11

LOGICAL HANDLING OF INCONSISTENT AND DEFAULT INFORMATION

Philippe Besnard
Luis Fariñas del Cerro*
Dov Gabbay**
Anthony Hunter**

IRISA
Rennes, France

** IRIT*
Université Paul Sabatier
Toulouse, France

*** Department of Computing*
Imperial College
London, UK

1 INTRODUCTION

The subjects of this chapter are two important and related kinds of uncertainty in information systems: inconsistent information and default (defeasible) information. In many information system applications, there is a need to represent and reason with inconsistent data. For example, in a tax collection agency, database records on individual taxpayers should be allowed to have inconsistent information, as such information could be used to direct enquiries by tax inspectors. Default information , such as rules that are usually true but are allowed to have exceptions, tends to reduce the size of databases significantly, yet without significant loss of utility for many applications. For example, a market research agency could use default information in its consumer profiles: for its kind of business such a level of accuracy could be deemed sufficient.

Once we allow uncertainty of either kind in an information system, we must also incorporate a reasoning component that would conclude answers from this more general information. Such a component must be based on an appropriate formal model of deduction. It would be natural to consider using classical logic for this purpose. Unfortunately, classical logic is unsatisfactory because it allows arbitrary conclusions to be drawn from inconsistent information. Similarly, default information cannot be handled adequately in classical logic because there is a dynamic need to change conclusions whenever new information is added to the system. These shortcomings force us to consider nonclassical logics for our model of reasoning. These alternative logics are referred to as *logics for practical reasoning*.

A number of logical systems have been developed for these forms of reasoning. In the following two sections we discuss some of the issues behind inconsistent information and default information, and then we present paraconsistent logic and default logic as important candidates for reasoning with these respective forms of information. Then, in Section 4, we present labeled deductive systems as a general framework for capturing these logics and tailoring them for individual applications. We conclude in Section 5 with a brief summary.

Note that in this chapter we do not review the literature on handling inconsistent information in relational databases. For this, see Chapters 3 and 4 in this book. Other references include [3, 4, 9, 17].

2 HANDLING INCONSISTENT INFORMATION

There are many situations in which information and its contrary both appear in an information system. In some situations such inconsistencies could be useful, such as in a collection database, where they could initiate profitable enquiries. In other situations they are undesirable, such as in a bank database of customer accounts, where they need to be identified and corrected (through a revision of the database).

In some situations it is not even clear that inconsistencies should be corrected. For example, the tax agency database may include an item of legislation that prohibits citizens from having more than one spouse. Now, suppose, quite unexpectedly from the point of view of the database designer, that this database includes a taxpayer who has two spouses. Although, this creates an inconsistency, revising the database might not be the most appropriate solution.

Using $\forall$ to represent "for all," $\wedge$ to represent conjunction, and $\rightarrow$ to represent implication, we can represent the previous example with these formulae

$$\begin{array}{l}\forall x, y\ \mathit{Spouse}(x,y) \wedge \mathit{Spouse}(x,z) \rightarrow y = z\\ \mathit{Spouse}(\mathit{MrBigamist}, \mathit{MsVictim})\\ \mathit{Spouse}(\mathit{MrBigamist}, \mathit{MsMisled})\\ \mathit{MsVictim} \neq \mathit{MsMisled}\end{array}$$

Using classical logic, it would be possible to infer any arbitrary conclusion from this information. This is because the classical logic incorporates the following proof rule, called *ex falso quodlibet*,

$$\frac{\alpha \quad \neg\alpha}{\beta}.$$

This proof rule states that from the two items α and $\neg\alpha$, any conclusion β may be inferred. Applying *ex falso quodlibet* to *MrBigamist*'s case (substituting $\mathit{MsVictim} = \mathit{MsMisled}$ for α), we could draw irrelevant and inappropriate conclusions, such as

$$\mathit{Rain\text{-}falls}(\mathit{mainly\text{-}on\text{-}the\text{-}plain}).$$

What is required for reasoning with databases in which inconsistencies are allowed to occur is a logic that does not incorporate the rule of *ex falso quodlibet.* One such class of logics is the paraconsistent logics (for a review of paraconsistent logics see [2]). These logics use the same language as classical logic, but they use only a subset of the proof rules. This implies that for any given database we may infer fewer conclusions. Reasoning that is supported by paraconsistent logics includes *modus ponens*,

$$\frac{\alpha \quad \alpha \rightarrow \beta}{\beta}.$$

So, for example, from

$$\begin{array}{l}\forall x, y\ \mathit{Spouse}(x,y) \rightarrow \mathit{Spouse}(y,x)\\ \mathit{Spouse}(\mathit{MrMartin}, \mathit{MrsMartin})\end{array}$$

we may infer

$$\mathit{Spouse}(\mathit{MrsMartin}, \mathit{MrMartin}).$$

Other reasoning that is supported by paraconsistent logics includes the rule of disjunctive introduction,

$$\frac{\alpha \rightarrow \gamma \quad \beta \rightarrow \gamma}{\alpha \vee \beta \rightarrow \gamma}.$$

In many ways, a paraconsistent logic is a useful substitute for classical logic. But it does lack some intuitive proof rules, such as *modus tollens*,

$$\frac{\alpha \rightarrow \beta \quad \neg\beta}{\neg\alpha}.$$

So, for example, from

$$\forall x, y \; \mathit{Spouse}(x, y) \rightarrow \mathit{Spouse}(y, x)$$
$$\neg \mathit{Spouse}(\mathit{MrMartin}, \mathit{MrsJones})$$

the conclusion $\neg\mathit{Spouse}(\mathit{MrsJones}, \mathit{MrMartin})$ cannot be inferred, although it is clearly a desirable inference.

Returning to the original information in the *MrBigamist* example, a paraconsistent logic allows us to infer useful conclusions from the data. Furthermore, it is very robust in the sense that regardless of the information that is introduced into the database, the reasoning process will always give sensible conclusions.

Another advantage of a paraconsistent logic is that it does not force any decision to be made on whether a particular item of information in the database is "false." Thus, we are not forced to decide which of $\mathit{Spouse}(\mathit{MrBigamist}, \mathit{MsVictim})$ and $\mathit{Spouse}(\mathit{MrBigamist}, \mathit{MsMisled})$ is false. Similarly, we do not have to decide which of $\mathit{MsMisled} \neq \mathit{MsVictim}$ or $\mathit{MsMisled} = \mathit{MsVictim}$ holds.

In general, a paraconsistent logic can be used to give guidance on the source of the inconsistency, and indicate actions that should be taken on the database. For example, paraconsistent logics can be used as a formal basis for truth maintenance systems, which are meant to partition the database into consistent subsets of data (for example, [18]).

3 HANDLING DEFAULT INFORMATION

It is noteworthy that practical reasoning relies much more on exploiting *general* rules (i.e., rules that are not necessarily universal) than on a myriad of individual facts. General rules tend to be less than perfectly accurate and may therefore have exceptions. Nevertheless, in modeling practical reasoning it is intuitive to resort to general rules and therefore allow the inference of useful conclusions, even if it does entail making some mistakes, because not all exceptions to these rules are necessarily known. Clearly, it is more efficient to state (and deal with) a single general proposition than to state (and deal with) possibly thousands of instances of such a general proposition.

An example of an application in which general rules would be beneficial is marketing, where decisions are usually based on generalities about customers rather than on perfectly accurate information about each and every customer.

Consider the following example. In general, a person who is a customer of a telephone company has a telephone instrument. Of course, exceptions exist: deaf people, for instance, have special instruments that are not "telephones" *stricto sensu*. So, if Fernandez is a customer of the telephone company, then it makes sense to conclude that he has a telephone. The statement "a customer of a telephone company has a telephone, unless proven otherwise" is *default information*. The principle by which it is to occur in reasoning is "if x is a customer of a telephone company, then x has a telephone, unless it is proven that x counts as an exception."

Such reasoning may be represented with this notation:

$$\frac{PhoneCustomer(x)\ :\ \neg Exception(x)}{HasPhone(x)}.$$

This rule is applied as follows. Given a certain value v for x, if

$$PhoneCustomer(v)$$

is inferred and

$$Exception(v)$$

cannot be proven, then

$$HasPhone(v)$$

is concluded (and is called a *default conclusion*).

As a special case, if

$$PhoneCustomer(Fernandez)$$

is inferred and

$$Exception(Fernandez)$$

cannot be proven, then

$$HasPhone(Fernandez)$$

is concluded.

Note the flexibility of this use of default information. From the fact that Fernandez is a customer of a telephone company, and in the absence of any evidence that Fernandez counts as an exception, the general rule leads to the conclusion that Fernandez has a telephone. Importantly, there is no need *to prove* that Fernandez is not an exception (for instance, that he is not deaf). It is sufficient to establish that no proof is available according to which Fernandez may be classified as an exception; clearly, this is much less demanding.

Furthermore, there is no need to have a list of all exceptions. The default information

$$\frac{PhoneCustomer(x)\ :\ \neg Exception(x)}{HasPhone(x)}$$

need not be modified as information about exceptions evolves. For example, suppose that "deaf people have no telephones." Then, adding the formula

$$\forall x\ Deaf(x) \rightarrow Exception(x)$$

is enough to block the default conclusion $HasPhone(x)$ for x that correspond to deaf persons (while still permitting the default conclusion $HasPhone(x)$ for x that do not correspond to deaf persons). As an illustration, consider

$$\begin{array}{l} PhoneCustomer(Baker) \\ PhoneCustomer(Cook) \\ Deaf(Cook) \end{array}$$

Then,

$$\frac{PhoneCustomer(x)\ :\ \neg Exception(x)}{HasPhone(x)}$$

can be applied to $x = Baker$ because $PhoneCustomer(Baker)$ can be established and $Exception(Baker)$ cannot be proven, so the default conclusion

$$HasPhone(Baker)$$

is inferred. Consider now the case $x = Cook$, and try to apply the default information,

$$\frac{PhoneCustomer(x)\ :\ \neg Exception(x)}{HasPhone(x)},$$

Clearly $PhoneCustomer(Cook)$ can be inferred. However, the general rule cannot be applied because $Exception(Cook)$ can be proven (via $Deaf(Cook)$ and $\forall x\ Deaf(x) \rightarrow Exception(x)$).

Conveniently, all new exceptions discovered with time can be taken into account by simply adding them to the knowledge base; there is no need to modify the general rule. The general rule will simply cease to yield some previous conclusions (the ones corresponding to the newly introduced exceptions). Consider this example:

$$\begin{array}{l} \mathit{PhoneCustomer}(\mathit{Fernandez}) \\ \mathit{Deaf}(\mathit{Fernandez}) \\ \forall x\ \mathit{Deaf}(x) \rightarrow \mathit{Exception}(x) \end{array}$$

When only $\mathit{PhoneCustomer}(\mathit{Fernandez})$ was known, $\mathit{HasPhone}(\mathit{Fernandez})$ was inferred. If, in addition, $\mathit{Deaf}(\mathit{Fernandez})$ and $\forall x\ \mathit{Deaf}(x) \rightarrow \mathit{Exception}(x)$ are known, then $\mathit{HasPhone}(\mathit{Fernandez})$ is no longer inferred. Such behavior is termed *nonmonotonic reasoning* because a conclusion drawn in the presence of certain information is withdrawn upon the introduction of additional information. That is, the set of conclusions does not increase monotonically as information increases.

An interesting observation is that some exceptions just *fail to obey* the general rule, while others *explicitly oppose* it. For instance, for deaf we might want to assert both

$$\forall x\ \mathit{Deaf}(x) \rightarrow \mathit{Exception}(x)$$

and

$$\forall x\ \mathit{Deaf}(x) \rightarrow \neg \mathit{HasPhone}(x)$$

whereas for hearing-impaired people, we might want to stay agnostic about whether or not they have telephones, and assert only

$$\forall x\ \mathit{HearingImpaired}(x) \rightarrow \mathit{Exception}(x)$$

Also, there is no need to know the reason why an item is an exception to the general rule. Indeed,

$$\begin{array}{l} \mathit{PhoneCustomer}(\mathit{Baker}) \\ \mathit{Exception}(\mathit{Baker}) \end{array}$$

is sufficient to block the default conclusion, without providing a reason why Baker is a telephone company customer without a telephone.

Observe that priorities among general rules can be rendered. Consider an example of a computer club whose members use telephone lines equipped with modems to transmit data. They must count as exceptions to our general rule, unless they have more than a single telephone line:

$$\frac{\mathit{MemberComputerClub}(x)\ :\ \mathit{SingleLine}(x)}{\mathit{Exception}(x)}.$$

Consider an individual *Smith*, such that

$$\begin{array}{l} \mathit{PhoneCustomer}(\mathit{Smith}) \\ \mathit{MemberComputerClub}(\mathit{Smith}) \end{array}$$

Because $\neg SingleLine(Smith)$ cannot be proven,

$$Exception(Smith)$$

is inferred by this new rule, and the earlier rule is blocked; i.e., $HasPhone(Smith)$ is not inferred.

If one tries to apply the earlier rule

$$\frac{PhoneCustomer(x)\ :\ \neg Exception(x)}{HasPhone(x)}$$

first, then $HasPhone(Smith)$ is inferred on condition that $\neg Exception(Smith)$ could not be inferred. Applying, the new rule now, yields $Exception(Smith)$, violating the proviso imposed on the earlier application, and thus voiding it.

To summarize, default logic [19] aims at formalizing reasoning from default information by means of formulas of classical logic and the so-called *default rules*, namely the expressions

$$\frac{\alpha\ :\ \beta}{\gamma}$$

where α, β, and γ are formulas of classical logic. The inference rules are those of classical logic plus a special mechanism to deal with default rules: basically, if α is inferred, and $\neg\beta$ cannot be inferred, then infer γ. The above examples demonstrated the main ideas; more complete treatment may be found in [1], for instance.

4 LABELED DEDUCTIVE SYSTEMS FOR PRACTICAL REASONING

Developing logics for practical reasoning creates new demands on the apparatus for defining the language and the proof theory. We approach this problem as follows: we augment the language by labeling its formulae, and we define the proof theory to manipulate both the formulae and the labels on formulae. This approach has driven the development of a general framework, called Labeled Deductive Systems (LDS), for presenting logics that handle labeled formulae [12, 14].

The basic unit of information in LDS is a labeled formula $i:\ \alpha$, where i is a label, and α is an unlabeled formula. A logic can then be defined in terms of allowed operations on the labeled formulae. For example, logical consequence can be defined on labeled formulae,

$$\frac{\{i_1 : \alpha_1, \cdots, i_n : \alpha_n\}}{j : \beta}$$

where $i_1, \ldots, i_n$ are labels, j is a function of $i_1, \ldots, i_n$, and $\alpha_1, \ldots, \alpha_n$, β are formulae.

Different applications of LDS are made possible by different definitions for the logical manipulation of formulae α and for the algebraic manipulation of the labels i. Furthermore, many existing logics fit into the LDS framework, including temporal logics [10], modal and many-valued logics [8], resource logics [15], and nonmonotonic logics [13, 11, 16].

4.1 LDS for Default Logic

We begin by showing how default logic can be handled by LDS. We assume the usual set of logical formulae, which we denote F, and we label each item in F with the symbol 0. We assume that default rules have the form,

$$i : \frac{\alpha : \beta}{\gamma},$$

where $\alpha, \beta, \gamma \in F$, and $i \in 2^{\mathcal{N}}$ is a unique label (i.e., i is a set of integers), and we let D denote the set of default rules. Instead of using quantified default rules, we use their instantiated form, as common in default logic. However, we also introduce labeling of the default rules. As before, we regard α as a precondition, β as a justification, and γ as the consequent. A database Δ is a subset of $D \cup F$.

An example of a database is the default rules

$$\{1\} : \frac{PhoneCustomer(Baker) : \neg Exception(Baker)}{HasPhone(Baker)}$$

$$\{2\} : \frac{PhoneCustomer(Cook) : \neg Exception(Cook)}{HasPhone(Cook)}$$

and the data

$$\{0\} : \quad PhoneCustomer(Baker)$$
$$\{0\} : \quad PhoneCustomer(Cook)$$

In the following we provide the definition of an *extension*, which is a consistent set of conclusions of a database.

The notion of an extension in the LDS framework, is captured with two mechanisms. The first mechanism concerns the derivation of formulae using the proof rules of classical logic and the default rules. When we apply a proof rule or a default rule, we keep track of the data and default rules used by propagating the labels. For instance, from $i:\ A$ and $j:\ B$ we obtain $i \cup j:\ A \wedge B$.

The second mechanism concerns the propagated labels. They indicate which default rules have been applied to obtain a specific formula, and hence each label delineates a subset of the data. Basically, we have to test that the default rules applied to infer a formula with a given label do not have their justification contradicted by the subset of data corresponding to that label.

So, for the above example, by the first mechanism we get the following labeled formulae,

$$\begin{array}{ll} \{0\}: & PhoneCustomer(Cook) \\ \{0\}: & PhoneCustomer(Baker) \\ \{2,0\}: & HasPhone(Cook) \\ \{1,0\}: & HasPhone(Baker) \end{array}$$

and by the second mechanism, we have the following extension,

$$\begin{array}{l} PhoneCustomer(Cook) \\ PhoneCustomer(Baker) \\ HasPhone(Cook) \\ HasPhone(Baker) \end{array}$$

4.2 LDS for Paraconsistent Logic

LDS can also be used for presenting paraconsistent logics. For this we consider the system C_ω of da Costa [7] that is formalized by the following proof method of Carnielli *et al* [5, 6]. First, da Costa introduces the notion of a well-behaved formula: $\neg(\alpha \wedge \neg\alpha)$ is not valid in general, but if it does hold for a formula α, it is a well-behaved formula, and is denoted α°. Second, each formula α is labeled with either a $+$ symbol or a $-$ symbol, and we call $+:\alpha$ and $-:\alpha$ *signed formulae*. Intuitively, $+:\alpha$ and $-:\alpha$ can be interpreted as α being true and α being false, respectively. Any set of sets of signed formulae is called a *form*.

Let α and β be two formulae. Below are a set of production rules that can be used to transform a set of formulae into either a new set of formulae, or set of sets of formulae.

$$
\begin{array}{l}
\{\delta, + : (\alpha \wedge \beta)\} \Rightarrow \{\delta, + : \alpha, + : \beta\} \\
\{\delta, - : (\alpha \vee \beta)\} \Rightarrow \{\delta, - : \alpha, - : \beta\} \\
\{\delta, - : (\alpha \rightarrow \beta)\} \Rightarrow \{\delta, + : \alpha, - : \beta\} \\
\{\delta, + : (\neg\neg\alpha)\} \Rightarrow \{\delta, + : \alpha\} \\
\{\delta, - : (\neg\alpha)\} \Rightarrow \{\delta, + : \alpha\} \\
\{\delta, - : (\neg\neg\alpha)\} \Rightarrow \{\delta, - : \alpha\} \\
\{\delta, - : (\alpha \diamond \beta)^\circ\} \Rightarrow \{\delta, - : (\alpha^\circ \diamond \beta^\circ)\}, \text{where } \diamond \in \{\wedge, \vee, \rightarrow\} \\
\{\delta, - : (\alpha \wedge \beta)\} \Rightarrow \{\{\delta, - : \alpha\}, \{\delta, - : \beta\}\} \\
\{\delta, + : (\alpha \vee \beta)\} \Rightarrow \{\{\delta, + : \alpha\}, \{\delta, + : \beta\}\} \\
\{\delta, + : (\alpha \rightarrow \beta)\} \Rightarrow \{\{\delta, - : \alpha\}, \{\delta, + : \beta\}\} \\
\{\delta, + : (\neg\alpha)\} \Rightarrow \{\{\delta, - : \alpha\}, \{\delta, - : \alpha^\circ\}\}
\end{array}
$$

Given a form C, we denote by $R(C)$ the result of applying one of the rules to the form. A tableau is a sequence of forms $C_1, \ldots, C_n$, such that $C_{i+1} = R(C_i)$. To test if a formulae can be inferred from a set of formulae, we label it with the $-$ symbol, add it to the data, and construct a tableau. The formula can be inferred if the tableau is closed. A tableau is closed if every set of formulae of its form is closed, and a set of formulae is closed if there is a formula α for which $+ : \alpha$ and $- : \alpha$ belong to that set.

For example, consider the following market research data on voting. In this example, there is a symmetry about whether or not Dick is a Pacifist. In other words, there is an argument that Dick is a Pacifist, and an argument that Dick is not a Pacifist.

$$
\begin{array}{l}
\mathit{Dick} \rightarrow (\mathit{Republican} \wedge \mathit{Quaker}) \\
\mathit{Quaker} \rightarrow \mathit{Pacifist} \\
\mathit{Republican} \rightarrow \neg \mathit{Pacifist} \\
\mathit{Dick}
\end{array}
$$

Running the tableaux rules for this set, the resulting open tableau is the proposed solution to the problem introduced by the inconsistency. Here we consider only the two main forms, one of which is closed and the other is not. The rest of the closed forms will be omitted.

$$
\begin{aligned}
C_0 &= \{+:(Dick \to (Republican \wedge Quaker)),\\
&\quad +:(Quaker \to Pacifist),\\
&\quad +:(Republican \to Pacifist), +:(Dick)\}\\
C_1 &= C_0 \cup \{+:(Republican \wedge Quaker)\}\\
C_2 &= C_1 \cup \{+:(Republican) + :(Quaker)\}\\
C_3 &= C_2 \cup \{+:Pacifist, +:(\neg Pacifist)\}\\
C_4 &= \{\{C_3 \cup \{-:Pacifist\}\}, \{C_3 \cup \{-:Pacifist^\circ\}\}\}
\end{aligned}
$$

The set $\{C_3 \cup \{-:Pacifist\}$ is closed, and the set $\{C_3 \cup \{-:Pacifist^\circ\}\}$ is not closed. This means that we can restrict our considerations to the following set of signed elementary expressions of the open set $-:Pacifist^\circ$, $+:Pacifist$, $+:Quaker$, $+:Republican$, $+:Dick$. This set gives us a solution to the problem in the sense that we consider *Dick* as a *Quaker*, *Pacifist*, and *Republican*, but his *Pacifism* is controversial. This also shows how even though the database is inconsistent, the technique allows us to identify *Pacifist*/$\neg$*Pacifist* as being central to this inconsistency problem.

The computational complexity of the deduction method presented is similar to classical logic. This is in contrast to the usual nonmonotonic logics, where complexity is extremely high. This is due to the fact that paraconsistent logics block certain deductions from inconsistencies, whereas many nonmonotonic logics, such as default logic, use consistency checking to ensure that each extension is free from inconsistencies.

Nevertheless, it is perhaps now evident that using default or defeasible data, and using inconsistent data are two interrelated problems. A significant part of reasoning with default rules is resolving inconsistencies. Similarly, many problems of inconsistencies in information arise from the use of default information.

Note that the above example captures an equivalence regarding the *Pacifist*/$\neg$*Pacifist* nature of *Dick*. On other words, there is no apparent way of determining a priority on the information. This contrasts with many examples in which some kind of priority can be identified to resolve the problem. We return to this issue later.

4.3 Skeptical and Credulous Views

In reasoning with both inconsistent information and default information, there is the question of whether to adopt a skeptical or credulous view. In a skeptical view, the logic is cautious and does not allow conflicting inferences, whereas in a credulous view, the logic is less cautious, and does allow conflicting inferences. The rationale behind a credulous view is that the user makes a selection from the conflicting inferences. For example, take the following defaults rules,

$$\{3\} : \frac{Aircraft(x) \ : \ RequireRunway(x)}{RequireRunway(x)}$$

$$\{4\} : \frac{Helicopter(x) \ : \ \neg RequireRunway(x)}{\neg RequireRunway(x)}$$

and the following facts

$$\begin{array}{ll} \{0\} : & Aircraft(Sikorsky) \\ \{0\} : & Helicopter(Sikorsky) \end{array}$$

From this, there are two extensions. The first contains $RequireRunway(Sikorsky)$ and the second contains $\neg RequireRunway(Sikorsky)$. A credulous view would allow both as possible inferences, whereas a skeptical view would allow neither. Similarly, in paraconsistent logics, we may have the following data,

$$\begin{array}{ll} \{5\} : & Aircraft(x) \rightarrow RequireRunway(x) \\ \{6\} : & Helicopter(x) \rightarrow \neg RequireRunway(x) \\ \{0\} : & Aircraft(Sikorsky) \\ \{0\} : & Helicopter(Sikorsky) \end{array}$$

In the same way, this paraconsistent logic gives both $RequireRunway(Sikorsky)$ and $\neg RequireRunway(Sikorsky)$ as inferences. As with default reasoning, a credulous view would allow both as acceptable inferences, whereas a skeptical view would allow neither. However, it is not clear in general whether reasoning should be skeptical or credulous.

4.4 Resolving Conflicts

One solution to these kinds of problems is to use the labels to resolve the conflict. Essentially, the labels can be used to capture extra information about the formulae in the database, and about the inferences, so that a judicious choice can be made. For a variety of applications, LDS meets the need for extra information about data. This may be further object-level information or metalevel information or semantic information. A label can represent a wide variety of notions. Take the labeled formula $k : \ \alpha$. The label k could capture any of the following:

- the fuzzy reliability of α
- the origin, or source, of α
- the priority of α
- the time when α holds
- a proof of α in, for example, a truth maintenance system

So, for example, with the database about aircraft, we could introduce an ordering over formulae that captures a notion of specificity. For the above database, the default,

$$\{4\} : \frac{\mathit{Helicopter}(x)\ :\ \neg \mathit{RequireRunway}(x)}{\neg \mathit{RequireRunway}(x)}$$

is more specific than the default,

$$\{3\} : \frac{\mathit{Aircraft}(x)\ :\ \mathit{RequireRunway}(x)}{\mathit{RequireRunway}(x)}$$

since helicopters are a subclass of aircraft. The ordering can then be used to allow the inference $\neg \mathit{RequireRunway}(\mathit{Sikorsky})$ in preference to its complement.

Note that using such a selection technique with paraconsistent logic then means the system can change inferences in the light of new information. Hence the behavior is very similar to that of default logic. For example, for the following data,

$$\begin{array}{ll} \{5\} : & \mathit{Aircraft}(x) \rightarrow \mathit{RequireRunway}(x) \\ \{6\} : & \mathit{Helicopter}(x) \rightarrow \neg \mathit{RequireRunway}(x) \\ \{0\} : & \mathit{Aircraft}(\mathit{Sikorsky}) \end{array}$$

The inference $\mathit{RequireRunway}(\mathit{Sikorsky})$ is selected. However, it is then retracted when the fact $\mathit{Helicopter}(\mathit{Sikorsky})$ is added to the this data.

As another example of how labels can be used to resolve conflict, consider a groupware system that collates office memos, and users query this system about company regulations. Suppose that information in memo $\{7\}$ includes the following statement,

$$\{7\} : \mathit{ExportCustomer}(x) \rightarrow \mathit{ChargeCustomerInDollars}(x)$$

and memo $\{8\}$ includes statements,

$$\begin{array}{ll} \{8\} : & \mathit{ExportCustomer}(x) \rightarrow \mathit{ChargeCustomerInDeutchmarks}(x) \\ \{8\} : & \neg \mathit{ChargeCustomerInDollars}(x) \vee \\ & \neg \mathit{ChargeCustomerInDeutchmarks}(x) \end{array}$$

If a user has the fact *ExportCustomer*(*Philips*), then the groupware data is inconsistent using classical logic. However, if the labels correspond to the data of the memo, then there is a preference for information from the more recent memo. In this example, if $\{7\}$ corresponds to 23 January 1992, and $\{8\}$ corresponds to 25 February 1993, then the inconsistency can be resolved. As before, if new information is added to the system, such as new memos, then inferences might have to be retracted.

This feature of retracting inferences in the light of new data is termed nonmonotonicity. It is not a desirable concept *per se*. It means a lack of monotonicity, and hence the lack of a property that classical logic and some of its close relatives have. The term nonmonotonic logic was used because the logics being developed for reasoning with default information seemed to have nonmonotonicity as their prime characteristic.

Nonmonotonicity is required when only a partial knowledge of a situation is possible. Rarely does a system have at its disposal all the information that would be desirable. However, to wait until all required information has been assimilated would involve delay, even infinite delay. Obviously this is not satisfactory. To ameliorate, some nonmonotonic reasoning mechanism must be resorted to. In other words, we argue that some form of plausible reasoning is required. Where everything pertinent to the investigation is known, monotonicity is more appropriate.

5 CONCLUSIONS

The ubiquitous usage by organizations of information that incorporates defaults and inconsistencies contrasts sharply with the low level of computer-based handling of such information. The situation will change as the expanding role of information technology means that handling of such information will become increasingly significant. Indeed, as defaults and inconsistencies pervade virtually any real-world scenario, techniques for handling them must be incorporated into any information system that attempts to provide a substantive model of the real world.

In this chapter, we have illustrated the argument to handle default and inconsistent information with examples of situations where such information could be potentially useful. There are a variety of techniques proposed to handle such information. However, it seems that only via formal techniques, such as logics of practical reasoning, can we hope to provide a viable framework for incorporating default and inconsistent information into information systems. This includes developing logical proof systems and means for harnessing extra information about formulae, such as semantic or metalevel information. In this way, we can identify and resolve the conflicts that arise when using default or inconsistent information.

Acknowledgments

The authors wish to thank Philippe Smets, Curtis Dyerson, and Ami Motro for helpful feedback.

REFERENCES

[1] P. Besnard. *Introduction to Default Logic.* Springer-Verlag, 1989.

[2] P. Besnard. Paraconsistent logics approach to knowledge representation. In *Proceedings of World Conference on the Fundamentals of Artificial Intelligence*, pages 107–114. Angkor, 1991.

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

[4] A. Borgida and K. Williamson. Accommodating exceptions in databases and refining the schema by learning from them. In *Proceedings of the Eleventh International Conference on Very Large Data Bases* (Stockholm, Sweden, August 21–23), pages 72–81, 1985.

[5] W. A. Carnielli, L. Fariñas del Cerro, and M. Lima-Marques. Contextual negation and reasoning with contradiction. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (*Sydney, Australia, August*)*, pages 532–537, 1991.

[6] W. A. Carnielli and M. Lima-Marques. Reasoning under inconsistent knowledge. *Journal of Applied Non-Classical Logics*, 2: 49–79, 1992.

[7] N. da Costa. On the theory of inconsistent information. *Notre Dame Journal of Formal Logic*, 15: 497–510, 1974.

[8] M. D'Agostino and D. Gabbay. Labelled refutation systems. In *Proceedings of the Workshop on Theorem Proving with Analytic Tableaux and Related Methods* (Marseilles, France, April), pages 243–281, 1993.

[9] H. Dreizen and S. Chang. Imprecise schema: A rationale for relations with embedded subrelations. *ACM Transactions on Database Systems*, 14(4): 447–479, December 1989.

[10] M. Finger and D. Gabbay. Labelled database management system. In *Proceedings of the International Conference on Database Theory*, volume 646 of *Lecture Notes in Computer Science*, pages 188–200. Springer-Verlag, 1992.

[11] D. Gabbay. Abduction in labelled deductive systems: A conceptual abstract. In *Symbolic and Quantitative Approaches to Uncertainty*, volume 548 of *Lecture Notes in Computer Science*, pages 3–12. Springer-Verlag, 1991.

[12] D. Gabbay. Labelled deductive systems. Technical report, Centrum fur Informations und Sprachverabeitung, Universitat Munchen, 1991.

[13] D. Gabbay. Theoretical foundations for non-monotonic reasoning, part 2: Structured non-monotonic theories. In *Proceedings of Scandinavian Conference on Artificial Intelligence*, pages 19–40. IOS Press, 1991.

[14] D. Gabbay. Labelled deductive systems: A position paper. In *Logic Colloquium 90*, volume 2 of *Lecture Notes in Logic*, pages 66–88. Springer-Verlag, 1993.

[15] D. Gabbay and R. de Queiroz. Extending the Curry-Howard interpretation to linear, relevant and other resource logics. *Journal of Symbolic Logic*, 57: 1319–1366, 1992.

[16] A. Hunter. A conceptualization of preferences in non-monotonic proof theories. In *Logics in AI*, volume 633 of *Lecture Notes in Computer Science*, pages 174–188. Springer-Verlag, 1992.

[17] P. King and C. Small. Default databases and incomplete information. *Computer Journal*, 34: 239–244, 1991.

[18] J. Martins and S. Shapiro. A model of belief revision. *Artificial Intelligence*, 35: 25–79, 1988.

[19] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13: 187–214, 1980.