UNCERTAINTY MANAGEMENT IN INFORMATION SYSTEMS

FROM NEEDS TO SOLUTIONS

EDITED BY

Amihai Motro
George Mason University
Fairfax, VA, USA

Philippe Smets
Université Libre de Bruxelles
Brussels, Belgium

KLUWER ACADEMIC PUBLISHERS
Boston/London/Dordrecht

2

SOURCES OF UNCERTAINTY, IMPRECISION, AND INCONSISTENCY IN INFORMATION SYSTEMS

Amihai Motro

Department of Information and Software Systems Engineering
George Mason University
Fairfax, VA 22030, USA

1 INTRODUCTION

An information system is a computer model of the real world. Like any other model, it captures an abstracted version of the real world, using a level of abstraction that is implied by the expected applications. As with any other model, the most important consideration is the integrity of the model; i.e., the accuracy of the representation. Unfortunately, our knowledge of the real world is often imperfect, thus challenging our ability to create and maintain information systems of integrity.

There are two solutions for upholding the integrity of an information systems in situations in which knowledge of the real world is imperfect.

The first solution is to *restrict* the model to that portion of the real world about which perfect information is available. Assume, for example, that the information model being used describes each employee with a record of several fields. Then, only the employees for whom perfect information is available in each of these fields would be included in the information system.

The second solution is to develop information models that allow the representation of imperfect information. Assume that the available information about the age of a particular employee is imperfect; for example, the age is only known to be within a specific range. If the information model had features for specifying and manipulating ranges, then this imperfect information could be captured in a system that still maintains its integrity.

Because the second solution often permits additional applications, most information systems adhere to information models that include at least some features for capturing imperfect information. For example, database systems can represent missing values, information-retrieval systems can match information to requests using a "weak" matching algorithm, and expert systems can represent rules that are known to be true only most or some of the time.

Yet, these capabilities are weak, in comparison with the variety and the degree of imperfection that is encountered in practice. Although the research community has shown persistent interest in this subject (see the extensive bibliography in Chapter 15), most of these efforts have yet to transcend experimental prototypes. By and large, commercial information systems have been slow to incorporate capabilities for handling imperfect information.

Nevertheless, many new applications require such capabilities (see Section 5). Without general systems that possess capabilities for handling imperfect information, such applications must be dealt with in an ad-hoc manner; this usually means that specific algorithms must be designed and specific systems must be built for each application that cannot be satisfied by the present generation of information systems.

This chapter is intended as a general introduction to the issues of imperfect information in information systems. Our goals are to *classify* the various kinds of imperfect information that are encountered in information systems (this fundamental task is undertaken in several other chapters as well), and to *survey* the basic solutions that have been proposed. Our treatment is *comprehensive*, in that we consider imperfections in all aspects of information systems, including the representation of imperfect information, imperfections in the specification of transactions, and imperfections in the processing of transactions. At the same time, our treatment is *abstract*, in that our classification and survey attempt to identify principles that are common to different kinds of information systems.

In our examples we consider three popular types of information systems: database systems, information-retrieval systems, and expert systems; databases are considered in more detail in Chapters 3, 4, and 5; expert systems are covered in Chapter 6, and information-retrieval systems are covered in Chapter 7.

For our goal of comprehensive and abstract treatment of the issues of imperfect information, we define here a simple, general model of an information system.

An information system includes a declarative component for *describing* the real world, and an operational component for *manipulating* this description. Typical

manipulations are of two different kinds: (1) *modifications* of the description, either to refine the model or to track any changes that may have occurred in the real world, and (2) *transformations* of the description, to derive implied descriptions. Thus, an information system can be abstracted as a description D of the real world, a stream of modifications, and a stream of transformations. Each modification m replaces the present description with a new description; each transformation t computes a new description from the present description (without changing the present description).

As an example, in a relational database system a description is a set of tables (i.e., a database); a modification can affect either the definition or the contents of tables (i.e., restructuring or update); and a transformation reduces the set of tables into a single table (i.e., query evaluation).

The objective of information systems is to provide their users with the information they need. In our terminology, such information is always the result $t(D)$ of transforming a description D with a transformation t. Thus, it is the quality of $t(D)$, rather than the quality of D, that should concern the designers and users of information systems. A result $t(D)$ may be imperfect, because either

- the initial description D is imperfect,
- the specification of a later modification m of the description is imperfect,
- the specification of the transformation t is imperfect, or
- the processing of t against D is imperfect.

Whereas most treatments of this subject focus exclusively on imperfect descriptions, in this chapter we shall examine all these disparate sources of imperfection. Sections 2 and 3 are devoted to imperfect descriptions: in Section 2 we classify the various kinds of imperfect descriptions, and in Section 3 we survey the basic solutions that have been proposed. Section 4 is devoted to imperfections in modifications, transformations, and processing. Finally, in Section 5 we speculate on the reasons commercial information systems have been slow to incorporate capabilities for dealing with imperfect information. We then describe several applications that challenge present database systems by requiring more powerful methods for dealing with imperfection, and we point to some promising areas of research; in particular, we postulate that suitable solutions could come from fusing information systems technology with various theories of artificial intelligence.

2 IMPERFECT DESCRIPTIONS: CLASSIFICATION

A basic assumption that we adopt is that imperfections permeate our *models* of the real world, not the real world itself. Hence, we assume that a perfect description of the real world does exist but may be unavailable. The stored description is therefore an *approximation* of an ideal description (this assumption is further elaborated upon in Section 3.6).

In this section we examine several basic issues regarding description imperfection; namely, the different *kinds* of imperfection, the different *elements* of descriptions that could be affected with imperfection, the different *causes* for imperfection, and the different *degrees* of imperfection.

2.1 The Basic Kinds of Imperfect Information

There have been many attempts to classify the various possible kinds of imperfect information. In this chapter we concern ourselves with three basic kinds: error, imprecision, and uncertainty. We note that other kinds of imperfection have been observed, including vagueness and ambiguity (see Chapter 8), but they are not as important for information systems.

Error. Erroneous information is the simplest kind of imperfect information. Stored information is erroneous when it is different from the true information. We take the approach that all errors, large or smalls, compromise the integrity of an information system, and should not be tolerated. An important kind of erroneous information is *inconsistency*. Occasionally, the same aspect of the real world is represented more than once (this could be in the same information system, or in different information systems that are considered together). When the different representations are irreconcilable, the information is inconsistent. Issues of information inconsistency are of particular relevance, given the present interest in information integration [20, 32].

Imprecision. Stored information is imprecise when it denotes a *set* of possible values, and the real value is one of the elements of this set. Thus, imprecise information is not erroneous and does not compromise the integrity of an information system. Specific kinds of imprecise information include *disjunctive* information (e.g., John's age is either 31 or 32), *negative* information (e.g., John's age is not 30), *range* information (e.g., John's age is between 30 and 35, or John's age is over 30), and information with *error margins* (e.g., John's age

is 34 ± 1 year). The two extreme kinds of imprecision are *precise* values and *null values*: a value is precise when the set of possibilities is a singleton; a null value usually denotes that no information is available, yet could be regarded as imprecise information where the set of possible values encompasses the entire domain of legal values.

Uncertainty. At times, our knowledge of the real world (precise or imprecise) cannot be stated with absolute confidence. This requires that we qualify our confidence in the information stated. Again, information with qualified certainty is not erroneous and does not compromise the integrity of an information system. Whereas the statement "John's age is either 31 or 32" denoted imprecision, the statement "John is probably 32" denotes uncertainty. At times, precision can be traded for certainty and vice-versa. A precise value may entail low certainty, but as this value is substituted by values that are progressively less precise, certainty increases gradually, until finally it is maximal for a value that is minimally precise; i.e., a null value (see also the hypothesis about an Information Maximality Principle in Chapter 8).

2.2 What Might Be Imperfect?

Depending on the model used, descriptions may take different forms, and imperfection can affect each of them.

Consider, for example, relational databases. The structures of the relational model admit different kinds of imperfection. The first kind involves imperfection at the level of data values; for example, the values of *Salary* in the relation *Earn (Employee, Salary)* might be imprecise. The second kind involves imperfection at the level of the tuple; for example, the values of each of the attributes of the relation *Assign (Employee, Project)* may be certain and precise, but there might be uncertainty about the exact assignment of employees to projects. A third kind involves imperfection at the level of the relation scheme (the structure); for example, there might be uncertainty whether employees may belong to more than one department, and hence what should be the proper description of this relationship [5].

As another example, consider an information-retrieval system that models each document with an identifier and a vector of keywords. There might be uncertainty at the level of a keyword; i.e., the appropriateness of a specific keyword to a given document may be questionable. In addition, there might be uncer-

tainty at the level of an entire document; i.e., the existence of some documents may be in doubt.

As a third example, consider an expert system that models real-world knowledge with facts and rules expressed in logic. There might be uncertainty about specific facts (similar to the tuple uncertainty in relational databases), and there might be uncertainty about specific rules; i.e., a rule might be only an approximation of the behavior of the real world.

2.3 Why Is It Imperfect?

Having excluded the possibility that reality itself is subject to imperfection, we can assume that a perfect description of any real-world object always exists. Thus, imperfect descriptions are solely due to the *unavailability* of these perfect descriptions. For example, the precise salary of Tom might be unknown, or the true relationship between Bordeaux wine and good health might be unclear. Yet, within this generic unavailability, we observe several specific causes (refer also to Chapter 5 for a more detailed survey of the sources of imperfect information).

Imperfect information might result from using *unreliable information sources*, such as faulty reading instruments, or input forms that have been filled out incorrectly (intentionally or inadvertently). In other cases, imperfect descriptions are a result of *system errors*, including input errors, transmission noise, delays in processing update transactions, imperfections of the system software, and corrupted data owing to failure or sabotage.

At times, the imperfect information is the unavoidable result of information gathering methods that require *estimation* or *judgment*. Examples include the determination of the subject of a document, the digital representation of a continuous phenomenon, and the representation of a phenomenon that is constantly varying.

In other cases, imperfections are the result of restrictions imposed by the *model*. For example, if the database schema permits storing at most two occupations per employee, descriptions of occupation would be incomplete. Similarly, the sheer *volume* of the information that is necessary to describe a real-world object might force the modeler to turn to approximation and sampling techniques.

2.4 How Imperfect Is It?

The relevant information that is available in the absence of perfect information may take different forms, each exhibiting a different *level* of imperfection. The amount of relevant information available offers an alternative classification of imperfections. The following discussion is independent of the particular structure affected by imperfection. It assumes an element e of the description models an object o of the real world. The element e might be a value, a tuple, a fact, a rule, and so on.

The least informative case (this is often referred to as *ignorance*) is when the mere *existence* of some real-world object o is in doubt. The simplest solution is to ignore such objects altogether. This solution, however, is unacceptable if the model claims to be *closed world* (i.e., the model guarantees that objects not modeled do not exist).

Ignorance is reduced somewhat when each element e of the description is assigned a value in a prescribed range, to indicate the likelihood that the modeled object exists. When the element is a fact, this value can be interpreted as the *confidence* that the fact holds; when it is a rule, this value can be interpreted as the *strength* of the rule (e.g., the proportion of cases in which the rule applies).

Assume now that existence is assured, but some of or all the information with which the model describes an object is *unknown*. Such information has also been referred to as *incomplete*, *missing*, or *unavailable*.

The quality of the information increases when the description of an object is known to come from a limited set of alternatives (possibly a range of values). This information is referred to as *disjunctive* information. Note that when a set of alternatives is simply the entire universe of possible values, this case reverts to the previous (less informative) case. When this set contains a single value, the available information is perfect.

The quality increases even further when each alternative is accompanied by a number describing the likelihood that it is indeed the true description. When these numbers indicate probabilities, the information is *probabilistic*, and when they indicate possibilities, the information is *possibilistic*. Again, when these numbers are unavailable, this case reverts to disjunctive information.

3 IMPERFECT DESCRIPTIONS: SOLUTIONS

Space considerations forbid discussion of all the different solutions that have been attempted for accommodating imperfections in descriptions. We sketch here five approaches that are significantly different from each other; they also exhibit sufficient generality to be applicable in different information systems. Of course, any approach for *describing* imperfect information must also address the *manipulation* (transformation and modification) of imperfect descriptions (e.g., querying and updating).[1]

3.1 Null and Disjunctive Values

Most data models insist that similar real-world objects be modeled with similar descriptions. The simplest example of this approach are models that use tabular descriptions. Each such table models a set of similar real-world objects: each row describes a different object, and the columns provide the different components of the description. Often, some elements of a particular description cannot be stated with precision and certainty. Occasionally, this problem may be evaded simply by not modeling any object whose description is imperfect. Often, however, the consequences of this approach are unacceptable, and imperfect descriptions must be admitted.

The least ambitious approach to admitting imperfect descriptions is to ignore all partial information that may be available about the imperfect parts of a description, and to model them with a pseudo-description, called *null*, that denotes unavailability [25, 11, 23].[2] The semantics of a null value is that *any* value from the corresponding domain of legal values is an equally probable candidate for the true value.

Once null values are admitted into descriptions, the model must define the behavior of transformations and modifications in the presence of nulls. This is not a simple task. For example, an extension to the relational calculus that is founded on a three-valued logic [10] has been the subject of criticism [12].

[1] Note the difference between manipulations of imperfect descriptions, which are discussed in this section, and imperfect manipulations, which are the subject of Section 4.

[2] Null values have also been used to denote *inapplicability*; i.e., that a specific attribute is inapplicable to a specific object. Such null values do not indicate imperfection of information. Hence, the term *null* will be used here to denote applicability but unavailability.

Inference in incomplete databases is discussed in [14], and updates of incomplete databases are discussed in [1].

Null values model the ultimate lack of information about a value (except that it exists). Other kinds of null values have been suggested that express some additional information. For example, two values in a database may be unavailable but are known to be identical. Such partial information may be modeled by using distinguishable instances of nulls (*marked nulls*) throughout the database in general, but the same null instance for these two values. Recording this partial information proves to be useful in the performance of joins.

At times, it is known that a missing value belongs to a more limited set of values (possibly, a range of values). This partial information has been modeled by *disjunctive values*. A disjunctive value is a set of values that includes the true value. Hence, disjunctive values are more informative than nulls (a null value is a specific kind of disjunctive value, in which the set of possible values is the entire domain). Disjunctive databases are discussed in [24, 26].

Clearly, null and disjunctive values both express *imprecision*. More on these kinds of imprecision can be found in Chapters 3 and 4.

3.2 Confidence Factors

Confidence factors denote confidence in various elements of the description. Hence, they offer a simple tool for representing *uncertainty*. Confidence factors have been applied in both information-retrieval systems and in expert systems but not in database systems proper.

In information-retrieval systems, confidence factors (often called *weights* or *relevance coefficients*) have been used to denote confidence that a specific keyword describes a given document (or alternatively, to denote the strength with which this keyword applies to the document) [45, 41]. Methods have even been developed for computing confidence factors automatically, by scanning documents and applying keyword counting techniques. The manipulation of these confidence factors is relatively simple, as they are easily accommodated in the vector space models that are often used in information retrieval. Refer to Chapter 7 for the use of relevance coefficients in information retrieval.

In expert systems, confidence factors have been used to denote confidence that stated facts and rules indeed describe real-world objects [18]. Such factors are

usually declared by the knowledge engineers as part of the knowledge acquisition process but can also be derived automatically as part of a knowledge discovery process [35]. The manipulation of confidence factors in expert systems is often straightforward; for example, assuming confidence factors in the range [0,1], when a rule with confidence p is applied to a fact with confidence q, the generated fact is assigned a confidence factor $p \cdot q$. Pragmatic considerations may have been the reason that commercial expert systems often prefer this mostly informal representation of uncertainty over more formal approaches that are based on probability theory. However, many objections have been raised against confidence factors, showing that the lack of firm semantics may lead to unintuitive results [34].

3.3 Probability Theory

Probabilistic information systems represent information with variables and their probability distributions. In a relational framework, the value of a particular attribute A for a specific tuple t is a variable $A(t)$, and this variable has an associated *probability distribution* $P_{A(t)}$. $P_{A(t)}$ assigns values in the range $[0,1]$ to the elements of the domain of attribute A, with the provision that the sum of all values assigned is 1. An example of a probabilistic value is the variable $Age(john)$ and this probability distribution:[3]

$$P_{Age(john)} = \left\{ \begin{array}{ll} 32 & 0.6 \\ 33 & 0.4 \end{array} \right.$$

The interpretation of this information is that with probability 0.6 the age of John is 32, with probability 0.4 it is 33, and the probability that John's age is some other value is 0.

A probabilistic relational model based on this approach and a suitable set of operators are defined in [3]. The model allows probability distributions that are incompletely specified: each such distribution is completed with a special null value, which is assigned the balance of the probability. It is also possible to define probability distributions for combinations of interdependent attributes. A somewhat different (but equivalent) model is given in [8]. In this model a database is a collection of objects in which each object has an associated set of attributes (a scheme), a set of tuples over that scheme (a domain), and a probability distribution function that assigns each tuple of the domain a probability (the sum of the probabilities for the entire domain of an object is 1).

[3] We use the key value $john$ to identify a specific tuple.

We have observed that disjunctive information; e.g., "32 or 33" is a form of imprecise information, whereas "32 with confidence 0.6" is a form of uncertain information. Consider now this statement "32 with probability 0.6 and 33 with probability 0.4." In many ways, such probabilistic information is a combination of both imprecision and uncertainty. It is imprecise because it denotes several different alternatives, and it is uncertain because every alternative is associated with a likelihood. Our conclusion is that the distinction between imprecision and uncertainty is often useful, but is not a "partitioning" classification. A similar conclusion is reached in Chapter 9, which offers more details on the use of probability theory for modeling imperfect information.

3.4 Fuzziness and Possibility Theory

The basic concept of fuzzy set theory is the *fuzzy set*. A fuzzy set F is a set of elements in which each element has an associated value in the interval [0,1] that denotes the *grade* of its membership in the set. An example of a fuzzy set (using a common notation) is $F = \{30/1.0, 31/1.0, 32/1.0, 33/0.7, 34/0.5, 35/0.2\}$. The elements 30, 31, and 32 are in this set with grade of membership 1.0, the elements 33, 34, and 35 have corresponding grades of membership 0.7, 0.5, and 0.2, and all elements not shown have grade of membership 0.

Several different models of databases have been based on fuzzy set theory. The simplest model extends relations, which are subsets of a Cartesian product of domains, to fuzzy relations; i.e., fuzzy subsets of the product of domains [7, 37]. Thus, each tuple in a relation is associated with a membership grade. For example, the tuple *(john, pascal)* belongs to the relation *Proficiency(Programmer, Language)* with membership grade 0.9. Associating a membership grade with each tuple may be regarded as a statement of *uncertainty*.

Alternatively, the same tuple may be interpreted as stating that John's proficiency in Pascal is 0.9. In this interpretation, the membership grades indicate the strength of the association between the components of the tuple (in this case, a programmer and a programming language). These different interpretations should not be confused, and must be taken into account when defining operations for manipulating fuzzy relations.

The theory of possibility is based on fuzzy set theory. In a relational framework, the value of a particular attribute A for a specific tuple t is a variable t_A, and this variable has an associated *possibility distribution* $\pi_{A(t)}$. $\pi_{A(t)}$ assigns values in the range $[0, 1]$ to the elements of the domain of attribute A. Using the same

variable $Age(john)$, an example of a possibilistic distribution is

$$\pi_{Age(john)} = \left\{ \begin{array}{ll} 30 & 1.0 \\ 31 & 1.0 \\ 32 & 1.0 \\ 33 & 0.7 \\ 34 & 0.5 \\ 35 & 0.2 \end{array} \right.$$

The interpretation of this information is that it is completely possible that the age of John is 30, 31, or 32, it is very possible that it is 33, it is somewhat possible that it is 34, it is remotely possible that it is 35, and it is completely impossible that it is any other age. If this individual possibility distribution is assigned a name; e.g., *early_thirties*, then it is also possible to interpret it as a definition of the linguistic term *early thirties*: it is a term that refers to 30–32 year olds with possibility 1.0, to 33 year olds with possibility 0.7, etc. Thus, possibility distributions may be used to describe vague linguistic terms.

Consider now standard relations, but assume that the elements of the domains are not values but possibility distributions [48, 36]. This provides the basis for an alternative fuzzy database model. Having possibility distributions for values permits specific cases where a value is one of several kinds: (1) A vague term, for example, a value of *Age* can be *early_thirties*; (2) a disjunctive value, for example, a value of *Department* can be {*shipping, receiving*}, or a value of *Salary* can be *40,000–50,000*; (3) a null value; (4) a simple value. Our earlier observation that probabilistic information expresses both uncertainty and imprecision applies to possibilistic information as well.

To manipulate fuzzy databases, the standard relational algebra operators must be extended to fuzzy relations. The first approach, in which relations are fuzzy sets but elements of domains are crisp, requires simple extensions to these operators. The second approach, where relations are crisp but elements of domains are fuzzy, introduces more complexity because the softness of the values in the tuples creates problems of value identification (e.g., in the join, or in the removal of replications after projections). Also, in analogy with standard mathematical comparators such as $=$ or $<$, which are defined via sets of pairs, the second approach introduces fuzzy comparators such as *similar-to* or *much-greater-than*, which are defined via fuzzy sets of pairs. These fuzzy operators offer the capability of expressing fuzzy (vague) retrieval requests.

The use of fuzzy set and possibility theory to represent and manipulate imperfect information is discussed in more detail in Chapter 10. Fuzziness in databases and information-retrieval systems is also the subject of [6]. Although

their basic axioms and much of their semantics are very different, probability theory, possibility theory, and confidence factors share a common intuitive concept, namely, they all *qualify* database values with numbers that attempt to describe the likelihood that these specific values are indeed the correct ones.

3.5 Information Distances

An approach to imprecision that has been applied successfully to both database systems and information-retrieval systems handles imprecise information with *distance*. The basic idea is to model the real world with apparently precise descriptions, to define the notion of distance between any two descriptions, and thus to create *neighborhoods* of descriptions.

Thus, any imprecision about a real-world object o is ignored, and an apparently precise description of it e is stored. It is then hoped that this "negligence" would be compensated for by having e somewhere in the neighborhood of the true description. When a request for information specifies this true description, e would be retrieved along with the other neighbors of the true description.

As an example, consider an information-retrieval system that describes documents with sets of keywords [41, 45]. Such systems often represent keyword sets with vectors: the dimension of each vector is the number of possible keywords, and a specific vector position is 1 if a particular keyword is in the set and 0 otherwise. Often, there is uncertainty about whether a specific vector is the true description of a given document. By establishing a distance among document descriptions, usually with some vector metric, and retrieving all the information in the neighborhood of a request vector, the negative effects of imprecisions in the description are diminished. Refer to Chapter 7 for a more detailed discussion of information-retrieval models.

As another example, consider relational database systems. Such systems describe objects with tuples, and often there is uncertainty regarding the value of some attribute in a given tuple. It is possible to establish a distance among the elements of the domain of this attribute. Then, when a query specifies a value of this attribute, all the tuples would be retrieved whose values for that attribute are in the neighborhood of the specified value [28].

We assumed here that descriptions are subject to imprecision (which is ignored) and requests are precise. The same solution applies when descriptions are precise and requests are imprecise. Such requests would be specified

with apparent-precision and would be answered with the neighborhood of the request. Again, the negative effects of imprecision in the request would be moderated. Imprecise requests are discussed in more detail in Section 4.1.

A refinement of this general method is to admit confidence factors (Section 3.2) in the descriptions and in the requests. For example, vectors describing keyword sets use values in the range [0,1] to denote the confidence that a specific keyword is relevant to the given document (or, alternatively, to denote the degree of relevance of this keyword). Similarly, request vectors use such values to denote the relative weights of various keywords in the overall request.

3.6 Soundness and Completeness

The final approach that we discuss [29] does not attempt to model imperfect information; instead, declarations are made of the portions of the database (i.e., *views*) that are prefect models of the real world (and thereby the portions that are possibly imperfect).

Thus, like distances (Section 3.5) and unlike disjunctive values (Section 3.1), confidence factors (Section 3.2), probabilistic values (Section 3.3) or possibilistic values (Section 3.4), the descriptions themselves have no special features for representing imperfection (i.e., they appear perfect). However, meta-information provides the distinction between perfect and imperfect information.

This approach interprets perfectness, which it terms *integrity*, as a combination of *soundness* and *completeness*. A description is sound if it includes *only* information that occurs in the real world; a description is complete if it includes *all* the information that occurs in the real world. Hence, a description has integrity if it includes the whole truth (completeness) and nothing but the truth (soundness).

With this information included in the database, the database system can *qualify* the perfectness of the answers it issues in response to queries: each answer is accompanied by statements that define the portions (i.e., views) that are guaranteed to be perfect. A technique is described in [29] for inferring the views of individual answers that are guaranteed to have integrity, from the views of the entire database that are known to have integrity.

The notion of view completeness is similar to an assumption that a certain view of the database is *closed world* [38]. Also, *open nulls* [16] are actually

declarations of views that are incomplete. The notion of view soundness is shown to be a generalization of standard database integrity constraints.

View soundness and view completeness have been extended in [33] to measure *relative* soundness and completeness of views (with respect to the real world). Let v denote the extension of some view in the database and let v_0 denote the real-world extension of the same view. Let $|v|$ denote the number of tuples in v. Then

$$\frac{|v \cap v_0|}{|v|}$$

expresses the proportion of the database answer that appears in the true answer. Hence, it is a measure of the soundness of v. Similarly,

$$\frac{|v \cap v_0|}{|v_0|}$$

expresses the proportion of the true answer that appears in the database answer. Hence, it is a measure of the completeness of v. Relative soundness and completeness are similar to the *precision* and *recall* measures used in information retrieval [41, 45], but whereas precision and recall measure the goodness of retrieved information relative to the stored information, soundness and completeness measure the goodness of stored information relative to the real information. In other words, they measure the *quality* of information sources. These measures are used to guide the *harmonization* of inconsistent answers in a multidatabase environment [32, 33].

4 IMPERFECT MANIPULATION AND PROCESSING

Although most of the work on imperfect information has focused on description imperfection, transaction and processing imperfection also have important impact on the quality of the information delivered to users. In this section we discuss briefly issues and solutions that concern imperfections in the definition of transformations (e.g., queries), in the definition of modifications (e.g., updates or restructuring operations), and in the processing of such transactions.

4.1 Transformations

Transformations are operations that derive new descriptions from stored descriptions. The most frequent type of transformation are queries. Imperfect queries may occur for different reasons.

At times, users of information systems have insufficient knowledge of the information system they are using: they might not have a clear idea of the information available in the system (or how it is organized), or they might not know how to formulate their requests using the tools provided by the system.

Requests for information formulated by such naive users often exhibit a high level of imperfection. They range from requests that cannot be interpreted by the system (for reasons that are either syntactical or semantical) to requests that do not achieve correctly the intentions of the users (or achieve them only in part).

Regardless of their level of expertise, occasionally users may try to access an information system with only a *vague* idea of the information they seek. For example, a user may be accessing an electronic catalogue for a product that would be "interesting" or "exceptional." Alternatively, users could have a clear idea of the information they want but might lack the information necessary to specify it to the system. An example is a user who wishes to look up the meaning of a word in a dictionary but cannot provide its correct spelling.

To summarize, we distinguish among (1) insufficient knowledge of the *information available* (or how it is organized), (2) vagueness with respect to the *information needed* (or how to denote it in terms acceptable to the system), and (3) insufficient knowledge of the *system languages and tools* that are used to formulate requests.

To address all these, the approach has been to develop alternative access tools. Browsers allow users to access information in either situation discussed above [27, 39, 2, 13]. Interactive query constructors conduct user-system dialogues to arrive at satisfactory formulations of user requests [47]. Vague query processors allow users to embed imprecise specifications in their requests (e.g., neighborhood queries, as described in Section 3.5) [21, 28]. Error-tolerant interfaces use relaxed formalisms in their interpretation of requests [30].

As previously mentioned, even after a request for information had been accepted by the system and its answer delivered to the user, uncertainty might

still persist because it is not always possible to verify that the request indeed achieved correctly the intention of the user. Often, the only assurance that the information delivered is the information needed is that the user is somewhat familiar with it. In the absence of such familiarity, uncertainty will persist.

Hence, one must accept that there are frequent instances where the answer that is delivered is imperfect, yet both the system and the user are unaware of this imperfection. Often, the imperfection of apparently perfect answers is revealed only when a conflicting answer to the same request is received from another information system.

4.2 Modifications

Modifications (update and restructuring) are operations that affect the descriptions stored in information systems. Like transformations (queries), modifications are defined by users, and we identify three similar main sources of imperfection: (1) insufficient knowledge of the system and its tools, (2) insufficient knowledge of the contents and organization of information stored in the system, and (3) uncertainty or imprecision concerning the information embedded in a modification.

Many of the approaches aimed at alleviating the problems of transformation imperfections (Section 4.1) are also applicable to modification imperfections. However, fewer tools have been developed to address modification imperfections. A possible explanation is that, whereas modifications may be attempted by users at all levels of expertise, it is often assumed that users who modify information should have good familiarity with the system.

The third source of imperfection, uncertain or imprecise information, is unrelated to expertise. One example is a request to add imprecise information; for example, "the new manager is either Paul or John." Another example is an imprecisely-specified request to delete information; for example, "some of the telephone numbers are no longer valid."

However, this kind of imperfection is not any different from the description imperfections that were the subject of Sections 2 and 3. Thus, the first request would be accommodated as any information of the kind "exactly one of the following values holds," and the second request would be accommodated as any information of the kind "some of the following values hold." (Of course,

if the system cannot model these kinds of imperfection, then it would not be able to handle these modifications.)

4.3 Processing

Even when a description D and a transformation t are free of imperfections, the result $t(D)$ may be imperfect because of the methods used by the system to process requests. In certain applications, an information system might allocate only limited computational resources to process a request [22]. For example, a recursive query to a genealogical database to list all the ancestors of a specific individual might be terminated after a predetermined period of time (presumably the number of ancestors retrieved by then would be sufficient). A query processor that provides monotonically improving partial answers under constraints of time or unavailability is described in [46]. In other applications, query processing might involve randomizations, sampling, or other estimation techniques (see Chapter 5). For example, a statistical database system might introduce perturbations into its answers deliberately, for reasons of security. In each case, the answers would exhibit imperfections.

Finally, it is sometime considered advantageous to sacrifice accuracy for the sake of *simplicity*. Recent research on *intensional answers* [31] has focused on the generation of abstract answers that describe the exhaustive answers compactly, albeit imperfectly. For example, a query to list the employees who earn over $50,000 might be answered simply and compactly "engineers," even when the set of engineers and the set of employees who earn over $50,000 are not exactly the same (e.g., when the two sets overlap substantially, or when one set contains the other).

5 CHALLENGES

Commercial information systems have been slow to incorporate capabilities for managing imperfect information. While solutions based on fuzzy set theory (Section 3.4) are gaining acceptance in several technologies, commercial information systems have not embraced this solution yet. In database systems, the only capabilities widely available are for handling null values (Section 3.1), and for specifying imprecise queries with regular expressions. Most commercial information-retrieval systems do not have the capabilities that come with the definition of distances (Section 3.5). The situation is somewhat better in expert

systems, in which commercial systems often permit certainty factors in their facts and rules (Section 3.2).

Examining the reasons for this slow acceptance may suggest directions for further research, and we offer here several possible reasons. First, information systems practitioners are concerned primarily with the *performance* of their systems. Here, many of the algorithms for matching uncertain data or for processing imprecise requests are extremely complex and inefficient.

Practitioners are also concerned with *compatibility*. This dictates that capabilities for managing uncertainty and imprecision should be offered as strict extensions of existing standards. Additionally, database practitioners have often been dissatisfied with various *idiosyncratic implementations* of such capabilities (minor examples are the inability to specify an incomplete date value, or the inability to sort answers with null values flexibly). This may have had a chilling effect on further implementations.

Another hindrance for database systems with capabilities for managing imperfect data may lie in the *expectations of users*. A fundamental principle of database systems has been that queries and answers are never open to subjective interpretations, and users of database systems have come to expect their queries to be interpreted unambiguously and answered with complete accuracy. In contrast, users of information-retrieval systems would be pleased with a system that delivers high rates of recall (proportion of relevant material retrieved) and precision (proportion of retrieved material that is relevant). Similarly, users of expert systems are aware that conclusions offered by automated experts are often questionable and would be satisfied with systems that were shown to have high rates of success. A database system that must adhere to the principles of unambiguity and complete accuracy cannot accommodate the full range of imperfect information; for example, it can accommodate null values or disjunctive data but not (because of its more subjective nature) probabilistic or possibilistic data.

Recently, new applications have emerged that require information systems with capabilities for managing imperfect information. Several of these challenging applications are described next. Their description is followed by a discussion of a possible source of suitable technology, and the challenges involved in adapting it for information systems.

5.1 Heterogeneous Database Environments

In recent years, the integration and interchange of information among heterogeneous database systems has been recognized as an important area of database system research and development. Multidatabase environments present a strong case for having uncertainty management capabilities. Even though individual database systems are usually careful to avoid internal inconsistencies (mostly with methodical design that avoids repeating the same information in multiple locations in the system), when information from independent systems with overlapping information is integrated, inconsistencies often surface. Thus, even when individual answers are free from imperfections, their integration in a global answer would introduce inconsistency. Therefore, database systems in a multidatabase environment should be capable of combining conflicting answers into a single, consistent answer, and then store and manipulate such information [32, 33].

5.2 Multimedia Databases

As discussed in Section 3, retrieval from traditional databases is usually based on *exact matching*, where a request for data establishes a specific retrieval goal, and the database system retrieves the data that match it exactly. Presently, the management of image databases largely follows the same paradigm: images are stored (in digitized form) in large databases but retrieval is performed on *textual descriptions* of these images that are stored along with the images themselves.

A more ambitious approach, such as IBM's Query By Image Content (QBIC) [15], is to retrieve images that match a given *image*. Image matching techniques are usually based on algorithms of *best matching*. As with the information-retrieval systems discussed earlier, the use of uncertainty formalisms is essential. A similar problem is to match handwritten addresses against a database of addresses.

5.3 Imputation and Knowledge Discovery

In various applications, notably in scientific and statistical projects, it is necessary to estimate missing data (nulls) from other data that are available. For example, a missing measurement is estimated by other measurements made by the same instrument at other times as well as by measurements made by other instruments at the same time. This process, usually referred to as *imputation*,

yields information of varying degrees of uncertainty. The management of this information cannot be done by traditional database techniques and requires the use of uncertainty techniques.

The inference of missing values from their contexts is related to the more general issue of discovering new knowledge in large databases. Database knowledge is usually *declared*: rules and constraints are assumed to be definite and accurate and to hold in any database instance (exceptions, if any, must be stated). In contrast, *discovered* knowledge is always subject to uncertainty: the patterns and rules are often tenuous and could be refuted by future data. Again, the management of such information requires uncertainty techniques.

5.4 Information Systems and Artificial Intelligence

Modeling imperfect information in information systems involves a classical dilemma. On one hand, we want a model as rich and powerful as possible; for example, a single concept of information that is capable of representing elegantly all known kinds of imperfect information as well as perfect information. On the other hand, database systems must abide by crucial constraints of simplicity and efficiency, both of their descriptions and of the manipulations of their descriptions.

Like information systems, the field of artificial intelligence too has been concerned with modeling accurately our knowledge of the real world. Numerous uncertainty theories have been developed. Mostly they are founded on nonclassical logics, on probability theory, on belief functions, or on possibility theory [4, 42].

It might be said that, traditionally, research in artificial intelligence has been emphasizing rich and powerful models, striving to model accurately all the minute nuances of reality. On the other hand, research in information systems has been emphasizing economical representations with lower expressiveness but with small representational overhead and high processing efficiency.

A notable case in point are the so-called *semantic data models* that were defined in the late 1970s to enhance the modeling capabilities of early database systems [17, 19]. These models incorporated modeling features, such as generalization and aggregation hierarchies, that had been adapted from research in artificial intelligence. An even earlier example is the adaptation of *semantic networks* for

databases [40]. A third and more recent example are *knowledge-rich database systems*, that incorporate inference rules expressed in mathematical logic [43, 44, 9]. Not surprisingly, a major research thrust among database researchers that have been working in this area has been the development of *highly efficient* inference techniques for a *limited* class of rules.

Thus, information systems may be said to have embraced "poor-person's AI," or, more graciously put, to have adapted AI concepts to work under the strict constraints of information systems. The adaptation of uncertainty theories that have been developed for artificial intelligence to operational techniques for information systems is an important challenge for information systems researchers. This challenge is the subject of Chapters 8–14.

Acknowledgments

This work was supported in part by NSF grant No. IRI-9007106 and by ARPA grant, administered by the Office of Naval Research under grant No. N0014-92-J-4038.

REFERENCES

[1] S. Abiteboul and G. Grahne. Update semantics for incomplete databases. In *Proceedings of the Eleventh International Conference on Very Large Data Bases* (Stockholm, Sweden, August 21–23), pages 1–12. Morgan Kaufmann, Los Altos, California, 1985.

[2] R. Agrawal, N. H. Gehani, and J. Srinivasan. OdeView: The graphical interface to Ode. In *Proceedings of ACM-SIGMOD International Conference on Management of Data* (Atlantic City, New Jersey, May 23–25), pages 34–43. ACM, New York, New York, 1990.

[3] D. Barbara, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Transactions on Knowledge and Data Engineering*, 4(5): 487–502, October 1992.

[4] Lea Sombe (P. Besnard, M.-O. Cordier, D. Dubois, L. Farinas del Cerro, C. Froidevaux, Y. Moinard, H. Prade, C. Schwind, and P. Siegel). Special issue, Reasoning under incomplete information in artificial intelligence. *International Journal of Intelligent Systems*, 5(4), September 1991.

[5] A. Borgida. Language features for flexible handling of exceptions in information systems. *ACM Transactions on Database Systems*, 10(4): 565–603, December 1985.

[6] P. Bosc and J. Kacprzyk, editors. *Fuzziness in Database Management Systems*. Physica-Verlag, Heidelberg, Germany, 1995.

[7] B. P. Buckles and F. E. Petry. A fuzzy representation of data for relational databases. *Fuzzy Sets and Systems*, 7(3): 213–226, May 1982.

[8] R. Cavallo and M. Pittarelli. The theory of probabilistic databases. In *Proceedings of the Thirteenth International Conference on Very Large Data Bases* (Brighton, England, September 1–4), pages 71–81. Morgan Kaufmann, Los Altos, California, 1987.

[9] S. Ceri, G. Gottlob, and L. Tanka. *Logic Programming and Databases*. Springer-Verlag, Berlin, Germany, 1990.

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

[11] C. J. Date. *Relational Database: Selected Writings*. Addison Wesley, Reading, Massachusetts, 1986.

[12] C. J. Date. NOT is not "not"! In *Relational Database Writings 1985–1989*. Addison Wesley, Reading, Massachusetts, 1990.

[13] A. D'Atri, A. Motro, and L. Tarantino. ViewFinder: an object browser. Technical report 95-115, Department of Information and Software Systems Engineering, George Mason University, February 1995.

[14] R. Demolombe and L. Farinas del Cerro. An algebraic evaluation method for deduction in incomplete data bases. *Journal of Logic Programming*, 5: 183–205, 1988.

[15] C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petkovic, and R. Barber. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4), July 1994.

[16] G. Gottlob and R. Zicari. Closed world assumption opened through null values. In *Proceedings of the Fourteenth International Conference on Very Large Data Bases* (Los Angeles, California, August 25–28), pages 50–61. Morgan Kaufmann, Los Altos, California, 1988.

[17] M. Hammer and D. McLeod. Database description with SDM: A semantic database model. *ACM Transactions on Database Systems*, 6(3): 351–386, September 1981.

[18] P. Harmon and D. King. *Expert Systems—Artificial Intelligence in Business.* John Wiley & Sons, New York, New York, 1985.

[19] R. Hull and R. King. Semantic database modeling: Survey, applications and research issues. *Computing Surveys*, 19(3): 201–260, September 1987.

[20] A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors. *Multidatabase Systems: An Advanced Solution for Global Information Sharing.* IEEE Computer Society Press, Los Alamitos, California, 1994.

[21] T. Ichikawa and M. Hirakawa. ARES: A relational database with the capability of performing flexible interpretation of queries. *IEEE Transactions on Software Engineering*, SE-12(5): 624–634, May 1986.

[22] T. Imielinski. Intelligent query answering in rule based systems. *Journal of Logic Programming*, 4(3): 229–257, September 1987.

[23] T. Imielinski. Incomplete information in logical databases. *Data Engineering*, 12(2): 29–40, June 1989.

[24] T. Imielinski and K. Vadaparty. Complexity of query processing in databases with or-objects. In *Proceedings of PODS-89, the 8th Symposium on Principles of Database Systems* (Philadelphia, PA, March 29–31), pages 51–65, 1989.

[25] D. Maier. *The Theory of Relational Databases.* Computer Science Press, Rockville, Maryland, 1983.

[26] J. Minker. On indefinite databases and the closed world assumption. In *Lecture Notes in Computer Science No. 138*, pages 292–308. Springer-Verlag, Berlin, Germany, 1982.

[27] A. Motro. BAROQUE: A browser for relational databases. *ACM Transactions on Office Information Systems*, 4(2): 164–181, April 1986.

[28] A. Motro. VAGUE: A user interface to relational databases that permits vague queries. *ACM Transactions on Office Information Systems*, 6(3): 187–214, July 1988.

[29] A. Motro. Integrity = validity + completeness. *ACM Transactions on Database Systems*, 14(4): 480–502, December 1989.

[30] A. Motro. FLEX: A tolerant and cooperative user interface to databases. *IEEE Transactions on Knowledge and Data Engineering*, 2(2): 231–246, June 1990.

[31] A. Motro. Intensional answers to database queries. *IEEE Transactions on Knowledge and Data Engineering*, 6(3): 444–454, June 1994.

[32] A. Motro. Multiplex: A formal model for multidatabases and its implementation. Technical Report ISSE-TR-95-103, Department of Information and Software Systems Engineering, George Mason University, March 1995.

[33] A. Motro and I. Rakov. Estimating the quality of data in relational databases. In *Proceedings of the 1996 Conference on Information Quality* Cambridge, Massachusetts, October 25–26, 1996.

[34] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.

[35] G. Piatetsky-Shapiro. Discovery, analysis and presentation of strong rules. In G. Piatetsky-Shapiro and W. Frawley, editors, *Knowledge Discovery in Databases*, pages 229–248. AAAI Press/MIT Press, Menlo Park, California, 1991.

[36] H. Prade and C. Testemale. Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. *Information Sciences*, 34(2): 115–143, November 1984.

[37] K. V. S. V. N. Raju and A. Majumdar. Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. *ACM Transactions on Database Systems*, 13(2): 129–166, June 1988.

[38] R. Reiter. On closed world data bases. In *Logic and Databases*, pages 55–76. Plenum Press, New York, New York, 1978.

[39] T. R. Rogers and R. G. G. Cattell. Object-oriented database user interfaces. Technical report, Information Management Group, Sun Microsystems, 1987.

[40] N. Roussopoulos and J. Mylopoulos. Using semantic networks for data base management. In *Proceedings of the First International Conference on Very Large Data Bases* (Framingham, Massachusetts, September 22–24), pages 144–172. ACM, New York, New York, 1975.

[41] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, New York, 1983.

[42] P. Smets and M. R. B. Clarke (Guest Editors). Special issue, Uncertainty, conditionals and non-monotonicity. *Journal of Applied Non-Classical Logics*, 1(2), 1991.

[43] J. D. Ullman. *Database and Knowledge-Base Systems, Volume I.* Computer Science Press, Rockville, Maryland, 1988.

[44] J. D. Ullman. *Database and Knowledge-Base Systems, Volume II.* Computer Science Press, Rockville, Maryland, 1989.

[45] C. J. van Rijsbergen. *Information Retrieval (Second Edition).* Butterworths, London, 1979.

[46] S. V. Vrbsky and J. W. S Liu. APPROXIMATE: a query processor that produces monotonically improving approximate answers. *IEEE Transactions on Knowledge and Data Engineering*, 5(6): 1056–1068, December 1993.

[47] M. D. Williams. What makes RABBIT run? *International Journal of Man-Machine Studies*, 21(4): 333–352, October 1984.

[48] M. Zemankova and A. Kandel. Implementing imprecision in information systems. *Information Sciences*, 37(1, 2, 3): 107–141, December 1985.