples without variables) on $\mathcal{R}$.

The underlying data structure of the UR model is the *representative instance*, which is suitable for storing all the data of the database in a single relation. The representative instance is a tableau over the universe U of attributes, defined for each database state $\mathbf{r}$ as follows. First, a tableau over U (called the *state tableau* for $\mathbf{r}$ and denoted by $T_{\mathbf{r}}$) is the union of all the relations in $\mathbf{r}$ extended to U by means of variables, initially different from one another. Then, a chase procedure is applied to $T_{\mathbf{r}}$ to satisfy the dependencies specified for the database.

Consider the following database with the functional dependency *professor* $\rightarrow$ *department*:

professor	*course*
John	Algebra
Bob	Calculus

professor	*department*
John	Physics
Tom	Chemistry

The corresponding representative instance is as follows

professor	*course*	*department*
John	Algebra	Physics
Bob	Calculus	v_1
John	v_2	Physics
Tom	v_3	Chemistry

Given a state **r** for a database scheme $\boldsymbol{\mathcal{R}} = \langle R_1, \ldots, R_n \rangle$, a relation u over the universe U is a *containing instance* for **r** if its projections on the relation schemes of $\boldsymbol{\mathcal{R}}$ contain the corresponding relations of **r**; formally $\pi_{R_i}(u) \supseteq r_i$, for $i = 1, \ldots, n$. Given a set of dependencies F, a relation w is a *weak instance* for **r** with respect to F if w is a containing instance for **r** and w satisfies F.

Then the database **r** is in general an incomplete description of a weak instance w. There are infinitely many weak instances for **r** that satisfy F (or a very large number if all attribute domains are finite). Thus, the only facts that can be deduced from the UR model, given the relations in the database **r** are those that hold in all weak instances. Note that weak instances may be obtained by replacing variables by constants of the appropriate domain in the representative instance.

The weak instance model was formalized in [MUV84]. However, as shown in [AB90], the weak instance model has one drawback: it considers all tuples for which incomplete information is available as existentially quantified with respect to the missing values. This fact is apparent from the way in which weak and representative instances are built, and especially, from the definition of the logical theories associated with them. In the example above, the usual definition of the weak instance model assumes that, for each professor, there is exactly one department and at least one course. For this reason, the values v_i stand for actual, unknown values. Therefore, it would be conceptually wrong to use this database to model an application where professors do not necessarily teach courses.

An alternative definition of the weak instance model is presented in [AB90]. The main difference is that, in weak instances, inexistent null values are allowed together with constants from the domain. Thus, a weak instance for a state is an extended relation (i.e., a set of tuples with constants and inexistent values) that satisfies the constraints and contains the base relations in its projections. Two possible weak instances for the database above are the following

professor	*course*	*department*
John	Algebra	Physics
Bob	Calculus	Biology
Tom	Calculus	Chemistry

professor	*course*	*department*
John	Algebra	Physics
Bob	Calculus	$\perp$
Tom	$\perp$	Chemistry

The first one is a weak instance according to the classic definition, whereas the second is not, since it contains inexistent values. Since each weak instance represents a possible world compatible with the data in the database, with the extended definition, the possibility that Tom teaches a course is open. For this reason, under the extended definition, variables in the representative instance are no-information or open null values, depending on the functional dependencies of the database.

The alternative definition of the weak instance model can be formalized as follows [AB90]. A first-order language is defined with a predicate symbol for each nonempty subset of the universe U. A first-order theory with four kinds of axioms is associated with every database state $\mathbf{r} = \langle r_1, \ldots, r_n \rangle$.

(1) A set DB of atomic sentences describing the relations in the database: for every relation $r \in \mathbf{r}$ and every tuple $t \in r$, there is the sentence $R(t)$. In the database discussed above, there are four sentences: $PC(\text{John}, \text{Algebra})$, $PC(\text{Bob}, \text{Calculus})$, $PD(\text{John}, \text{Physics})$, and $PD(\text{Tom}, \text{Chemistry})$

(2) A set CON of sentences stating that, for every tuple t satisfying a predicate X, for every subset Y of X, the subtuple $t[Y]$ satisfies the predicate associated with Y. In the example, there are 12 sentences:

$$\begin{array}{ll}
\forall p\, \forall c\, \forall d\; PCD(p,c,d) \rightarrow PC(p,c) & \forall p\, \forall c\; PC(p,c) \rightarrow P(p) \\
\forall p\, \forall c\, \forall d\; PCD(p,c,d) \rightarrow PD(p,d) & \forall p\, \forall c\; PC(p,c) \rightarrow C(c) \\
\vdots & \vdots
\end{array}$$

(3) A set DIS of sentences stating that all constants are distinct.

(4) A set DEP of first-order sentences representing the set F of dependencies. The functional dependency *professor* $\rightarrow$ *department* of the example above is represented by

$$\begin{array}{c}
\forall p\, \forall d_1\, \forall d_2\; PC(p,d_1) \wedge PC(p,d_2) \rightarrow d_1 = d_2, \\
\forall p\, \forall c\, \forall d\; PD(p,d) \wedge PC(p,c) \rightarrow PCD(p,c,d).
\end{array}$$

The first sentence enforces the dependencies with respect to non-null values. The second sentence formalizes the requirement that if tuples t, t' take the same values on P and $t[D]$ is not null, then $t'[D]$ is not null either, and it is equal to $t[D]$.

The results in [AB90] showed that the main properties of the original weak instance model are preserved. Several approaches to updating weak instances are studied in [AT89].

9 Null values in nested relational databases

In recent years there has been an increasing interest in *non-first normal form* or *nested relations* where attributes can take values which are sets, ordered lists, or relations.

Consider for example the following nested relation.

student	*dept*	*(course*	*(exam)**	*(project)*)**
		course	*(exam)**	*(project)**
			exam	*project*
Iris	CS	Databases	mid final	1NF
		Programming	final	ADA
Noam	languages	French	mid	Cyrano
		German	mid	Faust
		English	final	Othello

In this relation each student is registered in one department and takes a set of courses. For each course there is a set of exams and a set of projects.

The nested relational model, in addition to the usual algebraic operators, has two restructuring operators: *nest* (ν) and *unnest* (μ). Intuitively, nesting transform a nested relation into one which is "more deeply" nested, while unnesting flattens one level of a nested relation. The operator unnest* ($\mu^\star$) transforms a nested relation $r^\star$ into a flat relation r.

Since in the nested relational model, an attribute can be a relation as well, that is, a set of tuples, the domain of a complex attribute includes the empty set as a legal value. Suppose that in the nested relation of the example above we want to represent the tuple

David	EE	VLSI	mid final	$\emptyset$

where the attribute (*project*)* for the course VLSI takes as value the empty set, meaning that there is no project for the course. The empty set is in a sense a particular type of null value: indeed, unnesting a relation that contains an empty set introduces nulls into the resulting first normal form relation. For the tuple above, applying the operator μ^* causes no loss of information if explicit null values are allowed in the relation as follows

student	*dept*	*course*	*exam*	*project*
David	EE	VLSI	mid	$\perp$
David	EE	VLSI	final	$\perp$

Incorporating the empty set into nested relations has been a controversial issue, as it is not clear what the result of unnesting the empty set should be. Several authors state that unnesting the empty set produces "undefined value" and "unnatural value". In [AB86], when the empty set is unnested the resulting tuples are removed; however, such an approach is unsatisfactory since information is lost. Several researchers have assigned the *inexistent* interpretation of null values to the empty set [Mak77, AB86, SS86, GZ88]. Others [RKS89], in the context of the OWA, considered the semantics of the empty set to be the "no information" null.

The need for nulls is even more critical in a nested database than in a classical relational database. Since nested relations allows to represent multiple relationships in a single nested relation, we must also deal with the fact that one or more of those relationships may be unknown or inexistent at some time. Notice that similar considerations apply when considering the database under the universal relation model which is discussed in Section 8.

In order to ascertain if an extended algebra for nested relations is reasonable, the notion of *faithfulness* and *preciseness* are used [Mai83, RKS89]. An extended operator is *faithful* if it gives the same result on flat relations as the corresponding standard operator. For *preciseness*, let γ be a classical operator, $\bar{\gamma}$ be the operator for nested relations, and let α be another operator. Then $\bar{\gamma}$ is said to be a precise generalization of γ relative to α if one of the following conditions holds

$$\alpha(\bar{\gamma}(r)) = \gamma(\alpha(r)) \text{ if } \gamma \text{ is an unary operator;} \tag{9.1}$$

$$\alpha(r_1 \bar{\gamma} r_2) = \alpha(r_1)\gamma\alpha(r_2) \text{ if } \gamma \text{ is a binary operator;} \tag{9.2}$$

If α is taken to be the μ^* operator, the above conditions say that unnesting* the result of $\bar{\gamma}$ yields the flat relation obtained by applying the classical operator γ to the argument relations previously flattened.

However, for some choices of α, not all relational operators have a precise generalization. In this case, it is considered a weaker notion of an *adequate* and *restricted* generalization which captures $\gamma(\alpha(r))$ or $\alpha(r_1)\gamma\alpha(r_2)$ and as little extra as is possible.

Now consider the introduction of null values into the nested relations. As for classical relations, it is necessary to determine semantically the sets of possible worlds represented by a nested relation. For this reason a mapping *REP** has to be defined for associating to a nested relation containing nulls r^* the set of nested relations without nulls it represents.

When generalizing the algebra for nested relations with null values, there are two possible ways of establishing if the operators are reasonable. The first one, which is used for example in [RKS89], is equivalent to the strong correctness criterion for relational databases given in Section 3.1. It is stated by conditions (9.1)–(9.2) when α is taken to be the REP^* operator. As in the relational case, not all operators have a precise generalization relative to REP^* and the weaker notions of adequate and restricted are needed.

The second way of establishing if the operators are reasonable, which is used in Levene [Lev92], is based on the definitions of the operators γ for flat relations with nulls. Therefore, taking α as the μ^* operator, and $\bar{\gamma}$ as the operator on nested relations with null values, (9.1)–(9.2) state that $\bar{\gamma}$ correctly extends the operator γ. Levene uses Zaniolo's operators for flat relations with null values [Zan84].

The study of null values in nested relations has been based on the work of Zaniolo [Zan84], which used existential, inexistent, and no information null values. However, Zaniolo considered unmarked nulls and thus the equality relationship is modified so that two null values or a constant and a null value are not equated. This approach is not always adequate for representing incomplete information, especially in the presence of dependencies. Therefore, there is a need to extend the results of [RKS89] and [Lev92] for marked null values.

10 Maybe tuples

One of the extensions of the relational model proposed by Codd [Cod79, Cod86, Cod87] to manipulate existential null values was the *maybe tuples*. This approach was formalized by Biskup [Bis83].

Maybe tuples represent information that is possibly true, like "it is possible that Jean takes Calculus". To represent maybe tuples in a relational database, each relation must be composed of a *sure component* and a *maybe component*. Consider for example relation *takes* as follows:

	student	*course*
sure component	Pierre	Calculus
	Marc	Algebra
maybe component	Jean	Calculus

The third tuple is a maybe tuple: it represents an assertion that may or may not hold. Under the Closed World Assumption, relation *takes* represents the following set of possible worlds, where in each situation every tuple is certain.

$$\begin{aligned} M_1 &= \{(\text{Pierre,Calculus}),(\text{Marc,Algebra})\} \\ M_2 &= \{(\text{Pierre,Calculus}),(\text{Marc,Algebra}),(\text{Jean,Calculus})\} \end{aligned}$$

The relation is expressed in first-order logic as follows:

$$\begin{aligned} &\forall x\, \forall y\; takes(x,y) \rightarrow (x = \text{Pierre} \wedge y = \text{Calculus}) \vee \\ &\qquad (x = \text{Marc} \wedge y = \text{Algebra}) \vee (x = \text{Jean} \wedge y = \text{Calculus}), \\ &\forall x\, \forall y (x = \text{Pierre} \wedge y = \text{Calculus}) \vee \\ &\qquad (x = \text{Marc} \wedge y = \text{Algebra}) \rightarrow takes(x,y). \end{aligned}$$

Straightforward generalizations of relational operators were proposed by several researchers (e.g., [Bis81, Bis83]). However, to our knowledge, no study was done on integrity constraint checking and updates for databases containing maybe information.

11 Disjunctive Databases

Multiple efforts have been made since the 1970's to manipulate disjunctive information, especially in recent years. We cover here disjunctive information in relational databases. Since the fields of disjunctive logic programming and disjunctive deductive databases goes beyond the scope of this paper, we refer for example to [Min89, FM92].

To introduce general disjunctive facts in the relations the definition of tuple has to be generalized. Given a relation scheme $\mathcal{R}(A_1, \ldots, A_n)$, tuples of the form $\mathbf{t} = \mathbf{t}_1 \vee \ldots \vee \mathbf{t}_m$, where each $\mathbf{t}_i \in \text{dom}(A_1) \times \ldots \times \text{dom}(A_n)$, are called *disjunctive tuples*. Classical tuples are then called *definite tuples*. An example of a disjunctive relation *takes* is given below.

(*student, course*)
(Anne,Calculus)
(Marc,Algebra)∨(Marc,Physics)
(Paul,Algebra)∨(Martha,Algebra)

The first tuple is a definite tuple and the other two are disjunctive tuples.

Disjunctive information is closely related to maybe information. A first idea is to represent a disjunctive tuple such as (Marc, Algebra) ∨ (Marc, Physics) by introducing its literals (Marc,Algebra) and (Marc,Physics) as maybe tuples. However, this entails a loss of information. Indeed, from the disjunctive tuple, it can be inferred that Marc takes at least one course; such certain information is lost when representing the fact with two maybe tuples.

Maybe tuples can be produced from disjunctive tuples when the database evolves. Suppose that in the relation above it becomes known that Paul takes Algebra. If we replace (Paul, Algebra) ∨ (Martha, Algebra) by (Paul, Algebra), the information that Martha could take Algebra is lost. Therefore, (Martha, Algebra) must be introduced as a maybe tuple as shown in Figure 10. In addition, maybe information is also obtained from disjunctive information by

	(*student, course*)
sure component	(Anne,Calculus)
	(Marc,Algebra)∨(Marc,Physics)
	(Paul,Algebra)
maybe component	(Martha,Algebra)

Figure 10: Disjunctive relation *takes*

applying relational operators.

Consider now the semantics of disjunctive relations, that is, their set of possible worlds. For the relation of Figure 10, if disjunctions are interpreted as inclusive, the tuple (Marc, Algebra) ∨ (Marc, Physics) represents 3 possible cases: Marc only takes Calculus, Marc only takes Physics, or Marc takes both courses. For the maybe component, either Martha takes Algebra or she does not. Therefore, under a closed world interpretation, the relation has 6 models.

A revised closure assumption is needed for disjunctive relations, to infer negative information, without having to store it explicitly. As is well-known, the CWA is inconsistent with disjunctive information. For this reason, Minker [Min82] proposed the *Generalized Closed World Assumption* (GCWA). The semantic definition of the GCWA is that a ground fact can be assumed false if it appears in no minimal model.

However, it is not possible to represent maybe information if the GCWA is applied to the database. Furthermore, the GCWA interprets disjunctions as exclusive, not as inclusive [RT88]. As noted by Przymusinsky [Prz88], every closure assumption based on minimal models interprets disjunctions as exclusive. Several extensions of the GCWA have appeared in the literature [YH85, GP86, GPP86] but they are all based on minimal models.

Ross and Topor [RT88] defined the Disjunctive Database Rule (DDR). When applying this rule to disjunctive relations, all tuples not explicitly represented either as a definite tuple, as a literal of a disjunctive tuple, or as a maybe tuple are assumed false, and disjunctions are interpreted inclusively. Rajasekar, Lobo and Minker [RLM89] defined the *Weak Generalized Closed World Assumption* (WGCWA), and they proved its equivalence with the DDR.

The WGCWA applied to the relation above can be represented with the following first-order formula:

$$\begin{aligned}\forall x\, \forall y\ \mathit{takes}(x,y) \rightarrow\ & (x = \text{Anne} \wedge y = \text{Calculus}) \vee \\ & (x = \text{Marc} \wedge y = \text{Algebra}) \vee (x = \text{Marc} \wedge y = \text{Physics}) \vee \\ & (x = \text{Paul} \wedge y = \text{Algebra}) \vee (x = \text{Martha} \wedge y = \text{Algebra}).\end{aligned}$$

Consider now the representation of disjunctive relations in first-order theories. The disjunctive relation of Figure 10 is represented with the above formula, in addition to the following two axioms:

$$\begin{aligned}&\forall x\, \forall y\ (x = \text{Anne} \wedge y = \text{Calculus}) \rightarrow \mathit{takes}(x,y),\\ &\mathit{takes}(\text{Marc}, \text{Algebra}) \vee \mathit{takes}(\text{Marc}, \text{Physics}).\end{aligned}$$

The first formula indicates those tuples which are definite, i.e., the tuples surely belonging to the relation. The second formula represents the disjunctive fact.

Query evaluation in disjunctive relational databases was first addressed in [GM86]. There is defined an algorithm to compute all minimal answers to a query; however, the main disadvantage of their algorithm is its complexity.

Disjunctive relations are used in [YC88] for obtaining complete query evaluation in databases containing unknown null values. Although a generalization of the relational algebra to disjunctive relations is given, several of the results are not correct. Further, using disjunctive information to obtain completeness in query evaluation with unknown values is impractical because of the complexity of the algorithms. Indeed, methods for manipulating existential values such as conditional relations are far more efficient than increasing the complexity of manipulating disjunctive databases with the addition of existential values.

Liu and Sunderraman [LS90a, LS90b] also extended the relational algebra for disjunctive relations. They proposed a semantics for disjunctive relations different from ours in that it is based on minimal models. As discussed in [Zim92a], this implies that their semantics treat disjunctions as exclusive. Even if they state that their interpretation of disjunction is inclusive, several of their results are correct only in the exclusive case.

However, they do not give a correct definition of the intersection and join operators: they are defined as a selection on the Cartesian product. This definition is totally impractical since computing in the general case the Cartesian product of two disjunctive relations R_1 and R_2 amounts to make the product of a possible world of R_1 with a possible world of R_2, i.e., to compute $REP(R_1) \times REP(R_2)$ and to obtain the disjunctive normal form of $REP(R_1) \times REP(R_2)$. In addition, they do not define the division operator.

We have argued [Zim92a] that it is not always adequate to interpret disjunctions as exclusive. Using our semantics, we generalize the relational operators for disjunctive relations, and prove their correctness. In particular, we give new correct definitions of intersection, join, and division operators.

Disjunctive relations can be generalized a step further by introducing negative clauses. Suppose we want to add to the relation of Figure 10 the disjunction

$$\mathit{takes}(\text{Paul}, \text{Physics}) \vee \mathit{takes}(\text{Helen}, \text{Physics}) \vee \mathit{takes}(\text{Lula}, \text{Physics})$$

and suppose in addition that the following assertions hold:

Marc takes either Algebra or Physics but not both; and
if Paul takes Physics then Helen and Lula also take Physics.

To model such assertions, we have to introduce disjunctive tuples containing negative literals. Therefore, disjunctive relations are extended with an extra *negative component.*

The above assertions can be represented with the formulas

$$\neg takes(\text{Marc}, \text{Algebra}) \vee \neg takes(\text{Marc}, \text{Algebra}), \tag{11.1}$$
$$\neg takes(\text{Paul}, \text{Physics}) \vee takes(\text{Helen}, \text{Physics}), \text{ and} \tag{11.2}$$
$$\neg takes(\text{Paul}, \text{Physics}) \vee takes(\text{Lula}, \text{Physics}). \tag{11.3}$$

Therefore, the relation becomes as follows:

	(*student, course*)
sure component	(Anne,Calculus)
	(Marc,Algebra)∨(Marc,Physics)
	(Paul,Algebra)
	(Paul,Physics)∨(Helen,Physics)∨(Lula,Physics)
maybe component	(Martha,Algebra)
negative component	¬(Marc,Algebra)∨¬(Marc,Physics)
	¬(Paul,Physics)∨(Helen,Physics)
	¬(Paul,Physics)∨(Lula,Physics)

The semantics defined above can be extended to take into account negative tuples. These tuples act as integrity constraints excluding the models in which Marc takes both Algebra and Physics, and the models in which Paul takes Physics and Helen or Lula do not. Thus, the relation *takes* has 16 models, while the relation without the negative component has 42 models.

To represent in logic these generalized disjunctive relations, negative tuples have to be introduced as clauses of the theory. For example, relation *takes* above can be represented with the axioms for the sure and maybe components as defined above, in addition to the formulas (11.1)–(11.3).

Introducing negative tuples in disjunctive relations raises the question of consistency. Consider the following relation *takes*(*student, course*),

(*student, course*)
(Peter,Algebra)∨(Peter,Physics)
(Peter,Algebra)∨(Peter,Calculus)
(Peter,Physics)∨(Peter,Calculus)
¬(Peter,Algebra)∨¬(Peter,Physics)
¬(Peter,Algebra)∨¬(Peter,Calculus)
¬(Peter,Physics)∨¬(Peter,Calculus)

in which the negative component represents the constraint that Peter takes at most one course. This relation is inconsistent since it has no model.

The expressive power of disjunctive relations containing negative tuples is significant. Indeed, these relations allow to model every set of ground formulas from first-order logic, since the formulas can be transformed into disjunctive normal form and treated as a disjunctive database. In particular, for each disjunction in the database, it can be chosen whether it is exclusive or inclusive, removing the restriction of an a priori choice between these two interpretations. Moreover, the relational operators can be generalized to treat the negative component of the relations without increasing their complexity. To our knowledge, no study has addressed the introduction of negative tuples in disjunctive relations.

12 Probabilistic Databases

Two different types of probabilistic data can be introduced in a relation: probabilistic information about the association of values and probabilistic information about data values. These two kinds of informations are respectively represented with *type-1* and *type-2 probabilistic* relations.

Type-1 probabilistic relations generalize classical relations with a supplementary attribute $w_R(\mathbf{t})$ indicating the probability that tuple $\mathbf{t}$ belong to relation R. The probability attached to a tuple is supposed to be independent from that of other tuples. Figure 11 shows an example of a type-1 relation. This relation states, for example, that Martin surely takes Physics, and that

student	*course*	w_R
Martin	Physics	1.0
Martin	Biology	0.9
John	Physics	0.6

Figure 11: A type-1 probabilistic relation *takes*.

the probability that he takes Biology is 0.9. Thus, the probability that he does not take Biology is 0.1. Under the CWA, every pair (student,course) not explicitly represented in the relation has probability 0.

Relation *takes* represents 4 different possible worlds, which are classical relations with an associated probability, computed as the product of the probabilities for the presence or absence of each tuple of relation *takes*. These possible worlds are as follows:

$$\begin{array}{l}
M_1 = \{(\text{Martin,Physics}),(\text{Martin,Biology}),(\text{John,Physics})\} \;\; \mathcal{P}(M_1) = 0.54 \\
M_2 = \{(\text{Martin,Physics}),(\text{Martin,Biology})\} \;\; \mathcal{P}(M_2) = 0.36 \\
M_3 = \{(\text{Martin,Physics}),(\text{John,Physics})\} \;\; \mathcal{P}(M_6) = 0.06 \\
M_4 = \{(\text{Martin,Physics})\} \;\; \mathcal{P}(M_8) = 0.04
\end{array}$$

Note that the probabilities add up to 1.

Although type-1 relations enable to represent uncertainties on the association of data values, their role to capture uncertainties in data values is limited. For this reason the notion of probabilistic set is needed. A *probabilistic set* in a universe of discourse U is defined with a probability distribution $w_F : U \to [0,1]$ satisfying $\sum_{u \in U} w_F(u) \leq 1$. A probabilistic set F is written as follows:

$$F = \{u_1/w(u_1), u_2/w(u_2), \ldots, u_n/w(u_n)\},$$

where $u_i \in U$, for $i = 1, \ldots, n$. It is always assumed that the values in a probabilistic set are mutually exclusive, i.e., the attribute can take only one value.

For example, if a course taken by John is represented by the probabilistic set Courses = $\{\text{Algebra}/0.5, \text{Calculus}/0.4\}$, this means that John is registered in Algebra with probability 0.5 and is registered in Calculus with probability 0.4. Since the probabilities add up to 0.9, John is registered in no course with probability 0.1.

Type-2 probabilistic relations have key attributes that are deterministic, as in classical relations. Thus, each tuple represents a known entity or relationship. The other attributes may be deterministic or stochastic. The latter are described with the help of probabilistic sets. Thus, the domain for a key attribute is a classical set, while the domain for a non key attribute is either a classical set or a set of probabilistic sets. An example of a type-2 probabilistic relation *takes* is given in Figure 12.

The semantics of type-2 probabilistic relations can be expressed in terms of possible worlds as follows. In Figure 12, the first tuple represents three possibilities: John takes Algebra, John

student	*course*
John	Algebra/0.5 Calculus/0.4
Anne	Physics/0.5 Calculus/0.5

Figure 12: A type-2 probabilistic relation *takes*.

takes Calculus, or John takes no course. For the second tuple, since the probabilities of the course taken by Anne add up to 1.0, there are only two possibilities. Therefore, the relation has the following 6 possible worlds.

$M_1 = \{$(John,Algebra),(Anne,Physics)$)\}$ $\mathcal{P}(M_1) = 0.25$
$M_2 = \{$(John,Calculus),(Anne,Physics)$\}$ $\mathcal{P}(M_2) = 0.20$
$M_3 = \{$(John,$\perp$),(Anne,Physics)$\}$ $\mathcal{P}(M_3) = 0.05$
$M_4 = \{$(John,Algebra),(Anne,Calculus)$\}$ $\mathcal{P}(M_4) = 0.25$
$M_5 = \{$(John,Calculus),(Anne,Calculus)$\}$ $\mathcal{P}(M_5) = 0.20$
$M_6 = \{$(John,$\perp$),(Anne,Calculus)$\}$ $\mathcal{P}(M_6) = 0.05$

Note that we represent that John takes no course with the inexistent null value $\perp$.

To understand the differences between type-1 and type-2 relations, compare the first tuple of the relation in Figure 12 with the following:

student	*course*	μ_R
John	Algebra	0.5
John	Calculus	0.4

Here, tuples are unrelated or independent, i.e., John takes Algebra with probability 0.5 and the probability that he does not take Algebra is 0.5, but this is unrelated to his probability of taking or not taking Calculus. On the contrary, in Figure 12, John takes only one course among Algebra and Calculus.

Type-2 probabilistic relations are further generalized by allowing several attributes to have dependent probability distributions. Consider for example a relation representing the probability distribution of course grades:

course	*marks*
Algebra	0–14/0.7 15–18/0.2 19–20/0.1
Physics	0–18/0.7 19–20/0.3
Calculus	19–20/1

The probability of propositions like "student x takes course y and obtain grade z", can be obtained through a probabilistic join as follows:

student	(*course*, *grade*)
John	(Algebra,0–14)/0.35
	(Algebra,15–18)/0.10
	(Algebra,19–20)/0.05
	(Calculus,19–20)/0.4
Anne	(Physics,0–18)/0.35
	(Physics,19–20)/0.15
	(Calculus,19–20)/0.5

For example, since Anne takes Physics with probability 0.5, she will obtain 0–18 with probability 0.35 and 19–20 with probability 0.15.

The development of a formal logical system for reasoning about probability is recent. After the pioneering work by Nilsson [Nil86], the main development in probabilistic logics was achieved by Fagin, Halpern, and Meggido [FH89, FHM90] and by Halpern [Hal90].

In order to formalize probabilistic databases, we use one of Halpern's probabilistic logics. Given a first-order language for reasoning about a domain and a formula ϕ of this logic, probabilistic logic allows formulas of the form $w(\phi) \geq \frac{1}{2}$ which can be interpreted as "the probability that ϕ is satisfied is greater than or equal to $\frac{1}{2}$".

In order to distinguish between probabilities and objects of the domain, Fagin, Halpern and Meggido use a two-sorted logic where a sort $\mathcal{O}$ describes objects of the domain and a sort $\mathcal{F}$ describes probabilities. In addition, in order to represent probabilistic sets, we introduce a supplementary sort $\mathcal{S}$ describing sets of objects. Variables of sorts $\mathcal{O}$, $\mathcal{S}$ and $\mathcal{F}$ are denoted respectively by x, x^s and x^f.

Probabilistic databases are formalized with probabilistic theories. As in relational theories, each relation is associated with a object predicate for which there is a set of *extension axioms*. Also, probabilistic theories contain a non empty set of simple types, modeling different domains for the variables.

For example, the type-1 relation *takes* of Figure 11, can be represented with the following extension axioms:

$$\begin{aligned}
\forall x\, \forall y\ \textit{takes}(x,y) \rightarrow\ & (x = \text{Martin} \wedge y = \text{Physics}) \vee (x = \text{Martin} \wedge y = \text{Biology}) \vee \\
& (x = \text{John} \wedge y = \text{Physics}), \\
\forall x\, \forall y\ & (x = \text{Martin} \wedge y = \text{Physics}) \rightarrow \textit{takes}(x,y), \\
\forall x\, \forall y\, \forall z^f\ & w(\textit{takes}(x,y)) = z^f \wedge 0 < z^f < 1 \leftrightarrow (x = \text{Martin} \wedge y = \text{Biology} \wedge z^f = 0.9) \vee \\
& (x = \text{John} \wedge y = \text{Physics} \wedge z^f = 0.6).
\end{aligned}$$

The first extension axiom realizes the closure of the relation by stating all the tuples belonging to it. With this axiom we can deduce for example that Anne does not take Physics. The second extension axiom states the tuples which surely belong to the relation, i.e., the tuples having probability 1.0. Finally, the third extension axiom specifies the tuples belonging to the relation with probability greater than 0 and less than 1.0.

Further, in probabilistic theories we need axioms for stating the independence of events like:

$$\begin{aligned}
& w(\textit{takes}(\text{Martin}, \text{Biology}) \wedge \textit{takes}(\text{John}, \text{Physics})) = \\
& \qquad w(\textit{takes}(\text{Martin}, \text{Biology})) \times w(\textit{takes}(\text{John}, \text{Physics})),
\end{aligned}$$

stating that both events are independent. Still other axioms could impose other conditions. For example

$$w(\textit{takes}(\text{Martin}, \text{Biology}) \leftrightarrow \textit{takes}(\text{Anne}, \text{Biology})) = 0.9$$

states that, with 0.9 probability, Martin takes Biology iff Anne also does.

Consider now the type-2 probabilistic relation given in Figure 12. First of all, the probabilities of the course taken by John are represented defining a probabilistic set ϵ_1 as follows:

$$\begin{aligned}
&\forall x\; x \in \epsilon_1 \rightarrow x = \text{Algebra} \vee x = \text{Calculus},\\
&\forall x\; \forall z^f\; w(x \in \epsilon_1) = z^f \wedge 0 < z^f < 1 \leftrightarrow\\
&\qquad\qquad (x = \text{Algebra} \wedge z^f = 0.2) \vee (x = \text{Calculus} \wedge z^f = 0.4),\\
&\forall x\; \forall y\; x \in \epsilon_1 \wedge y \in \epsilon_1 \rightarrow x = y.
\end{aligned}$$

Notice that the last axiom represents that John takes only one course. The probabilistic set ϵ_2 representing the probabilities of the course taken by Anne is defined in similar way. Finally, relation *takes* is represented with the following extension axioms:

$$\begin{aligned}
&\forall x\; \forall y\; \mathit{takes}(x, y) \rightarrow (x = \text{John} \wedge y \in \epsilon_1) \vee (x = \text{Anne} \wedge y \in \epsilon_2),\\
&\forall x\; \forall y\; \forall z^f\; w(\mathit{takes}(x, y)) = z^f \wedge 0 < z^f < 1 \leftrightarrow\\
&\qquad\qquad (x = \text{John} \wedge y \in \epsilon_1 \wedge z^f = w(y \in \epsilon_1)) \vee\\
&\qquad\qquad (x = \text{Anne} \wedge y \in \epsilon_2 \wedge z^f = w(y \in \epsilon_2)).
\end{aligned}$$

The probabilistic approach for modeling uncertainty in relational databases has not been deeply studied so far. Preliminary models for probabilistic relational databases have been proposed in [GH86, CP87]. The Probabilistic Database Model defined in [BGMP90, BGMP92] studies probabilistic relations similar to our type-2 probabilistic relations, although they considered a different semantics. We studied in [Zim92a, Zim92c, Zim92b] the manipulation of type-1 probabilistic relations from an algebraic point of view, extending the relational operators and proving their correctness. Also, type-1 probabilistic relations are formalized in a logical framework with probabilistic theories, and we give a sound and complete query evaluation algorithm for such theories. The corresponding studies for type-2 probabilistic relations suggest an open area of research.

13 Conclusion

The shades of uncertainty and incompleteness are intricate. Much work has been devoted recently to the field of imperfect knowledge. Unfortunately, the approaches and methodologies adopted differ widely, which makes a fruitful survey difficult. This paper has tried to compare various proposals by viewing them through the same tools, namely possible worlds and logic.

Syntactical objects like null values, disjunctive information, or probabilities are used to represent different kinds of incompleteness and uncertainty. To formally specify the meaning of these objects, the notion of possible world is an invaluable tool applicable to many different cases: it allows to define the semantics of information by associating to each of type of incompleteness and uncertainty the various possible underlying realities that it represents. As we have tried to show, logic is a convenient formalism to precisely state the different semantics assigned to each type of information. Classical first-order logic can be used to represent incomplete information, whereas probabilistic logics can be used to represent uncertainty.

Another important concern is the manipulation of the database in the presence of incompleteness and uncertainty; query processing, integrity constraint checking, and database updating have to be adapted to the imperfect knowledge and its associated semantics. The notion of possible worlds is also paramount for specifying the semantics and correctness of data manipulation.

The research of the last few years has definitely established that the field is a complex one and that exact results with a precise and adequate semantics are expensive to obtain.

References

[AB86] S. Abiteboul and N. Bidoit. Non first normal form relations: An algebra allowing data restructuring. *Journal of Computer and System Sciences*, 33:361–393, 1986.

[AB90] P. Atzeni and M.C. De Bernardis. A new interpretatin for null values in the weak instance model. *Journal of Computer and System Sciences*, 41(1):25–43, 1990.

[AG85] S. Abiteboul and G. Grahne. Update semantics for incomplete databases. In *Proc. 11th Int. Conf. on Very Large Databases*, pages 1–12, Stockholm, 1985.

[AKG91] S. Abiteboul, P. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. *Theoretical Computer Science*, 78(1):159–187, January 1991.

[ANS75] ANSI/X3/SPARC. Study group on Data Base Management Systems Interim Report. *SIGMOD FDT Bull.*, 7(2), 1975.

[AT89] P. Atzeni and R. Torlone. Approaches to updates over weak instances. In *Proc. 2nd Symp. on Mathematical Fundamentals of Database Systems, Visegrad, Hungary, June 1989, Springer, LNCS 364*, 1989.

[BGMP90] D. Barbará, H. García-Molina, and D. Porter. A probabilistic relational data model. In F. Bancilhon, C. Thanos, and D. Tsichritzis, editors, *Proc. of the Int. Conf. on Extending Database Technology, EDBT'90*, LNCS 416, pages 60–74, Venice, Italy, 1990. Springer-Verlag.

[BGMP92] D. Barbará, H. García-Molina, and D. Porter. The management of probabilistic data. *IEEE Trans. on Knowledge and Data Engineering*, 4(5):487–502, 1992.

[Bis81] J. Biskup. A formal approach to null values in database relations. In H. Gallaire, J. M., and J.-M. Nicolas, editors, *Advances in Data Base Theory*, volume 1, pages 299–341. Plenum Press, 1981.

[Bis83] J. Biskup. A foundation of Codd's relational maybe-operations. *ACM Trans. on Database Systems*, 8(4):608–636, December 1983.

[Cod75] E.F. Codd. Understanding relations (installment # 7). *FDT Bull. of ACM SIGMOD*, 7(3–4):23–2, 1975.

[Cod79] E.F. Codd. Extending the database relational model to capture more meaning. *ACM Trans. on Database Systems*, 4(4):397–434, December 1979.

[Cod86] E.F. Codd. Missing information (applicable and inapplicable) in relational databases. *SIGMOD Record*, 15(4):53–78, December 1986.

[Cod87] E.F. Codd. More commentary on missing information in (applicable and inapplicable) relational databases. *SIGMOD Record*, 16(1):42–50, March 1987.

[CP87] R. Cavallo and M. Pittarelli. The theory of probabilistic databases. In *Proc. 13th Int. Conf. on Very Large Databases*, Brighton, U.K., 1987.

[Dat86] C. J. Date. *Relational Databases: Selected Writings.* Addison-Wesley, Reading, Mass., 1986.

[DF88] R. Demolombe and L. Fariñas Del Cerro. An algebraic evaluation method for deduction in incomplete databases. *Journal of Logic Programming*, 5(3):183–205, September 1988.

[FH89] R. Fagin and J.Y. Halpern. Uncertainty, belief and probability. In *Proc. of the 1989 Int. Joint Conf. on Artificial Intelligence*, pages 1375–1381, 1989.

[FHM90] R. Fagin, J. Halpern, and N. Meggido. A logic for reasoning about probabilities. *Information and Computation*, 87:78–128, 1990.

[FM92] J.A. Fernandez and J. Minker. Disjunctive deductive databases. In *Proc. of the International Conference on Logic Programming and Automated Reasoning*, pages 332–356, St. Petersburg, Russia, 1992.

[GH86] E. Gelenbe and G. Hebrail. A probability model of uncertainty in databases. In *Proc. of the Int. Conf. on Data Engineering*, 1986.

[GM86] J. Grant and J. Minker. Answering queries in indefinite databases and the null value problem. In B. Buchanan and B. & E. Shortliffe, editors, *Advances in Computing Research*, volume 3, pages 247–267. JAI Press, 1986.

[GP86] M. Gelfond and H. Przymusinska. Negation as failure: Careful closure procedure. *Artificial Intelligence*, 30:273–287, 1986.

[GPP86] M. Gelfond, H. Przymusinska, and T.C. Przymusinsky. The extended closed world assumption and its relationship to parallel circumscription. In *Proc. 5th ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 133–139, 1986.

[Gra77] J. Grant. Null values in a relational data base. *Information Processing Letters*, 6(5):156–157, October 1977.

[Gra84] G. Grahne. Dependency satisfaction in databases with incomplete information. In *Proc. 10th Int. Conf. on Very Large Databases*, pages 37–45, Singapore, 1984.

[Gra91] G. Grahne. *The Problem of Incomplete Information in Relational Databases.* LNCS 554. Springer-Verlag, 1991.

[GZ88] G. Gottlob and R. Zicari. Closed world databases opened through null values. In *Proc. 14th Int. Conf. on Very Large Databases*, pages 50–61, Los Angeles, August 1988.

[Hal90] J.Y. Halpern. An analysis of first–order logics of probability. *Artificial Intelligence*, 46(3):311–350, June 1990.

[Hon82] P. Honeyman. Testing satisfaction of functional dependencies. *Journal of the ACM*, 29(3):668–667, 1982.

[HPRV89] G. Hulin, A. Pirotte, D. Roelants, and M. Vauclair. Logic and databases. In André Thayse, editor, *From Modal Logic to Deductive Databases*, pages 279–350. Wiley, 1989.

[Hul90] G. Hulin. Relational databases with marked null values: A new approach. Manuscript M333, Philips Research Laboratory Belgium, January 1990.

[IL83] T. Imieliński and W. Lipski, Jr. Incomplete information and dependencies in relational databases. In *Proc. ACM-SIGMOD Int. Conf. on Management of Data*, pages 178–184, San Jose, Calif., May 1983.

[IL84] T. Imieliński and W. Lipski, Jr. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, October 1984.

[Imi86] T. Imieliński. Query processing in deductive databases with incomplete information. In *Proc. ACM-SIGMOD Int. Conf. on Management of Data*, pages 268–280, Washington, May 1986.

[Imi89] T. Imieliński. Incomplete information in logical databases. *IEEE Data Engineering*, 12(2):29–40, 1989.

[IV89] T. Imielinski and K. Vadaparty. Complexity of query processing in databases with or-objects. In *Proc. 8th ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, 1989.

[KS91] P. King and C. Small. Default databases and incomplete information. *The Computer Journal*, 34(3):239–244, 1991.

[Ler86a] N. Lerat. Evaluation de requêtes sur une base de données logique satifaisant un ensemble de dépendances. In *2èmes Journées Bases de Données Avancées*, pages 233–243, Giens, April 1986.

[Ler86b] N. Lerat. Query processing in incomplete logical databases. In G. Ausiello and P. Atzeni, editors, *Proc. Int. Conf. on Database Theory*, LNCS 243, pages 261–277, Rome, Italy, September 1986. Springer-Verlag.

[Lev92] M. Levene. *The Nested Universal Relation Database Model.* LNCS 595. Springer-Verlag, 1992.

[Lip79] W. Lipski Jr. On semantic issues connected with incomplete information databases. *ACM Trans. on Database Systems*, 4(3):262–296, September 1979.

[Lip81] W. Lipski Jr. On databases with incomplete information. *Journal of the ACM*, 28(1):41–70, January 1981.

[Lip84] W. Lipski Jr. On relational algebra with marked nulls. In *Proc. 3rd ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 201–203, Warterloo, Canada, 1984.

[LS90a] K.-C. Liu and R. Sunderraman. Indefinite and maybe information in relational databases. *ACM Trans. on Database Systems*, 15(1):1–39, March 1990.

[LS90b] K.-C. Liu and R. Sunderraman. On representing indefinite and maybe information in relational databases: a generalization. In *Proc. of the 6th IEEE Int. Conf. on Data Engineering*, pages 495–502, Los Angeles, California, USA, February 1990.

[Mai83] D. Maier. *The Theory of Relational Databases.* Pitman Publ., London, 1983.

[Mak77] A. Makinouchi. A consideration on normal form of not-necessarily-normalized relation in the relational data model. In *Proc. 3rd Int. Conf. on Very Large Databases*, pages 447–453, Tokyo, Japan, October 1977.

[MG88] Z. Michalewicz and L.J. Groves. Sets and uncertainty in relational databases. In B. Bouchon, L. Saitta, and R.R. Yager, editors, *Uncertainty and Intelligent Systems, IPMU'88*, LNCS 313, pages 127–137, Urbino, Italy, July 1988. Springer-Verlag.

[Min82] J. Minker. On indefinite databases and the closed world assumption. In D. W. Loveland, editor, *Proceedings of the 6th Conference on Automated Deduction*, LNCS 138, pages 292–308, New York, USA, June 1982. Springer-Verlag.

[Min89] J. Minker. Toward a foundation of disjunctive logic programming. In E.L. Lusk and R.A. Overbeek, editors, *Proceedings of the North American Logic Programming Conference*, pages 1215–1235. MIT Press, 1989.

[MU83] D. Maier and J. Ullman. Maximal objects and the semantics of the universal relation databases. *ACM Trans. on Database Systems*, 8(1):1–14, 1983.

[MUV84] D. Maier, J.D. Ullman, and M. Vardi. On the foundations of the universal relation model. *ACM Trans. on Database Systems*, 9(2):283–308, 1984.

[MY87] Z. Michalewicz and A. Yeo. Sets in relational databases. In *Proc. of the Canadian Information Processing Society*, pages 237–245, Edmonton, November 1987.

[Nil86] N.J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986.

[Prz88] T.C. Przymusinsky. On the declarative semantics of deductive databases and logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming.* Morgan Kaufmann, Los Altos, California, 1988.

[Rei78] R. Reiter. On closed world data bases. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 55–76. Plenum Press, 1978.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1, 2):81–132, 1980.

[Rei84] R. Reiter. Towards a logical reconstruction of relational database theory. In Michael Brodie, John Mylopoulos, and Joachim Schmidt, editors, *On Conceptual Modelling*, pages 191–238. Springer-Verlag, Berlin, 1984.

[Rei86] R. Reiter. A sound and sometimes complete query evaluation algorithm for relational databases with null values. *Journal of the ACM*, 33(2):349–370, April 1986.

[Rei90] R. Reiter. What should a database know? In *Proceedings of the Symposium on Computational Logic*, pages 96–113, Brussels, Belgium, 1990.

[RKS89] M.A. Roth, H.F. Korth, and A. Silberschatz. Null values in nested relational databases. *Acta Informatica*, 26:615–642, 1989.

[RLM89] A. Rajasekar, J. Lobo, and J. Minker. Weak generalized closed world assumption. *Journal of Automated Reasoning*, 5(3):293–307, 1989.

[RRHD88] T. Risch, R. Reboh, P. Hart, and R. Duda. A functional approach to integrating database and expert systems. *Communications of the ACM*, 31(12):1424–1437, February 1988.

[RT88] K. Ross and R. Topor. Inferring negative information from disjunctive databases. *Journal of Automated Reasoning*, 4:397–424, 1988.

[Sow84] J.F. Sowa. *Conceptual Structures.* Addison-Wesley, 1984.

[SS86] H.-J. Schek and M.H. Scholl. The relational model with relation-valued attributes. *Information Systems*, 11(2):137–147, 1986.

[Ull82] J. Ullman. The U.R. strikes back. In *Proc. of ACM Symp. on Principles of Database Systems*, pages 10–22, Los Angeles, 1982.

[Ull83] J. Ullman. Universal relation interfaces for database systems. In *Proc. of the 9th IFIP World Computer Congress*, pages 243–252, Paris, France, 1983.

[Vas79] Y. Vassiliou. Null values in data base management: A denotational approach. In Philip A. Bernstein, editor, *Proc. ACM-SIGMOD Int. Conf. on Management of Data*, pages 162–169, Boston, Massachusetts, USA, May 1979. Assoc. for Computing Machinery.

[Vas80] Y. Vassiliou. Functional dependencies and incomplete information. In *Proc. 6th Int. Conf. on Very Large Databases*, pages 260–269, Montreal, Canada, October 1980.

[Win86] M. Winslett Wilkins. Updating logical databases containing null values. In Giorgio Ausiello and Paolo Atzeni, editors, *Proc. Int. Conf. on Database Theory*, LNCS 243, pages 421–435, Rome, Italy, September 1986. Springer-Verlag.

[Win88a] M. Winslett Wilkins. A framework for comparison of update semantics. In *Proc. of ACM Symp. on Principles of Database Systems*, pages 315–324, Austin, Texas, 1988.

[Win88b] M. Winslett Wilkins. A model-based approach to updating databases with incomplete information. *ACM Trans. on Database Systems*, 13(2):167–196, 1988.

[YC88] L.Y. Yuan and D.-A. Chiang. A sound and complete query evaluation algorithm for relational databases with null values. In *Proc. ACM-SIGMOD Int. Conf. on Management of Data*, pages 74–81, Chicago, June 1988.

[Yeo87] A. Yeo. Sets in relational databases. Master's thesis, Victoria University, Wellington, New Zealand, Department of Computer Science, 1987.

[YH85] A. Yahya and L. Henschen. Deduction in non-Horn databases. *Journal of Automated Reasoning*, 1:141–160, 1985.

[Zan84] C. Zaniolo. Database relations with null values. *Journal of Computer and System Sciences*, 28:142–166, 1984.

[Zim92a] E. Zimányi. *Incomplete and Uncertain Information in Relational Databases.* PhD thesis, Université Libre de Bruxelles, Belgium, July 1992.

[Zim92b] E. Zimányi. Probabilistic relational databases. Technical Report RR 92-02, INFODOC, Université Libre de Bruxelles, Belgium, October 1992. Submitted to publication.

[Zim92c] E. Zimányi. Query evaluation in probabilistic databases. Technical Report RR 92-01, INFODOC, Université Libre de Bruxelles, Belgium, September 1992.