NON-MONOTONIC KNOWLEDGE EVOLUTION IN VLKDBs

Christian ESCULIER
Laboratoire de Génie Informatique B.P. 53x - 38041 GRENOBLE - France

Abstract

Non-monotonic knowledge evolutions and exceptions constitute a complex theoritical and practical problem. The state of the art shows a rich and surprising diversity of approaches. Their study along a common framework reveals a few basic issues which require further theoritical investigation.

In the context of very large knowledge/data bases (VLKDBs), specific constraints must be taken into account. They require a specific approach of the problem, which can be sketched by a set of basic guidelines.

In order to illustrate the importance of the problem, a proposal called semantic tolerance is briefly described. Its various relationships with user interfaces, rule administration, advanced design techniques and long-term transactions show that non-monotonic knowledge evolution constitute a central problem in advanced databases, and consequently would deserve a major research effort by the database community.

1 Introduction

A large part of the current theoritical research activities in the database area attempts to capture more semantics about some universe of discourse, which itself can be more and more complex. All these activities have as a common consequence the extension of the knowledge incorporated into and managed by advanced database systems (what motivates to use the terms knowledge/data bases – KDBs – and knowledge/data base systems – KDBSs).

Proceedings of the 16th VLDB Conference
Brisbane, Australia 1990

The problem discussed in this paper concerns the non-monotonic evolution of this knowledge, and more precisely situations where an instance and a rule are in conflict – not because the instance is an error, but because the rule is imperfect. These situations are almost ignored in the database paradigm, where the universe of discourse and its behaviour are considered as regular. This traditional assumption cannot be made anymore by the database research community.

In Very Large Knowledge/Data Bases (VLKDBs), the number of rules – encompassing statics and dynamics – can be of several thousands, the number of primitive instances (provided by users) can be of several millions or billions, and the population of users is both large (thousands to millions) and highly heterogeneous (in expertise and in reliablity).

In such context, the non-monotonic evolution of knowledge becomes a very important theoritical and practical issue. Its theoritical importance is clearly illustrated by the effort made by the Artificial Intelligence (AI) community to address the problem. Unfortunately, the solutions proposed in AI do not seem easily transposable to VLKDBs. Its practical importance is illustrated by the classical 90/10 ratio advocated by practitioners, which says that 90 % of the work in a data processing center is dedicated to solve the remaining 10 % of the cases, which are not regular.

The basic objectives of the paper are :

– to sketch an investigation of the problem and of the various proposed solutions, in order to show its complexity and the need for specific research efforts in the context of VLKDBs;

– to sketch a proposal (called semantic tolerance) which addresses non-monotonic knowledge evolution in VLKDBs and which could serve as a basis for future research and developments.

It is clearly hopeless to try to address the problem in depth in ten pages. Consequently, this paper is only a short introduction which attempts to emphasize its importance and its complexity. Semantic tolerance – which is briefly described – is essentially intended to illustrate how non-monotonic knowledge evolution interferes with classical engineering techniques and with

user interfaces, rule administration long-term transactions and database design techniques.

The paper is structured as follows :

- section 2 proposes a basic characterization of non-monotonic knowledge evolution.
- section 3 gives an overview of the various approaches which can be found in Programming Languages (PL), in Databases and in AI. These approaches (usually kept clearly separated) can be situated along some kind of *continuum*, which shows a progressive transformation of the problem;
- section 4 reconsiders non-monotonic knowledge evolution and proposes some guidelines to handle it in VLKDBs;
- section 5 describes the basic elements of semantic tolerance and indicates its relationships with various proposals of the state of the art and with aspects of knowledge and data management, which at first glance do not seem related with non-monotonicity;
- section 6 concludes the paper.

2 Knowledge evolution and non-monotonicity

With perfect rules, the evolution of a KDB corresponds to a continuous monotonic growth (M. Ginsberg characterizes monotonicity in saying : *"As the set of beliefs grows, so does the set of conclusions that can be drawn from those beliefs."* [GINS 87]).

This ideal situation is unfortunately very rare in practice. Most of the time, the evolution is much more complex. Instances – which model appropriately the universe of discourse, and consequently which are not errors – do not satisfy one or several rules because these rules are imperfect. It can be, for example, because they do not comply to some integrity constraint, or because they contradict some derived instance(s).

Such instances, which are not errors, and which do not satisfy the rule base are usually called **exceptions** (cf the Oxford Dictionary – *"Exception : something which does not follow the rule"*). The evolution of knowledge when an exception occurs is a **non-monotonic evolution** : new knowledge requires to reconsider old conclusions. The classical example to investigate this problem is the *Tweety* example :

- a derivation rule says that birds can fly :

$$r_1 = (\forall x,\ \mathrm{BIRD}(x) \rightarrow \mathrm{FLY}(x))$$

- a primitive instance says that *Tweety* is a bird :

$$p_1 = \mathrm{BIRD}(Tweety)$$

- the theorem derived from these two axioms says that *Tweety* can fly :

$$d_1 = \mathrm{FLY}(Tweety)$$

- but the observation of the universe of discourse contradicts this conclusion. The KDB should actually contain the primitive instance :

$$p_2 = \neg\ \mathrm{FLY}(Tweety)$$

and "forget" the wrong derived instance d_1 (to preserve its consistency). This example shows that the evolution of knowledge, when the case of *Tweety* is considered, requires to take into account an instance which models appropriately the universe of discourse, but which does not satisfy the considered rule because the rule is imperfect. (Similar examples could be given for integrity constraints or behaviour rules instead of derivation rules). This basic characterization of non-monotonic knowledge evolutions and exceptions can be used as a starting point to study the various proposals of the state of the art. These proposals provide a rich background and emphasize a variety of aspects of the problem. Consequently, after the review of the various proposals, it will be possible to provide a more sophisticated characterization of non-monotonicity of knowledge evolution, in order to address it in the specific context of VLKDBs.

3 A state of the art

The non-monotonic evolution of knowledge is studied in various areas. The main ones are programming languages, databases and artificial intelligence, this last area being by far the richest one [WIED 88], [ESCU 88b].

3.1 PLs : exception handlers and overriding

In classical programming languages, there is no way to take into account an exception if the exception is not anticipated. It means that actual exceptions cannot be handled, because as soon as an exception is anticipated, it is not an exception anymore ! The traditional approach is to define specific pieces of code – *exception handlers* – which are called when the corresponding "exception" is encountered [GOOD 75], [CRIS 79], [LISK 79], [LUCK 80], [CRIS 82].

In object-oriented programming languages [PIWI 88] and OODBs – [COMA 84], [ANHA 87], [LERI 89] in particular –, the most important feature

related with this problem is the possibility to redefine a method, a structure or an attribute value when the definition contained in the (super-)type is not satisfactory. This technique, called *over-riding*, is extremely powerful but it means that rules are very weak and can easily be contradicted either by other rules or by specific instances. Consequently, over-riding is a very interesting programming technique, but in order to deal with exceptions as such it would be necessary to enforce some control over its use. Otherwise, exceptions are considered as quite ordinary instances.

3.2 DBs : rule base dynamic revision

In the database area, exceptions are most of the time considered as a secondary problem. Consequently, it is usually addressed as a side effect of a more general approach. As an example, M. Kersten [KERS 86] shows how the mechanism of demon can be used to deal with exceptions (among other problems). As another example, R. Ramirez [RAMI 88] uses derived relations to host exceptions (among other elements).

The only team which addressed quite specifically the problem of exceptions in DBs is the database team of Rutgers University, with A. Borgida, T. Mitchell and K. E. Williamson ([BORG 85], [BOWI 85], [BOMW 86], [BORG 87]).

Their approach is extremely interesting because it emphasizes the importance of the end-user in the knowledge evolution process : when a user submits an instance which does not comply to an integrity constraint, if the user confirms the submitted instance, the system takes it into account, stores it permanently in the database and revises the rule to exclude the exceptional instance. Consequently, the revised r_1 rule when *Tweety* is known as an exceptional bird, is reformulated by the system as :

$$r_1' = (\forall x, x \neq \textit{Tweety},\ \text{BIRD}(x) \rightarrow \text{FLY}(x))$$

Formally, this approach does not fit in the mathematical framework of non-monotonicity because the new set of axioms is not a super-set of the old one. There is a permanent revision of the rule base, which is modified each time an exception occurs.

Consequently, a major problem with this proposal is that the rule base is not stable. Another problem too is that the control of exceptional instances relies only on the user confirmation. It should nevertheless be considered as a key contribution, because it took a database perspective, and because it provided very interesting investigations of the link between exceptions and progressive database design [BOWI 85].

3.3 AI : non-monotonicity and uncertainty

Most of the theoritical works on exceptions take place in artificial intelligence. Two major streams of research can be identified : a stream which starts from logics – investigating the problem under the general topic of non-monotonicity – and a stream which starts from probability – under the topic of uncertainty.

The first stream proposes various solutions which either preserve the basic truth values True and False, or increase the number of truth values (directly or indirectly) on the basis of quite distinct motivations and formal techniques [ISRA 80], [PERL 86]. A very interesting synthesis and compilation of papers representing major contributions in this stream is done by M. Ginsberg in [GINS 87].

The second stream is usually considered as clearly separated from investigations on non-monotonicity because it uses numerical truth values.

In fact, the two streams can be perceived altogether along a continuum of solutions which constitutes a nice basis to study the problem of knowledge evolution in VLKDBs. The proposals are briefly presented in what follows along this continuum, from 2-value logics till numerical approaches.

3.3.1 AI : abnormality predicates and defaults

The only proposal in the orthodoxy of 2-value logics is due to J. McCarthy [McCA 80] with the concept of *abnormality predicate*. The idea is that the considered rule applies to an instance only if this instance is normal, or more precisely if the instance is not abnormal. Formally, the rule about birds becomes :

$$r_1^* = (\forall x, \neg AB(x) \wedge \text{BIRD}(x) \rightarrow \text{FLY}(x))$$

where AB is the predicate which is used to say that a given instance is abnormal.

A major advantage of this proposal is that the rule does not need to be revised when an exception occurs.

A major problem is that the predicate AB must be evaluated against all instances of BIRD, and the key issue is to "minimize" the set of exceptions [LIFS 85], [McCA 86]. If the value of $AB(\textit{Tweety})$ is not explicitly provided, the evaluation of the rule requires either the use of the Closed World Assumption (CWA) or an extra truth value to indicated the absence of knowledge. (Remark : this research is the starting point of circumscription and goes much further than non-monotonicity. It provides a nice illustration of the formal importance of the problem.)

Another approach which was originally proposed in 2-value logics is due to R. REITER with the *default logic* [REIT 80]. The basic idea is to extend Modus Ponens in order to draw conclusions in the absence of information only if these conclusions are not in contradiction with some knowledge in the KDB.

In the example, the rule is formulated as follows :

$$r_1^* = (\forall x(\ \mathrm{BIRD}(x) \wedge \mathsf{M}\ \mathrm{FLY}(x)) \rightarrow\ \mathrm{FLY}(x))$$

where M is a meta-operator which interferes with the inference rule and must be interpreted as "in the absence of any contradictory information".

The most interesting feature in this proposal is that it introduces in the rule definition an element with reduces its strength. The inference rule is not any more the plain Modus Ponens because it must be able to auto-criticize its results. Conclusions produced by this weak rule are either withdrawn (when they are in conflict with some already existing instance(s)) or established as ordinary conclusions (when there is no conflict). A major problem with this approach is that default rules themselves can be in conflict [PLET 88]. It shows that the hierarchy introduced by the meta-operator is not satisfactory. The default logic was proposed in a 2-value logic by R. Reiter, but it should be related with a multi-value logic. M. Ginsberg shows in [GINS 86] that it requires in fact three extra truth values : TRUE-BY-DEFAULT, FALSE-BY-DEFAULT and BOTH-TRUE-AND-FALSE-BY-DEFAULT.

3.3.2 AI : modalities, auto-epistemology and justifications

A proposal which has numerous similarities with default logic is due to D. McDermott and J. Doyle, who use a *modality* to control consistency in the derived conclusions [McDD 80]. The formalism is :

$$r_1^* = (\forall x(\ \mathrm{BIRD}(x) \wedge \mathcal{M}\ \mathrm{FLY}(x)) \rightarrow\ \mathrm{FLY}(x))$$

where $\mathcal{M}$ is a modal operator, interpreted as "if it is consistent to infer". In the S5 modal system, McDermott and Doyle show that this approach cannot be based on undisputable TRUE and FALSE truth values. They propose to classify formulas as *"doubtless, provable, conceivable and arguable"*. Investigating other modal systems, they extend this list with other values such as *"uncontroversial, undeniable, safe, forseeable, realizable, assumable, ..."*.

Advantages of this proposal are that it relies on a well established formal basis, and that it emphasizes that extra truth values are required because the weakness of the derivation rule impacts the quality of the conclusions. A difficulty of this proposal, in the context of VLKDBs, is that these numerous truth values are very complex to handle.

Modalities have been used by R.C. Moore in another perspective, with the *auto-epistemic logic* [MOOR 84], [MOOR 85].

The basic idea is that knowledge cannot be separated from people who observe the universe of discourse : it relies only on beliefs of such people. As a consequence, the rule about birds becomes :

$$r_1^* = [\forall x(\ \mathrm{BIRD}(x) \wedge \neg\mathcal{L}\neg\ \mathrm{FLY}(x)) \rightarrow\ \mathrm{FLY}(x)]_{u_i}$$

which means that if the user which submits the instance does not believe that the given bird cannot fly, then it will be considered as able to fly. (Further investigations are done by M. Genesereth and N. Nilsson in their *"sentential logics of beliefs"* [GENI 87] and by K. Konolidge [KONO 87].)

Formally, this proposal is very similar to the previous one (the modal operators are related by $\mathcal{L} = \neg\mathcal{M}\neg$), but it emphasizes the importance of the user, what seems to be a major element in investigating knowledge evolution [STAM 86]. Exceptions are due to the imperfection of rules, and this imperfection cannot be abstracted from the persons who play the role of "sensors" with respect to the universe of discourse. This fundamental dimension appears in Borgida's proposal, but it seems to be discarded in McCarthy, Reiter or McDermott/Doyle proposals.

The concept of belief is tightly related with the concept of *justification* or *endorsement*, which leads to approaches where every formula is accompagnied with informations which say how and why the formula is in the KDB. The major proposals in this direction are due to J. Doyle [DOYL 79], J. de Kleer [deKL 86] and P.R. Cohen [COHE 85]. Once more, multiple truth values seem to be required (J. Doyle proposes UNKNOWN and TRANSIENT as extra truth values, the former when neither p nor $\neg p$ are believed, the latter when both are justified). The diversity of justifications or endorsements, as proposed by the authors, is such that it leads to extremely complex KDBs, even with a simple universe of discourse.

3.3.3 AI : uncertainty, possibility and probability

When logics is the starting point in investigating non-monotonic knowledge evolution, the need for multiple truth values [RINE 84] seems to be a common conclusion. Consequently, these non-monotonic approaches

can be connected with works which stem from a numerical expression of truth, where truth values can be as numerous as needed.

In this case, the bird rule could be something like :

$$r_1^* = ((\forall x,\ \mathrm{BIRD}(x) \rightarrow\ \mathrm{FLY}(x)),.8)$$

which can be interpreted in saying that in 80 percent of the cases (actual or potential) birds can fly.

When *Tweety* is "introduced" in the database as a bird, the user must associate a confidence in the assertion which says that it is a bird – example : "(BIRD(*Tweety*), 1)" if the user is absolutely sure that it is a bird –, and the system will combine the numerical truth values of the primitive instance ("1" here) and of the rule (".8" here) in order to associate a truth value to the conclusion which says that *Tweety* can fly.

The various ways to combine truth values of the premisses to elaborate the truth value of the conclusion are a major issue in numerical approaches.

L. Zadeh proposal is based on fuzzy sets : the *possibility theory* [ZADE 78], complemented by a *fuzzy logic* [ZADE 80]. It uses a simple computation of the numerical truth value, called possibility, noted Π :

$$\Pi(\varphi_\alpha \vee \varphi_\beta) = \mathrm{MAX}(\Pi(\varphi_\alpha), \Pi(\varphi_\beta))$$

and

$$\Pi(\varphi_\alpha \wedge \varphi_\beta) = \mathrm{MIN}(\Pi(\varphi_\alpha), \Pi(\varphi_\beta))$$

P. Smets shows – in a humoristic example where a man uses this approach while he is looking for a woman who is both young and rich [SMET 82] – that the possibility theory misses the right choice. To propose a computation of truth values which does not miss it, the author uses very sophisticated techniques – based on T-norms and S-norms. In the precise case under investigation, such sophistication is certainly required (but is probably not sufficient !).

Numerous proposals are based on possibilities (cf for example D. Dubois and H. Prade [DUPR 88] who relate possibility with necessity and with interval analysis [ALHE 83], and, as reference books, [YAGE 82] and [KALE 86]).

As discussed with Smets' proposal, the possibility theory can be seen as an over simplification in the computation of truth values. P. Cheeseman [CHEE 86] shows that the original possibility theory corresponds to a specific case of maximal dependency, and consequently is unsatisfactory in other cases.

Using the interpretation of R.T. Cox which considers "plausibility" as the estimated measure of the probability which would be observed over a long period of time [COX- 79], the well established results of the probability theory is, in principle, applicable to KDBs [SHAF 76]. The problem, of course, is that using probability is much more complex.

4 Guidelines for handling non-monotonicity in VLKDBs

On the basis of these proposals, it is possible : 1) to revise the concept of non-monotonic knowledge evolution – possibly in emphasizing its most fundamental components – and 2) to define guidelines which take into account these fundamental characteristics and the specific environment of VLKDBs.

4.1 Fundamental characteristics of non-monotonic knowledge evolution

The study of the various proposed solutions leads to a variety of perceptions of the problem, which altogether provide a richer perspective. With classical PLs or with current DBMSs, non-conforming instances are simply rejected : actual exceptions (it means non-anticipated non-conforming instances) are not taken into account because considered as errors.

With O-O languages, the other extreme position is observed : over-riding is a basic programming technique, and exceptions have nothing special compared with normal instances.

– Between these extremes, proposals in the database area and in artificial intelligence illustrate a progressive transformation of the problem. Going through rule permanent revision, abnormality predicate, default logic, modal operators, auto-epistemology, justifications and uncertainty, rules and derived instances become softer and softer.

Non-monotonicity appears as a consequence of a more fundamental phenomenon which could be described in saying that an exception reveals the relative quality of a rule compared with the quality of an instance. In classical PLs and DBs, rules are always considered as perfect. In the approaches dealing with non-monotonicity, the knowledge in general – rules and instances – is considered as liable to imperfection. It means that a balance between rule quality and instance quality must be taken into account.

It opens a wide space of investigation between normal knowledge (which conforms to the defined rules)

and errors (which don't). In the following part of this section, an attempt is made to initiate this investigation in the specific context of VLKDBs.

4.2 Basic guidelines

4.2.1 No possible automatic handling

Non-monotonic knowledge evolution is due to the opposition between the possible imperfection of rules and the possible imperfection of instances. Its automatic handling requires either to make the assumption that rules are perfect (what is done in current DBMSs and in classical PLs – even if the expression "exception handler" is used to designate some of the rules), or to make the assumption that instances are perfect (it is the case in McCarthy and Reiter proposals, for example). These assumptions are obviously both inacceptable in most of the real cases, and particularly in VLKDBs where rules capture lots of semantics and where users are very numerous and constitute a highly heterogeneous population with respect to reliability.

In VLKDBs, it is not possible to deal with non-monotonic knowledge evolution in an automatic fashion because neither rules nor instances can constitute a reliable basis for such automatic handling.

4.2.2 A balance between rules administrators and end-users

When rules administrators define the rule base, they are confident in some basic rules – such as unicity rules on identifiers (at least in a first step !) –, but for numerous other rules, they cannot guarantee that they encompass all possible situations of the universe of discourse. The richer the rule base, the weaker its overall "perfection". With a large population of end-users, instances submitted to the system contain usually a high percentage of errors. The basic role of the system is to control these instances in order to guarantee an acceptable overall quality. But this control – based on integrity constraints in current DBMSs – cannot be considered as perfectly liable, because the corresponding rules are not perfect. Consequently, the enforcement of rules over instances cannot lead to the rejection of all non-conforming instances. Because both rules and instances can be imperfect, it is necessary to evaluate the respective quality of both, what means that a balance must be established between rules administrators and end-users. Such balance can be found in Borgida's, Moore's, Doyle's, Cohen's and numerical proposals.

In VLKDBs, rule enforcement must incorporate a contradictory evaluation of rule quality versus instance quality, representing the balance between rules administrators and end-users.

4.2.3 Formalizing imperfection of rules

The formalization of the potential imperfection of a rule definition requires to address two basic issues :

– is it possible to preserve the rule definition unmodified along the knowledge evolution process ?

– is it possible to formulate a rule in a non-absolute fashion without introducing multiple truth values ?

In other words, with a rule which says that all birds can fly, and knowing the possible existence of a bird which cannot fly, is it possible to preserve the rule as it is ? and if the rule is weakened (for example in saying that "most birds can fly"), what about the strength of conclusions which can be drawn from these rules ?

Borgida's approach – which modifies the rule for each exception – does not try to preserve the rule definition. The imperfection of the rule base is made apparent by its unstability, due to the integration of exceptions in rule definitions. In VLKDBs, such permanent revision of the rule base is probably not acceptable because the size and the complexity of the rule base would soon make its management impossible, and because the large population of its users need a firm reference for using the KDB.

McCarthy proposal preserves rule stability and uses ordinary logic – abnormality is handled via a predicate, in conjunction with the basic predicate of the rule –, but the problem is to evaluate the abnormality predicate in the absence of information. Using the Closed World Assumption leads to a major difficulty because it does not correspond to the actual evolution of knowledge [REIT 88].

In the other proposals – which all try to preserve rule stability – there is an attempt to include in the rule definition "something" which reduces its strength. It can be a meta-operator, a modal operator, a justification or a numerical measure of uncertainty.

In order to preserve the stability of the rule base, it seems necessary to incorporate in rule definitions an explicit formal element which represents their imperfection.

4.2.4 Imperfection of conclusions

With "weak" rules, truth values cannot be anymore the undisputable True and False values. This is quite interesting with respect to ordinary instances, which

are – in principle – out of the scope of non-monotonic knowledge evolution. The problem in 2-value logics is clearly stated by R. Bathnagar and L. Kanal in their introduction to [BHKA 86] : *The decision arrived at by the proof procedure says nothing about the amount of uncertainty attached to the decision.* Is it meaningful to draw an absolute conclusion – such as "Coco can fly" – on the basis of a rule such as "most birds fly", even if it is sure that "Coco" is a bird ? This issue is discussed, for example, by Ginsberg about the underlying multiple truth values of Reiter's proposal.

Because knowledge evolution is non-monotonic – most rules cannot be considered as perfect –, conclusions are themselves liable to imperfection, even in normal cases. This phenomenon is quite interesting because it means that even when everything goes well, conclusions must be considered with suspicion.

> *Conclusions obtained on the basis of imperfect rules and imperfect primitive instances are themselves liable to imperfection – even in situations where knowledge evolution is monotonic.*

4.2.5 References to multiple truth values ?

The formalization of the imperfection of conclusions is related with the various truth values of the underlying logics. The state of the art shows a progressive inflation in the number of truth values from Reiter's proposal (five values, as indicated by Ginsberg) to the numerical approaches (with a potential infinite set of values). In a VLKDB, with very numerous users at very different levels of expertise, it seems quite difficult to use fancy truth values. It seems reasonable to make the assumption that – in a VLKDB – the only truth values which are meaningful from the (average) user point of view are True, False and Unknown.

A pragmatic possible solution to deal with the potential underlying fancy truth values – without explicit references – is to involve the user in the process which evaluates the acceptability of the submitted instances. This is the basic idea in Borgida's proposal, which requires the user confirmation to accept an exception. It would be interesting to develop user interfaces which handle a variety of truth values via a variety of levels of involvement of users in the evaluation process. As an example, it could go from a simple warning (when the instance is almost "normal") to a complex multiple confirmation (involving several users at various authorization levels).

> *In VLKDBs, the number and diversity of users are such that it is practically impossible to make explicit references to fancy truth values. As a pragmatic substitute, the potential imperfection of instances – submitted or derived – can be taken into account in requiring the involvement of user(s) in the evaluation of their acceptability, the "strength" of such involvement depending on the exceptionality of the instances.*

4.3 A specific research area

The state of the art and these basic guidelines show that the problem of non-monotonic knowledge evolution, in the context of VLKDBs, is extremely rich and complex.

– formal problems : as investigated in AI, the formal problem of non-monotonicity reveals other formal problems, which are probably more fundamental. The numerous works which have been done on the subject suffer from their relative isolation. A global formal investigation (taking into account both symbolic and numeric approaches) would provide an interesting theoritical basis for future research.

– ergonomic problems : no automatic handling means that humans play a major role in co-operating with the system to evaluate the acceptability of non-conforming instances. An ergonomic research is necessary to define how the user interface can appropriately "model" the surprise attached with any piece of information which is not quite "normal". In VLKDBs, this is probably a key issue because the population of users is highly heterogeneous.

– technical problems : from the system point of view, several major problems appear, such as the integration of exceptional instances in the KDB, the management of "ordinary" manipulations confronted with exceptions and the control of exceptional manipulations. These problems need to be addressed by the research community in order to provide a sound and general theoritical basis for future engineering work.

– economical problems : the classical 90/10 ratio can be used to infer that the cost of the management of exceptions (in a broad sense of the word) is equivalent to the cost of the management of normal instances. It would probably be very interesting to investigate this issue in various DP environments to know more precisely what is the cost of non-monotonic knowledge evolution in existing systems. If such investigation shows that the economical "weight" of non-monotonicity is quite important, it could provide a new perspective in the database community for original research and developments.

5 An overview of semantic tolerance

The objective of this section is to provide an illustration of a research work undertaken in the specific context of VLKDBs. It is only an overview, a formal proposal being given in [ESCU 89], and more detailed presentations in [ESCU 87] and [ESCU 88a].

In what follows, the emphasis is essentially on the links which can be established between semantic tolerance and various aspects of KDB management, in order to show the importance of the problem of non-monotonic knowledge evolution.

5.1 The basic principle of semantic tolerance

In all engineering activities – except in software engineering – specifications take into account the imperfection of the engineering process.

As an example, in mechanics, the specification of a diameter is given by a formula such as :

$$D = 86mm \pm .2mm$$

– "$86mm$" is the **nominal** value of the diameter. It is the ideal dimension, which provides the best functionality for the considered part. But it can be considered as unreachable, because there is no way to manufacture a part with exactly this diameter.

– "$\pm .2mm$" is the **tolerance** associated with the nominal value, which defines what actual dimensions should be considered as acceptable, around the ideal unreachable dimension.

This concept of tolerance is essential in engineering. The given example comes from mechanics, but comparable examples could come from chemistry, electricity, optics, etc. Tolerance is essential for the simple reason that it acknowledges the actual complexity of reality and it takes into account the fact that humans can only get partial control over it.

Surprisingly, in software engineering, rule definition is done as if humans had the absolute control of the reality which is modelled. Specifications are considered as perfect and all possible situations are supposed to be encompassed. As an example, an integrity constraint is something absolute : there is no provision for unexpected situations which do not satisfy the constraint and which, nevertheless, are not errors.

The basic principle of semantic tolerance is to transpose the classical concept of tolerance in the KDB environment, incorporating in rule specifications the definition of a "fringe" in order to take into account the complexity and the potential imperfection of the modelling process (considered, in this context, as a transposition of the manufacturing process).

5.2 Tolerant rule definition

Given a rule R_σ as formulated in a KDBS, the transposition of the engineering concept of tolerance leads to the following tolerant version of the rule :

$$R^*_\sigma = (R_\sigma \pm (\varepsilon_\sigma, \delta_\sigma))$$

where :

– R_σ is the **nominal rule**;

– ε_σ is the **tolerance predicate**, which is essentially used to differentiate exceptions from errors;

– δ_σ is the **compensating operation** [CERI 88], undertaken by the system when an exception occurs, in order to incorporate the exception into the KDB and to preserve the consistency of the corresponding theory.

– $(\varepsilon_\sigma, \delta_\sigma)$ constitute the **tolerance** associated with the nominal rule.

Remark : the sign "$\pm$" is used in the formulation of tolerant rules as a symbol to recall the underlying concept of tolerance as used in engineering.

5.3 Enforcement of a tolerant rule

When an instance is submitted to the system, if it satisfies the nominal rule R_σ it is considered as a **normal instance**, and consequently it is accepted by the system. If it does not satisfy the nominal rule, the system evaluates the tolerance predicate which is associated with the nominal rule in order to decide whether the non-conformance to the rule is actually due to the fact that the instance is erroneous, or if it is due to the imperfection of the nominal rule itself (in this case, the instance is an exception and must be accepted). This evaluation introduces a balance between rules administrators – who define rules – and end-users – who submit instances.

5.4 Examples of tolerance predicates

The tolerance predicate is the support for such balance. It can refer to specific users to take into account their potential reliability, but it can also refer to data values, to exception set cardinality and to temporal elements in order to provide a better control over non-monotonic knowledge evolution.

- Tolerance refering to end-users.

When a user can be considered as highly reliable with respect to a given nominal rule, the tolerance says that all non-conforming instances submitted by this user must be considered as exceptions (and never as errors). It leads to a tolerance predicate such as :

$$\varepsilon_\sigma = \text{X-USER}(u_1)$$

When the user reliability is a bit lower, the tolerance can be more limited, a confirmation being required to take into account the non-conforming instance. Example :

$$\varepsilon_\sigma = \text{X-CONFIRM}(u_2)$$

Such tolerance predicates are based on a limitation of the population authorized to submit exceptions and/or on a specific involvement of the users in the evaluation of the acceptability of the submitted non-conforming instances – as an example a simple confirmation, or a more complex series of confirmations involving various levels of responsibility. They rely on the same basic principle that the proposals made by Moore, Doyle or Cohen. The major difference is that the specific constraints due to VLKDBs require a drastic simplification in the way beliefs, justifications or endorsements are treated. The user confirmation is an important element in Borgida's proposal, where it is used to distinguish exceptions from errors. With respect to this aspect, semantic tolerance can be considered as an extension of Borgida's proposal (because tolerance refering to end-users is not limited to confirmation, and because other tolerances are proposed). A major difference, in another perspective, is that semantic tolerance preserves the stability of the rules.

- Tolerance refering to attribute values.

  To increase the control of the system, to make sure that "reasonable" instances only are accepted, the tolerance predicate can be defined in reference to attribute values. Example :

  $$\varepsilon_\sigma = \text{X-VALUE}(\text{ specie} = \text{kiwi })$$

  This kind of tolerance encompasses the classical approach where pre-defined exceptions are excluded from the normal processing (cf exception handlers in programming languages or McCarthy's proposal when the abnormality predicate is specifically valuated).

- Tolerance refering to the maximum number of exceptions.

  Because exceptions are difficult to handle in the everyday operation of any VLKDB, some limitation on the maximum number of exceptions tolerated in the KDB can be defined in the tolerance predicate. Example :

  $$\varepsilon_\sigma = \text{X-CARD}(0.01 \times \text{CARD bird-set})$$

  This tolerance says that no more than 1 % of the bird instances can be accepted by the system as exceptions with respect to the associated nominal rule. The motivation for such tolerance is essentially pragmatic, because in VLKDBs, some control must be kept on the overall "manageability" of the system.

- Tolerance refering to temporal elements.

  Phenomenons related with time are of major importance in VLKDBs ([ADIB 87]). In the context of non-monotonicity, temporality appears for exceptions which cannot be tolerated for an indefinitely long period of time. It corresponds to the concept of temporary exceptions, which is quite classical in every organization, but which is poorly supported by current systems (in particular with the strict for-ever-or-not-at-all approach linked to the classical concept of transaction). For example, the tolerance predicate which limits the existence of an exception in the database for at most a period of 2 months is :

  $$\varepsilon_\sigma = \text{X-LIFETIME}(\text{ 2 mo})$$

  In this framework, it is quite easy to address the problem of inconsistency within a transaction (what leads, in current systems, to defer integrity constraint checking). The tolerance predicate needs only to limit the acceptability of non-conforming instances until the commit is issued :

  $$\varepsilon_\sigma = \text{X-LIFETIME}(\triangleright \mathit{commit})$$

  Temporal tolerance provides an interesting insight on the problems related with transactions in general, and with long-term transactions in particular [BAKK 85], because the basic idea is to deal with instances which cannot be considered as fully acceptable, the "final" commit being delayed in some future, which is nevertheless too far to keep these instances "outside" the KDB.

- Composite tolerance.

  A tolerance predicate can involve several elementary tolerances in order to provide a better control over exceptions. Example :

  $\varepsilon_\sigma =$ X-CONFIRM(u_2) $\wedge$ X-LIFETIME(2 mo)

5.5 Integration and manipulation of exceptions

When a non-conforming instance is considered as "good" because the tolerance predicate is satisfied, the system must integrate this instance into the KDB and, furthermore, provide the facilities required to manipulate the KDB containing the various exceptions.

The integration work which must be done by the system – corresponding to δ_σ – is highly dependent on the kind of rule which is considered and on the kind of tolerance (in particular for temporary tolerances). It cannot be discussed here because it requires detailed investigations (cf [ESCU 89]).

5.6 Original KDB design techniques based on tolerance

The basic principle of semantic tolerance comes from a universal technique used to define specifications in taking into account the impossibility to manufacture "perfect" products. Quite logically, the transposition of tolerance in the KDB world leads to original design techniques. The basic idea is to consider the following correlative evolutions :

– on one hand the evolution of the precision of the rules (during the very first design phases, tolerance is very large, and progressively it is reduced until possibly strict rule definitions),

– on the other hand the evolution of the size and potential reliability of the population of users which can interact with the system (during the first phases, the population is very limited, restricted to highly reliable users, and progressively the KDB is opened to wider populations until the final user population).

This approach is quite original compared with current design techniques. It takes its foundations in the adjustement between rule quality and instance quality, which is made possible at various levels in expressing tolerance predicates for various populations of users. A detailed discussion of this approach can be found in [ESCU 90]. This work and Borgida's proposal for refining the database schema on the basis of encountered exceptions ([BOWI 85]) show the strong relationship between non-monotonic knowledge evolution and design techniques.

6 Conclusion

Non-monotonic knowledge evolutions and exceptions have been studied in various areas – usually kept unrelated. The diversity of the numerous proposals is of course very interesting, but it can be perceived as a weakness, revealing that more fundamental underlying problems have not yet been recognized and formally investigated. For example, non-monotonicity is only the consequence – in a given context – of the imperfection of a rule, challenged by an instance of better quality. Consequently, the state of the art cannot be considered as providing a sound theoritical framework. An important formal research effort is necessary to address the problem and to isolate fundamental issues which would provide a uniform consistent theoritical basis.

In VLKDBs, the problem must be considered in integrating the associated specific constraints.

For example, the issue of the underlying multiple truth values becomes very complex when considering the size and heterogeneity of the end-users population interacting with the system.

Another example is provided by the problem of the stability of the rule base, which is not a major issue in "small" and/or classical databases, but which becomes very important in VLKDBs because the size and complexity of the rule base require as much stability as possible to keep the whole KDB manageable.

Consequently, in VLKDBs, the problem is not only a fundamental theoritical problem. It raises also major technical, practical and ergonomic problems.

Semantic tolerance is briefly described in the paper in order to provides a general framework to discuss the problem. Such framework seems particularly interesting because it addresses non-monotonic knowledge evolution in the specific context of VLKDBs, and because it takes advantage of a universal engineering technique, which is surprisingly not used in data processing.

But the major goal of the paper is not to propose a specific approach. Its goal is to show that non-monotonic knowledge evolution in VLKDBs is a key issue. Semantic tolerance provides a good illustration of this fact because it is strongly related with various research domains, such as :

– user interfaces (to express adequately the quality of the submitted instances),

– rule administration (to take into account the potential imperfection of some rules, and the conse-

quences of such imperfection in the every day life of the database),

– advanced design techniques (where end-users can actually participate to the design process from its very beginning) and

– long-term transactions (because a temporary exception is a typical situation which requires to revisit the classical transaction concept)

These relationships show that the problem is a central problem, of a major theoritical, technical and economical importance.

Almost all the database research is based on the assumption that the universe of discourse and its modelling satisfy the property of regularity. This assumption was needed in the past to symplify the overall problem of data modelling, but nowadays it is clearly unacceptable to investigate advanced database techniques (in particular VLKDBs).

This paper is an attempt to emphasize the need for a new area of research within the database paradigm, to address what is called here "non-monotonic knowledge evolution".

Acknowledgements

The production of this paper would not have been possible without the contribution of Gio Wiederhold (Stanford University) and of Michel Adiba and the LGI database team (Grenoble University). The author would like to thank them cordially.

References

[ALHE 83] G. ALEFELD, J. HERZBERGER *Introduction to Interval Computations*, Academic Press, 1983.

[ANHA 87] T. ANDREWS, C. HARRIS *Combining Language and Database Advances in an Object-Oriented Development Environment*, OOPSLA Int. Conf., 1987.

[BAKK 85] F. BANCILHON, W. KIM, H.F. KORTH *A Model of CAD Transactions*, Proc. VLDB Conference, Stockholm 1985.

[BHKA 86] R.K. BHATNAGAR, L.N. KANAL *Handling Uncertain Information : A Review of Numeric and Non-numeric Approaches*, in ref. [KALE 86].

[BOMW 86] A. BORGIDA, T. MITCHELL, K. E. WILLIAMSON *Learning Improved Integrity Constraints and Schemas from Exceptions in Data and Knowleddge Bases*, in On Knowledge Base Management Systems, BRODIE and MYLOPOULOS Eds, Springer-Verlag, 1986.

[BORG 85] A. BORGIDA *Language Features for Flexible Handling of Exceptions in Information Systems*, ACM TODS, vol 10, No 4, Dec 1985.

[BORG 87] A. BORGIDA *A Syntax and Semantics for Class Hierarchies with Contradictions*, Research report, 1987.

[BOWI 85] A. BORGIDA, K. E. WILLIAMSON *Accomodating Exceptions in Databases, and Refining the Schema by Learning from Them*, in Proc. VLDB, Stockholm, 1985.

[CERI 88] S. CERI, F. GARZOTTO *Specification and Management of Database Integrity Constraints through Logic Programming*, Research Report, 1988.

[CHEE 86] P. CHEESEMAN *Probabilistic versus Fuzzy Reasoning* in [KALE 86].

[COHE 85] P.R. COHEN *Heuristic Reasoning about Uncertainty : an Artificial Intelligence Approach*, Pitman Publ., 1985.

[COMA 84] G. COPELAND, D. MAIER *Making SMALLTALK a Database System*, ACM-SIGMOD Int. Conf., 1984.

[COX- 79] R.T. COX *Of Inference and Inquiry - An Essay in Inductive Logic*, in The Maximum Entropy Formalism, LEVINE and TRIBUS Eds, M.I.T. Press, 1979.

[CRIS 79] F. CRISTIAN *Le traitement des exceptions dans les programmes modulaires*, Thèse de doctorat, Université de Grenoble, 1979.

[CRIS 82] F. CRISTIAN *Exception Handling and Software Fault Tolerance*, IEEE Trans. on Computers, vol C-31, No 6, June 1982.

[deKL 86] J. de KLEER *An Assumption Based TMS*, Artificial Intelligence, 28, 1986 (in [GINS 87]).

[DOYL 79] J. DOYLE *A Truth Maintenance System*, Artificial Intelligence, 12, 1979 (in [GINS 87]).

[DUPR 88] D. DUBOIS, H. PRADE *Possibility Theory*, Plenum Press, New York, 1988.

[ESCU 87] C. ESCULIER *La Tolérance Sémantique dans les Bases de Données*, Proc. 3èmes Journées BD Avancées, Port-Camargue, 1987.

[ESCU 88a] C. ESCULIER *Inheritances with Exceptions : An Approach based on Semantic Tolerance*, Proc. IFIP 2.6/8.1 Conference on The role of Artificial Intelligence in Databases and Information Systems, Guangzhou, Pop. Rep. of China,1988.

[ESCU 88b] C. ESCULIER *Non-monotonicity and Uncertainty in Semantic Tolerant KBMSs*, Stanford University Research Report, Nov 1988.

[ESCU 89] C. ESCULIER *Introduction à la tolérance sémantique*, thèse de doctorat, Grenoble I University, 1989.

[ESCU 90] C. ESCULIER *Towards Advanced I.S. Design Based on Semantic Tolerance*, IFIP Int. Conf. on Human Factors in I.S. Analysis and Design, Shaerding, Austria, 1990.

[GENI 87] M.R. GENESERETH, N.J. NILSSON *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann Publ., 1987.

[GINS 86] M.L. GINSBERG *Multi-valued Logics*, in Proc. Fifth National Conference on Artificial Intelligence, 1986 (in [GINS 87]).

[GINS 87] M.L. GINSBERG (Editor) *Readings in Non-monotonic Reasoning*, Morgan Kaufmann Publ., 1987.

[GOOD 75] J.B. GOODENOUGH *Exception Handling : Issues and a Proposed Notation*, ACM Communication, vol 18, No 12, Dec 1975.

[ISRA 80] D.J. ISRAEL *What's wrong with Non-monotonic Logic*, Proc. First Annual Conference on Artificial Intelligence, Stanford, 1980 (in [GINS 87]).

[KALE 86] L.N. KANAL, J.F. LEMMER (Editors) *Uncertainty in Artificial Intelligence*, North-Holland, 1986 (reprinted 1988).

[KERS 86] M. KERSTEN, F. SCHIPPERS *Using the Guardian Programming Paradigm to Support Database Evolution*, Proc. IFIP TC2 DS-2 Conference, Albufeira, 1986.

[KONO 87] K. KONOLIGE *On the Relation between Default Theories and Autoepistemic Logic*, in [GINS 87].

[LERI 89] C. LECLUSE, P. RICHARD *The O2 Database Programming Language*, in Proc. VLDB, Amsterdam, 1989.

[LIFS 85] V. LIFSCHITZ *Computing Circumscription*, in Proc. Ninth International Joint Conference on Artificial Intelligence, 1985 (in [GINS 87]).

[LISK 79] B.H. LISKOV, A. SNYDER *Exception Handling in CLU*, IEEE Trans. Software Eng., vol SE-5, 1979.

[LUCK 80] D.C. LUCKMAN, W. POLAK *ADA Exception Handling : An Axiomatic Approach*, ACM Trans. Prog. Lang. Syst., vol 2, No 2, 1980.

[McCA 80] J. McCARTHY *Circumscription - a Form of Non-monotonic Reasoning*, Artificial Intelligence, 13, 1980 (and in [GINS 87]).

[McCA 86] J. McCARTHY *Applications of Circumscription to Formalizing Common-sense Knowledge*, Artificial Intelligence, 28, 1986 (in [GINS 87]).

[McDD 80] D. McDERMOTT, J. DOYLE *Non-monotonic Logic I*, Artificial Intelligence, 13, 1980 (in [GINS 87]).

[MOOR 84] R.C. MOORE *Possible Worlds Semantics for Autoepistemic Logic*, in Proc. 1984 Non-monotonic Reasoning Workshop, AAAI, New Paltz, 1984 (in [GINS 87]).

[MOOR 85] R.C. MOORE *Semantical Considerations on Non-monotonic Logic*, Arrtificial Intelligence, 25, 1985 (in [GINS 87]).

[PERL 86] D. PERLIS *On the Consistency of Commonsense Reasoning*, Computational Intelligence, 2, 1986.

[PIWI 88] L.J. PINSON, R.S. WIENER *An Introduction to Object-Oriented Programming and SMALLTALK*, Addison-Wesley Pub. Company, 1988.

[PLET 88] U. PLETAT *Thruth Maintenance for Logic Programs*, Research Report, Stanford, 1988.

[RAMI 88] R. G. RAMIREZ, R.DATTERO, J. CHOOBINEH *Extension of Relational Views to Derived Relations with Exceptions*, subm. paper, 1988.

[REIT 80] R. REITER *A Logic for Default Reasoning*, Artificial Intelligence, 13, 1980 (in [GINS 87]).

[REIT 88] R. REITER *On Integrity Constraints*, in Proc. TARK II, Asilomar, CA, 1988.

[RINE 84] D. RINE (Editor) *Computer Science and Multiple-valued Logic*, North-Holland, 1984.

[SHAF 76] G. SHAFTER *A Mathematical Theory of Evidence*, Princeton University Press, 1976.

[SMDP 88] P. SMETS, E.H. MAMDANI, D. DUBOIS, H. PRADE (Editors) *Non-Standard Logics for Automated Reasoning*, Academic Press, 1988.

[SMET 82] P. SMETS *Elementary Semantic Operators*, in [YAGE 82].

[STAM 86] R. STAMPER *The Processing of Business Semantics : Necessity and Tools*, Proc. IFIP TC2 DS-2 Conference, Albufeira, 1986.

[WIED 88] G. WIEDERHOLD, C.R. ROLLINGER, J. GEMANDER *Uncertainty and Methods for Dealing with Uncertainty*, IBM Research Report, 1988.

[YAGE 82] R. YAGER (Editor) *Fuzzy Sets and Possibility Theory*, Pergamon, 1982.

[ZADE 78] L.A. ZADEH *Fuzzy Sets as a basis for a Theory of Possibility*, in Fuzzy Sets and Systems, vol 1, No 1, 1978.

[ZADE 80] L.A. ZADEH *Inference in Fuzzy Logic*, Proc. 10th Annual Symposium on Multi-valued Logic, 1980.