Reasoning with inconsistency in structured text

Anthony Hunter
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
a.hunter@cs.ucl.ac.uk

December 1, 1999

Abstract

Reasoning with inconsistency involves some compromise on classical logic. There is a range of proposals for logics (called paraconsistent logics) for reasoning with inconsistency each with pros and cons. Selecting an appropriate paraconsistent logic for an application depends on the requirements of the application. Here we review paraconsistent logics for the potentially significant application area of technology for structured text. Structured text is a general concept that is implicit in a variety of approaches to handling information. Syntactically, an item of structured text is a number of grammatically simple phrases together with a semantic label for each phrase. Items of structured text may be nested within larger items of structured text. The semantic labels in a structured text are meant to parameterize a stereotypical situation, and so a particular item of structured text is an instance of that stereotypical situation. Much information is potentially available as structured text including tagged text in XML, text in relational and object-oriented databases, and the output from information extraction systems in the form of instantiated templates. In this review paper, we formalize the concept of structured text, and then focus on how we can identify inconsistency in items of structured text, and reason with these inconsistencies. Then we review key approaches to paraconsistent reasoning, and discuss the application of them to reasoning with inconsistency in structured text.

1 Introduction

Inconsistency is a ubiquitous phenomenon in the real-world, and so as computers are used in ever more sophisticated roles, it is an increasingly important issue in computer science. Inconsistency is not necessary a bad thing. Rather an inconsistency can be useful. For example, it can be an alert to a potential problem, or offer the basis for conflict resolution, or indicate an interesting anomalous situation. For a discussion on the value of inconsistency see [GH91, GH93].

The best approach to inconsistency is not necessarily immediate eradication. Rather, we need to analyse each inconsistency and then act on it in a context-sensitive way. In this paper, we will consider some of the issues of handling inconsistency, and in particular look at ways that we can reason in the presence of inconsistency, analyse inconsistency, and act on inconsistency. In order to compare proposals for reasoning with inconsistency, it is worthwhile considering the needs of applications. Different logics are appropriate for different applications. Here we focus on the potentially significant application area of technology for structured text.

Syntactically, an item of structured text is a data structure containing a number of grammatically simple phrases together with a semantic label for each phrase. The set of semantic labels in a structured text is meant to parameterize a stereotypical situation, and so a particular item of structured text is an instance of that stereotypical situation. Using appropriate semantic labels, we can regard a structured text as an abstraction of an item of text.

For example, news reports on corporate acquisitions can be represented as items of structured text using semantic labels including `buyer`, `seller`, `acquisition`, `value`, and `date`. Each semantic label provides semantic information, and so an item of structured text is intended to have some semantic coherence. Each phrase in structured text is very simple — such as a proper noun, a date, or a number with unit of measure, or a word or phrase from a prescribed lexicon. For an application, the prescribed lexicon delineates the types of states, actions, and attributes, that could be conveyed by the items of structured text.

Much material is potentially available as structured text. This includes items of text structured using XML tags (see for example [GQ99, Pfa99]), the output from information extraction systems given in templates (see for example [CL96, Gri97, ARP98]), and databases used by some online news agencies where journalists file reports as structured text and these entries are used by editors to generate free text news reports in different languages. The notion of structured text also overlaps with semi-structured data (for reviews see [Abi97, Bun97]).

Whilst structured text is useful as a resource, there is a need to develop techniques to handle, analyse, and reason with it. We may want to check consistency of structured text with what we already know, and we may want to draw inferences from structured text even if there are inconsistencies between items of structured text and the domain knowledge. To support this, we also require a domain knowledgebase, which may include integrity constraints, default rules (rules that normally hold but can have exceptions), and various database resources.

We have assumed the words and phrases are sufficiently simple and restricted to not require natural language processing. In most cases, we will represent each word or phrase by a constant symbol in the logic. For some phrases, we may represent it as a function symbol with constant symbols as arguments (Eg. a date could be a function symbol with arguments for the day, month, and year).

In this paper, we provide basic definitions for structured text, and for the notion of a stereotype. We then show how structured text can be translated into classical logic to allow reasoning with domain knowledge. We then review some key approaches to reasoning with inconsistent information in knowledgebased systems, and show how we can use them for reasoning with inconsistency in structured text.

2 Formalizing structured text

In this section we will formalize the notions of structured text and of stereotype.

2.1 Structured text

Here we adopt some basic definitions that should be easy to view as an adaptation of ideas in a variety of fields in XML, relational and object-oriented databases, language engineering, and knowledgebased systems.

Definition 2.1 *A* **word** *is a string of alphanumeric characters, and a* **phrase** *is a string of one or more words. A* **text entry** *is either a phrase or a null value. A* **semantic label** *is a phrase.*

Example 2.1 *Examples of words include* `John`, `France`, `drive`, `happy`, `23`, *and* `3i`, *and exampleples of phrases include* `University of London`, `John`, `23 April 1999`, *and* `warm and sunny`.

Assumption 2.1 *In this paper, we will assume the set of semantic labels and the set of text entries are disjoint.*

Definition 2.2 *If ϕ is a semantic label, and ψ is a text entry, then $\langle \phi : \psi \rangle$ is an atomic feature.*

Example 2.2 *Examples of atomic features include:*

$$\langle \texttt{name} : \texttt{John} \rangle$$

$$\langle \texttt{name: University of London} \rangle$$

$$\langle \texttt{today's weather: sunny and windy} \rangle$$

Definition 2.3 **Complex features** *are defined as follows: (1) if $\langle \phi : \psi \rangle$ is an atomic feature, then $\langle \phi : \psi \rangle$ is a complex feature; and (2) if ϕ is a semantic label and $\sigma_1, ..., \sigma_n$ are complex features, then $\langle \phi : \sigma_1, ..., \sigma_n : \phi \rangle$ is a complex feature.* **An item of structured text** *is just a complex feature.*

Assumption 2.2 *In this paper, we assume that the order of the items $\sigma_1, ..., \sigma_n$ in a complex feature $\langle \phi : \sigma_1, ..., \sigma_n : \phi \rangle$ is not important. So for example, the following two are assumed to be equivalent:*

$$\langle \texttt{customer} : \langle \texttt{name} : \texttt{John Smith} \rangle, \langle \texttt{fax number: 0171 111 2222} \rangle : \texttt{customer} \rangle$$

$$\langle \texttt{customer} : \langle \texttt{fax number: 0171 111 2222} \rangle, \langle \texttt{name} : \texttt{John Smith} \rangle : \texttt{customer} \rangle$$

Example 2.3 *An example of a complex feature is:*

```
⟨weather report:
          ⟨date: 23 April 1999⟩,
          ⟨city: London⟩,
          ⟨today's weather: cold and wet⟩,
          ⟨tomorrow's weather: sunny and windy and cool⟩
:weather report⟩
```

2.2 Stereotypes

Essentially, a set of semantic labels, called a stereotype, is used to delineate a recurring situation. A structured report is then an item of structured text where the semantic labels are those of the stereotype[1].

Definition 2.4 *If ϕ is a semantic label, and Ψ is a set of text entries, then $[\phi : \Psi]$ is an atomic stereotype.*

Definition 2.5 *An atomic feature $\langle \phi : \psi \rangle$ is an instance of an atomic stereotype $[\phi : \Psi]$ iff $\psi \in \Psi$.*

Definition 2.6 *Each atomic stereotype is a complex stereotype, and if ϕ is a semantic label, and $\Gamma_1, ..., \Gamma_n$ are complex stereotypes, and $\Delta, ..., \Delta$ are complex stereotypes, then the following is a complex stereotype*

$$[\phi : \Gamma_1, ..., \Gamma_n :: \Delta_1, ..., \Delta_m : \phi]$$

The complex stereotypes on the left of the :: symbol are **compulsory stereotypes** *and the complex stereotypes on the right are* **optional stereotypes**.

We will view a complex feature as being an instance of a complex stereotype if and only if the complex feature is composed of an instance of each of the compulsory stereotypes and possibly an instance of one or more of the optional stereotypes. This is captured in the following definition.

Definition 2.7 *A complex feature $\langle \phi : \sigma_1, .., \sigma_k : \phi \rangle$ is an instance of a complex stereotype*

$$[\phi : \Gamma_1, ..., \Gamma_n :: \Delta_1, ..., \Delta_m : \phi]$$

iff σ_1 is an instance of one of $\Gamma_1, ..., \Gamma_n, \Delta_1, ..., \Delta_m$ and .... and σ_k is an instance of one of $\Gamma_1, ..., \Gamma_n, \Delta_1, ..., \Delta_m$ and for each $\Gamma_i \in \{\Gamma_1, ..., \Gamma_n\}$ there is a $\sigma_j \in \{\sigma_1, .., \sigma_k\}$ such that σ_j is an instance of Γ_i.

Example 2.4 *An example of a stereotype called* `weather report` *is:*

```
⟨weather report:
     ⟨date: X1⟩,
     ⟨city: X2⟩,
     ⟨today's weather: X3⟩,
     ::
     ⟨tomorrow's weather: X3⟩
:weather report⟩
```

where `X1` *is the set of all dates, and* `X2` *is the set all cities, and* `X3` *is the following set.*

{`sunny and cold, wet and cold, sunny and warm, wet and warm and windy, ......`}

Note, the notion of a complex stereotype does not incorporate an explicit form of disjunction. So for example for the following two complex stereotypes, where `X1` is a set of firstnames and `X2` is a set of surnames,

[1] This subsection could be easily skipped on a first reading of this paper.

$$\langle \texttt{name:}\langle \texttt{firstname: X1}\rangle\texttt{:name}\rangle$$

$$\langle \texttt{name:}\langle \texttt{surname: X2}\rangle\texttt{:name}\rangle$$

we cannot collapse them into one stereotype where there is a disjunction of the stereotypes ⟨`firstname: X1`⟩ and ⟨`surname: X2`⟩.

Definition 2.8 *If a complex feature θ is an instance of a complex stereotype λ, then θ is* **a structured report***, and the stereotype of the structured report is λ. If λ is*

$$[\phi : \Gamma_1, ..., \Gamma_n :: \Delta_1, ..., \Delta_m : \phi]$$

then the name of the stereotype is ϕ. Note, more than one stereotype can have the same name.

A stereotype is a generalization of a number of concepts in computing including relational database schema, templates in information extraction, and frames in knowledgebased systems.

There is also a similarity between stereotypes and document type declarations (DTDs) in XML. However, DTDs incorporate many implementation features that are not appropriate here such as default values, a form of structural inheritance, and a framework for the surprisingly involved question of handling escape characters.

3 Logical reasoning with structured text

To represent items of structured text, we adopt classical first-order logic.

Assumption 3.1 *For logical reasoning we assume the usual language of classical first-order logic using the usual symbols $\forall$ and $\exists$ for quantification and the usual symbols $\wedge, \vee, \rightarrow$ and $\neg$ for logical connectives.*

Assumption 3.2 *Domain knowledge is some set of classical first-order formulae.*

An item of structured text is isomorphic to a tree. A tree formed from an item of structured text is called a structured text tree. Using the following definition helps us to consider translations of items of structured text into logical formulae by using a structured text tree as an intermediate.

Definition 3.1 *To form a structured text tree from an item of structured text, apply the following two rules exhaustively:*

Complex feature rule *For a complex feature of the form $\langle \phi : \sigma_1, .., \sigma_n : \phi \rangle$, let ϕ be the root of the tree, and let each of $\sigma_1, ..., \sigma_n$ be used to give a subtree. For each σ_i, if σ_i is a complex feature, apply the complex feature rule by recursion, and if σ_i is an atomic feature, then apply the atomic feature rule.*

Atomic feature rule *For an atomic feature of the form $\langle \phi : \psi \rangle$, let ϕ be the root, and let ψ be the child of ϕ.*

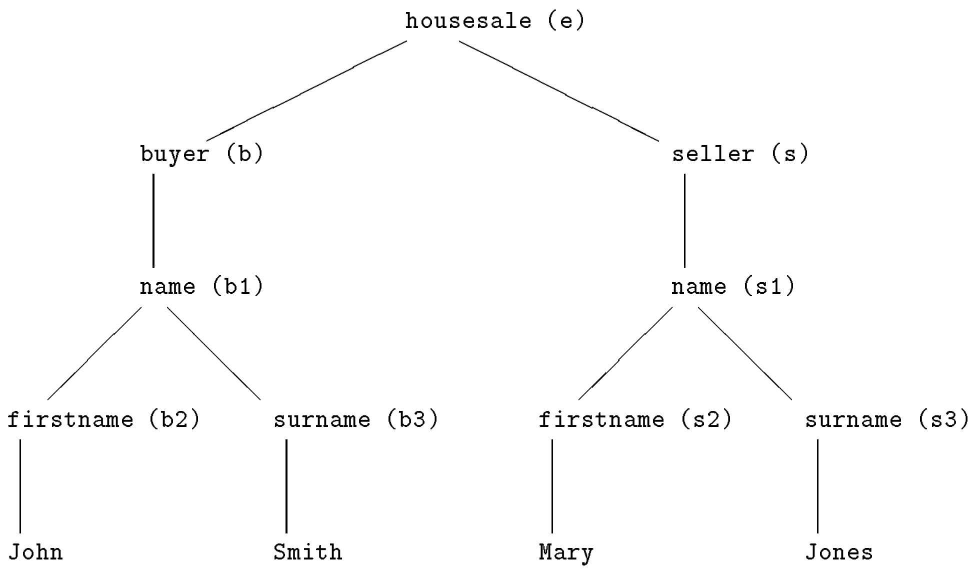

Figure 1: The structured text tree generated for Example 3.1. The unique reference label is given in parentheses.

For each node in the structured text tree that is a semantic label, we assume that the node also has a unique reference label.

So each node in the structured text tree is labelled with either a semantic label or a text entry. If a semantic label has come from an atomic feature, then we call it an atomic semantic label, otherwise we call it a complex semantic label. From this definition, we see that for an item of structured text the following hold: (1) all non-leaf nodes are semantic labels; (2) all leaf nodes are text entries; and (3) all penultimate nodes are atomic semantic labels.

Furthermore, if the item of structured text is a structured report, then we can see the structured text tree excluding the leaves gives the structure of the corresponding stereotype, and the name of the stereotype of the item of structured text is the root of the tree.

Example 3.1 *To illustrate generating a structured text tree, consider the following item of structured text, giving the tree in Figure 1.*

$$
\begin{array}{l}
\langle \texttt{housesale} : \\
\quad \langle \texttt{buyer} : \langle \texttt{name} : \langle \texttt{firstname} : \texttt{John} \rangle, \langle \texttt{surname} : \texttt{Smith} \rangle : \texttt{name} \rangle : \texttt{buyer} \rangle, \\
\quad \langle \texttt{seller} : \langle \texttt{name} : \langle \texttt{firstname} : \texttt{Mary} \rangle, \langle \texttt{surname} : \texttt{Jones} \rangle : \texttt{name} \rangle : \texttt{seller} \rangle \\
: \texttt{housesale} \rangle
\end{array}
$$

Now we consider how we can obtain a classical logic formula from one of these structured text trees.

Definition 3.2 *X is an offspring node of Y iff X is a child of Y or there is a Z such that Z is a child of Y and X is an offspring node of Z.*

Definition 3.3 *The set of formulae that can be formed from a structured text tree is obtained by exhaustively applying the following rules: (1) If X is a complex semantic label in the tree, and Y is the unique reference label for the parent of X, and Z is the unique reference label for X, then $X(Y, Z)$ is a formula; (2) If X is an atomic semantic label in the tree, and Y is the unique reference label for the parent of X, and Z is the text entry for the child of X, then $X(Y, Z)$ is a formula; (3) If X is the root of the tree and X is not an atomic semantic label, and Y is the unique reference label for X, then $X(Y)$ is a formula; and (4) If X is the root of the tree and X is an atomic semantic label, and Y is the unique reference label for X, and Z is the text entry, then $X(Y, Z)$ is a formula.*

In Definition 3.3, rule (1) gives the unique identifier for the item of structured text, rule (2) gives the path from the root node to any other non-leaf node, rule (3) gives text entry for each branch, and rule (4) gives the text entry when the tree represents an atomic feature.

Example 3.2 *Continuing Example 3.1, we obtain formulae including the following using Definition 3.3:*

$$\begin{gathered}\mathtt{housesale(e)}, \mathtt{buyer(e,b)}, \mathtt{seller(e,s)}, \\ \mathtt{name(b,b1)}, \mathtt{firstname(b1,John)}, \mathtt{surname(b1,Smith)}, \\ \mathtt{name(s,s1)}, \mathtt{firstname(s1,Mary)}, \mathtt{surname(s1,Jones)}\end{gathered}$$

In the following section, we review key paraconsistent logics, and then in Section 5, we will use paraconsistent logics for reasoning with the logical representation of items of structured text.

4 Paraconsistent logics for knowledgebased systems

Approaches to inconsistent information include database revision and paraconsistent logics. The first approach effectively removes data from the database to produce a new consistent database. In contrast, the second approach leaves the database inconsistent, but prohibits the logics from deriving the trivial inferences that follow from *ex falso quodlibet*. Unfortunately, the first approach means we may lose useful information — we may be forced to make a premature selection of our new database, or we may not even be able to make a selection. We focus here on the advantages and disadvantages of key proposals for a paraconsistent approach.

In the following subsections, we consider a range of paraconsistent logics that give sensible inferences from inconsistent information. We consider (1) Weakly-negative logics which use the full classical language, but a subset of the classical proof theory; (2) Four-valued logic which uses a subset of the classical language and a subset of the classical proof theory, together with an intuitive four-valued semantics; (3) Argumentative logics which reason with consistent subsets of classical formulae; and (4) Quasi-classical logic which uses classical proof theory but restricts the notion of a natural deduction proof by prohibiting the application of elimination proof rules after the application of introduction proof rules.

These options behave in quite different ways with data. None can be regarded as perfect for handling inconsistent information in general. Rather, they provide a spectrum of approaches. However, in all the approaches we cover, we aim to stay as close to classical reasoning as possible, since classical logic has many appealing features for knowledge representation and reasoning. We therefore do not aim to cover approaches based on defeasible logics or modal logics.

4.1 Weakly-negative logics

To avoid trivialization, weakly-negative logics compromise on classical proof theory. They allow, for example, normal notions of conjunction, such as $\alpha \wedge \beta$ gives α, but they are substantially weaker in terms of negation. There are a number of ways in which this can be achieved. One way is to drop some of the axiom schema of classical logic so that ex falso quodlibet and reductio ad absurdum do not hold. One proposal is a paraconsistent logic called C_ω logic proposed by da Costa [dC74]. Below we give a presentation of C_ω. All the schema in the logic C_ω are schema in classical logic.

Definition 4.1 *The logic C_ω is defined by the following axiom schema together with the modus ponens proof rule.*

$$\begin{gathered}
\alpha \rightarrow (\beta \rightarrow \alpha) \\
(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow (\alpha \rightarrow \gamma)) \\
\alpha \wedge \beta \rightarrow \alpha \\
\alpha \wedge \beta \rightarrow \beta \\
\alpha \rightarrow (\beta \rightarrow \alpha \wedge \beta) \\
\alpha \rightarrow \alpha \vee \beta \\
\beta \rightarrow \alpha \vee \beta \\
(\alpha \rightarrow \gamma) \rightarrow ((\beta \rightarrow \gamma) \rightarrow (\alpha \vee \beta \rightarrow \gamma)) \\
\alpha \vee \neg\alpha \\
\neg\neg\alpha \rightarrow \alpha
\end{gathered}$$

This proof theory gives the C_ω consequence relation.

Example 4.1 *To illustrate the use of C_ω, consider the following set of formulae.*

$$\{\alpha \rightarrow (\beta \wedge \gamma),\ \gamma \rightarrow \delta,\ \beta \rightarrow \neg\delta,\ \alpha\}$$

In this example, there is a symmetry about whether or not α is a δ. In other words, there is an argument that α is a δ, and an argument that α is $\neg\delta$. Using the proof theory we can derive inferences including α, β and γ. We can also derive both δ and $\neg\delta$.

In C_ω, rules such as modus tollens and disjunctive syllogism fail.

$$\frac{\alpha \rightarrow \beta, \neg\beta}{\neg\alpha} \qquad \text{[Modus tollens]}$$

$$\frac{\alpha \vee \beta, \neg\beta}{\alpha} \qquad \text{[Disjunctive syllogism]}$$

Many useful equivalences fail also such as the following,

$$\neg\alpha \vee \beta \not\equiv \alpha \rightarrow \beta$$

$$\neg\neg\alpha \not\equiv \alpha$$

In this sense weakly-negative logics are sub-systems of classical logic. In particular compromising on negation means that many classical inference steps involving negation fail in weakly-negative logics. Furthermore, the removal of certain classical inference rules means that the propositional connectives in the language do not behave in a classical fashion. In the case of C_ω the classical "sense" of negation – and as a result also the interdefinability of the classical connectives – has been traded in exchange for non-trivialisation. One manifestation of this, as discussed by Besnard [Bes91], is the following.

Example 4.2 *In C_ω, disjunctive syllogism, $((\alpha \vee \beta) \wedge \neg\beta) \rightarrow \alpha$, does not hold, whereas modus ponens, $(\alpha \wedge (\alpha \rightarrow \beta)) \rightarrow \beta$, does hold. So, for example, α does not follow from the database: $\{(\alpha \vee \beta), \neg\beta\}$, whereas α does follow from the database: $\{(\neg\beta \rightarrow \alpha), \neg\beta\}$.*

The logic C_ω is only one of a number of interesting weakly-negative logics. Further proof rules can be added to C_ω to give a stronger, and yet still non-trivializable, logic. For example, PI^s logic by Batens [Bat80] and VI logic by Arruda [Arr77]. Other weakly-negative logics can be defined by alternative, but similar weakenings, such as for relevant logics by Anderson and Belnap [AB75].

Weakly-negative logics are useful for rule-based reasoning with information since the logic supports modus ponens. They can be used to give guidance on the nature of an inconsistency and facilitate actions that should be taken on the database. Furthermore, they can be used without recourse to consistency checks. Finally, paraconsistent logics can be used as a formal basis for truth maintenance [MS88].

4.2 Four-valued logic

The four-valued logic of Belnap [Bel77] provides an interesting alternative to the logics presented in the previous section in that it has an illuminating and intuitive semantic characterization to complement its proof theory.

Definition 4.2 *The language for four-valued logic is a subset of classical logic. Let $\mathcal{P}$ be the usual set of formulae of classical logic that is formed using the connectives $\neg, \wedge$ and $\vee$. Then the set of formulae of the language, denoted $\mathcal{Q}$, is $\mathcal{P} \cup \{\alpha \rightarrow \beta \mid \alpha, \beta \in \mathcal{P}\}$, and hence implication is not nestable.*

Definition 4.3 *A formula in the language can be one of "true", "false", "both" or "neither", which we denote by the symbols T, F, B, and N, respectively.*

Example 4.3 *For the database $\{\alpha, \neg\alpha, \beta\}$, an acceptable assignment of truth values is such that α is B, $\neg\alpha$ is B, β is T, and γ is N.*

Intuitively we can view this form of assignment in terms of an "Approximation" lattice (see Figure 2 (left)). As more "information" is obtained about a formula, the truth-value "increases". In other words, if we know nothing about a formula, it is N. Then as we gain some information, it becomes either T or F. Finally, if we gain too much information it becomes B.

Definition 4.4 *For the semantics, we assume a distributive lattice, the "Logical" lattice (see Figure 2 (right)). We also assume an involution operator $*$ satisfying the conditions (1) $\alpha = \alpha^{**}$, and (2) if $\alpha \leq \beta$ then $\beta^* \leq \alpha^*$, where $\leq$ is the ordering relation for the lattice.*

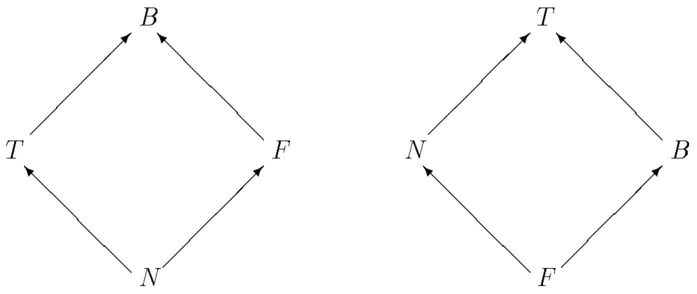

Figure 2: The "Approximation" lattice (left) and the "Logical" lattice (right)

Definition 4.5 *The semantic assignment function observes monotonicity and complementation, in the logical lattice, so $x \wedge y$ is the meet of $\{x, y\}$ and $x \vee y$ is the join of $\{x, y\}$, giving the following truth tables (Tables 1 – 3) for the $\neg, \wedge,$ and $\vee$ connectives. Let α, β be formulae. The inference β from α is valid iff $\beta \leq \alpha$, where $\leq$ is the ordering relation for the logical lattice. Let $\alpha \rightarrow \beta$ signify that the inference from α to β is valid in all four values, ie. that α entails β.*

There is no $\alpha \in \mathcal{Q}$ such that the semantic assignment function always assigns the value T. However, there are formulae that never take the value F, for example $\alpha \vee \neg\alpha$. Though the set of formulae that never take the value F is not closed under conjunction. For example, consider $(\alpha \vee \neg\alpha) \wedge (\beta \vee \neg\beta)$ when α is N and β is B.

To complement the semantics, the following is a definition for the proof theory for four-valued logic.

Definition 4.6 *Let $\alpha, \beta, \gamma \in \mathcal{L}$. The following are the proof rules for four-valued logic. Let $\alpha \leftrightarrow \beta$ signify that α and β are semantically equivalent, and can be intersubstituted in any context.*

$$\alpha_1 \wedge ... \wedge \alpha_m \rightarrow \beta_1 \vee ... \vee \beta_n \textit{ provided some } \alpha_i \textit{ is some } \beta_j$$
$$(\alpha \vee \beta) \rightarrow \gamma \textit{ iff } \alpha \rightarrow \gamma \textit{ and } \beta \rightarrow \gamma$$
$$\alpha \rightarrow (\beta \wedge \gamma) \textit{ iff } \alpha \rightarrow \beta \textit{ and } \alpha \rightarrow \gamma$$
$$\alpha \rightarrow \beta \textit{ iff } \neg\beta \rightarrow \neg\alpha$$
$$\alpha \rightarrow \beta \textit{ and } \beta \rightarrow \gamma \textit{ implies } \alpha \rightarrow \gamma$$
$$\alpha \rightarrow \beta \textit{ iff } \alpha \leftrightarrow (\alpha \wedge \beta) \textit{ iff } \beta \leftrightarrow (\alpha \vee \beta)$$

In addition, the following extends the definition of the proof theory.

$$\alpha \vee \beta \leftrightarrow \beta \vee \alpha$$
$$\alpha \wedge \beta \leftrightarrow \beta \wedge \alpha$$
$$\alpha \vee (\beta \vee \gamma) \leftrightarrow (\alpha \vee \beta) \vee \gamma$$
$$(\alpha \wedge \beta) \wedge \gamma \leftrightarrow \alpha \wedge (\beta \wedge \gamma)$$
$$\alpha \wedge (\beta \vee \gamma) \leftrightarrow (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$$
$$\alpha \vee (\beta \wedge \gamma) \leftrightarrow (\alpha \vee \beta) \wedge (\alpha \vee \gamma)\neg\neg\alpha \leftrightarrow \alpha$$
$$\neg(\alpha \wedge \beta) \leftrightarrow \neg\alpha \vee \neg\beta$$
$$\neg(\alpha \vee \beta) \leftrightarrow \neg\alpha \wedge \neg\beta$$
$$\alpha \leftrightarrow \beta \textit{ and } \beta \leftrightarrow \gamma \textit{ implies } \alpha \leftrightarrow \gamma$$

α	N	F	T	B
$\neg\alpha$	N	T	F	B

Table 1: Truth table for negation

$\wedge$	N	F	T	B
N	N	F	N	F
F	F	F	F	F
T	N	F	T	B
B	F	F	B	B

Table 2: Truth table for conjunction

$\vee$	N	F	T	B
N	N	N	T	T
F	N	F	T	B
T	T	T	T	T
B	T	B	T	B

Table 3: Truth table for disjunction

.

Example 4.4 *To illustrate the use of the proof theory consider the following example. As with the use of C_ω, there is an argument for δ and an argument for $\neg\delta$.*

$$\begin{gathered} \alpha \rightarrow (\beta \wedge \gamma) \\ \gamma \rightarrow \delta \\ \beta \rightarrow \neg\delta \\ \alpha \end{gathered}$$

From $\alpha \rightarrow (\beta \wedge \gamma)$, we get $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$. From $\alpha \rightarrow \beta$ and $\beta \rightarrow \neg\delta$, we get $\alpha \rightarrow \neg\delta$. From $\alpha \rightarrow \gamma$ and $\gamma \rightarrow \delta$, we get $\alpha \rightarrow \delta$. Hence, α is equivalent to $\alpha \wedge \delta$ and $\alpha \wedge \neg\delta$. However, the four-valued consequence relation deviates from the C_ω consequence relation in that we cannot detach δ from α nor $\neg\delta$ from α. This is in part due to proof theory incorporating neither modus ponens nor and elimination.

Four-valued logic provides a natural and intuitive form of paraconsistent reasoning. The semantic characterization based on the approximation lattice and logical lattice could be applicable for reasoning with facts. In particular, the logic seems useful for aggregating conflicting information. However, there are problems with reasoning with rules, particularly with respect to the lack of modus ponens and disjunctive syllogism. As with weakly-negative logics, the four-valued proof theory can be used without recourse to consistency checks.

There is a range of proposals for logics that are based on increasing the set of truth values beyond the Boolean values. A number of interesting proposals are based on three values with various meanings placed on the third value such as by Kleene and Bochvar (for a review see [Haa78]). More recently, some interesting variants of four valued logic have been developed (for a review see [VM98]).

4.3 Argument systems

One of the most obvious strategies for handling inconsistency in a database is to reason with consistent subsets of the database. This is closely related to the approach of removing information from the database that is causing an inconsistency. Here, we explore some of the issues relating these approaches in the context of classical proof theory.

Definition 4.7 *Let Δ be a database. Then:*

$$\begin{aligned}
&\mathsf{CON}(\Delta) = \{\Pi \subseteq \Delta | \Pi \not\vdash \bot\}\\
&\mathsf{INC}(\Delta) = \{\Pi \subseteq \Delta | \Pi \vdash \bot\}\\
&\mathsf{MC}(\Delta) = \{\Pi \in \mathsf{CON}(\Delta) | \forall \Phi \in \mathsf{CON}(\Delta) \Pi \not\subset \Phi\}\\
&\mathsf{MI}(\Delta) = \{\Pi \in \mathsf{INC}(\Delta) | \forall \Phi \in \mathsf{INC}(\Delta) \Phi \not\subset \Pi\}\\
&\mathsf{FREE}(\Delta) = \bigcap \mathsf{MC}(\Delta)
\end{aligned}$$

Hence $\mathsf{MC}(\Delta)$ is the set of maximally consistent subsets of Δ; $\mathsf{MI}(\Delta)$ is the set of minimally inconsistent subsets of Δ; and $\mathsf{FREE}(\Delta)$ is the set of information that all maximally consistent subsets of Δ have in common. We also have the following relationship.

$$\bigcap \mathsf{MC}(\Delta) = \Delta - \bigcup \mathsf{MI}(\Delta)$$

We can consider a maximally consistent subset of a database as capturing a "plausible" or "coherent" view on the database. For this reason, the set $\mathsf{MC}(\Delta)$ is important in many of the definitions presented in the rest of this section. Furthermore, we consider $\mathsf{FREE}(\Delta)$, which is equal to $\bigcap \mathsf{MC}(\Delta)$, as capturing all the "uncontroversial" information in Δ. In contrast, we consider the set $\bigcup \mathsf{MI}(\Delta)$ as capturing all the "problematical" data in Δ.

Example 4.5 *Let $\Delta = \{\alpha, \neg\alpha, \alpha \rightarrow \beta, \neg\alpha \rightarrow \beta, \gamma\}$. This gives two maximally consistent subsets, $\Phi_1 = \{\alpha, \alpha \rightarrow \beta, \neg\alpha \rightarrow \beta, \gamma\}$, and $\Phi_2 = \{\neg\alpha, \alpha \rightarrow \beta, \neg\alpha \rightarrow \beta, \gamma\}$. From this $\bigcap \mathsf{MC}(\Delta) = \{\alpha \rightarrow \beta, \neg\alpha \rightarrow \beta, \gamma\}$, and a minimally inconsistent subset $\Psi = \{\alpha, \neg\alpha\}$.*

A problem with using inferences from consistent subsets of an inconsistent database is that they are only weakly justified in general. To handle this problem, we can adopt the notion of an argument from a database, and a notion of acceptability of an argument. An argument is a subset of the database, together with an inference from that subset. Using the notion of acceptability, the set of all arguments can be partitioned into sets of (arguments of) different degrees of acceptability. This can then be used to define a class of consequence relations (see for example [BDP93, EGH95]).

Definition 4.8 *Let Δ be a database. An argument from Δ is a pair, (Π, ϕ), such that $\Pi \subseteq \Delta$ and $\Pi \vdash \phi$. An argument is consistent, if Π is consistent. We denote the set of arguments from Δ as $\mathsf{An}(\Delta)$, where $\mathsf{An}(\Delta) = \{(\Pi, \phi) | \Pi \subseteq \Delta \wedge \Pi \vdash \phi\}$. Γ is an argument set of Δ iff $\Gamma \subseteq \mathsf{An}(\Delta)$.*

Definition 4.9 *Let Δ be a database. Let (Π, ϕ) and (Θ, ψ) be any arguments constructed from Δ. If $\vdash \phi \leftrightarrow \neg\psi$, then (Π, ϕ) is a rebutting defeater of (Θ, ψ). If $\gamma \in \Theta$ and $\vdash \phi \leftrightarrow \neg\gamma$, then (Π, ϕ) is an undercutting defeater of (Θ, ψ).*

Rebutting defeat, as defined here, is a symmetrical relation. One way of changing this is by use of priorities, such as in systems based on explicit representation of preference (eg [Bre89, CRS93, BDP95]), or as in systems based on specificity (eg [Poo85]).

For a database Δ, an argumentative structure is any set of subsets of $\mathsf{An}(\Delta)$. The intention behind the definition for an argumentative structure is that different subsets of $\mathsf{An}(\Delta)$ have different degrees of acceptability. Below, we present one particular argumentative structure A^*, and then explain how the definition captures notions of acceptability.

Definition 4.10 *The following sets constitute the argumentative structure* A^**, where* Δ *is a database.*

$$
\begin{array}{lcl}
\mathsf{AT}(\Delta) & = & \{(\emptyset,\phi)|\emptyset\vdash\phi\} \\
\mathsf{AF}(\Delta) & = & \{(\Pi,\phi)|\Pi\subseteq\mathsf{FREE}(\Delta)\wedge\Pi\vdash\phi\} \\
\mathsf{AB}(\Delta) & = & \{(\Pi,\phi)|\Pi\in\mathsf{CON}(\Delta)\wedge\Pi\vdash\phi\wedge(\forall\Phi\in\mathsf{MC}(\Delta),\psi\in\Pi\Phi\vdash\psi)\} \\
\mathsf{ARU}(\Delta) & = & \{(\Pi,\phi)|\Pi\in\mathsf{CON}(\Delta)\wedge\Pi\vdash\phi\wedge \\
 & & \quad(\forall\Phi\in\mathsf{MC}(\Delta)\Phi\not\vdash\neg\phi)\wedge(\forall\Phi\in\mathsf{MC}(\Delta),\psi\in\Pi\Phi\not\vdash\neg\psi)\} \\
\mathsf{AU}(\Delta) & = & \{(\Pi,\phi)|\Pi\in\mathsf{CON}(\Delta)\wedge\Pi\vdash\phi\wedge(\forall\Phi\in\mathsf{MC}(\Delta),\psi\in\Pi\Phi\not\vdash\neg\psi)\} \\
\mathsf{A}\forall(\Delta) & = & \{(\Pi,\phi)|\Pi\in\mathsf{CON}(\Delta)\wedge\Pi\vdash\phi\wedge(\forall\Phi\in\mathsf{MC}(\Delta)\Phi\vdash\phi)\} \\
\mathsf{AR}(\Delta) & = & \{(\Pi,\phi)|\Pi\in\mathsf{CON}(\Delta)\wedge\Pi\vdash\phi\wedge(\forall\Phi\in\mathsf{MC}(\Delta)\Phi\not\vdash\neg\phi)\} \\
\mathsf{A}\exists(\Delta) & = & \{(\Pi,\phi)|\Pi\in\mathsf{CON}(\Delta)\wedge\Pi\vdash\phi\}
\end{array}
$$

The naming conventions for the argument sets are motivated as follows. T is for the tautological arguments - i.e. those that follow from the empty set of premises. F is for the free arguments - (due to Benferhat et al [BDP93]) - which are the arguments that follow from the data that is free of inconsistencies. B is for the backed arguments - i.e. those for which all the premises follow from all the maximally consistent subsets of the data. RU is for the arguments that are not subject to either rebutting or undercutting. U is for the arguments that are not subject to undercutting. $\forall$ is for the universal arguments - (essentially due to Manor and Rescher [MR70], where it was called inevitable arguments) - which are the arguments that follow from all maximally consistent subsets of the data. R is for the arguments that are not subject to rebutting. $\exists$ is for existential arguments - (essentially due to Manor and Rescher [MR70]) - which are the arguments with consistent premises.

The definitions for $\mathsf{A}\exists$, AF, AT should be clear. We therefore focus on the remainder. AR allows an argument (Π,ϕ) only if there is no maximally consistent subset that gives $\neg\phi$. AU allows an argument (Π,ϕ) only if for all items ψ in Π, there is no maximally consistent subset that gives $\neg\psi$. ARU combines the conditions of the AR and AU. Notice that AR and $\mathsf{A}\forall$ have very similar definitions, with the only difference being "$\Phi\not\vdash\neg\phi$" in AR versus "$\Phi\vdash\phi$" in $\mathsf{A}\forall$. A similar remark applies to AU and AB. Therefore $\mathsf{A}\forall$ and AB are strengthenings of AR and AU, respectively (i.e. "$\not\vdash\neg\phi$" replaced with "$\vdash\phi$"). We summarize the relationship between these sets in the diagram in Figure 3. The main features to notice are that A^* is a linear structure, and that there is an equivalence of AF, AB, ARU, and AU.

Example 4.6 *We give an example of a database, and some of the items in each argument set. Take* $\Delta=\{\alpha,\neg\alpha\}$*. Then* $(\{\alpha,\neg\alpha\},\alpha\wedge\neg\alpha)\in\mathsf{An}(\Delta)$*,* $(\{\alpha\},\alpha)\in\mathsf{A}\exists(\Delta)$*,* $(\{\alpha\},\alpha\vee\beta)\in\mathsf{AR}(\Delta)$*, if* $\beta\not\vdash\alpha$*,* $(\{\},\alpha\vee\neg\alpha)\in\mathsf{A}\forall(\Delta)$*. Furthermore,* $\mathsf{A}\forall(\Delta)=\mathsf{AF}(\Delta)=\mathsf{AB}(\Delta)=\mathsf{ARU}(\Delta)=\mathsf{AU}(\Delta)=\mathsf{AT}(\Delta)$*.*

Example 4.7 *As another example, consider* $\Delta=\{\neg\alpha\wedge\beta,\alpha\wedge\beta\}$*. Then for* $\Pi=\{\alpha\wedge\beta\}$*,* $(\Pi,\beta)\in\mathsf{A}\exists(\Delta)$*,* $(\Pi,\beta)\in\mathsf{AR}(\Delta)$*, and* $(\Pi,\beta)\in\mathsf{A}\forall(\Delta)$*. But there is no* $\Pi\subseteq\Delta$ *such that* $(\Pi,\beta)\in\mathsf{AU}(\Delta)$*,* $(\Pi,\beta)\in\mathsf{ARU}(\Delta)$*,* $(\Pi,\beta)\in\mathsf{AB}(\Delta)$*, or* $(\Pi,\beta)\in\mathsf{AF}(\Delta)$*.*

Each argument set in A^* induces a consequence relation.

$$
\begin{array}{c}
\mathsf{AT}(\Delta) \\
| \\
\mathsf{AF}(\Delta) = \mathsf{AB}(\Delta) = \mathsf{ARU}(\Delta) = \mathsf{AU}(\Delta) \\
| \\
\mathsf{A}\forall(\Delta) \\
| \\
\mathsf{AR}(\Delta) \\
| \\
\mathsf{A}\exists(\Delta) \\
| \\
\mathsf{An}(\Delta)
\end{array}
$$

Figure 3: Partial order on A^* induced by $\subseteq$. Sets lower in the diagram are subsets of sets higher in the diagram.

Definition 4.11 *A consequence closure for each argumentative structure is denoted* Cx*, where* $\mathsf{x} \in \{\mathsf{T}, \mathsf{F}, \mathsf{B}, \mathsf{RU}, \mathsf{U}, \forall, \mathsf{R}, \exists, \mathsf{n}\}$*, and defined as follows,*

$$\mathsf{Cx}(\Delta) = \{\phi | \exists \Pi \subseteq \Delta (\Pi, \phi) \in \mathsf{Ax}(\Delta)\}$$

A consequence relation, denoted $\vdash_{\mathsf{X}}$, can then be generated from the consequence closure as expected.

$$\Delta \vdash_{\mathsf{X}} \phi \text{ iff } \phi \in \mathsf{Cx}(\Delta)$$

The argumentative logics are, in a key sense, far more restricted than the other paraconsistent logics considered in this review: If a pair of formulae are mutually inconsistent, then none of these argumentative logics will derive any consequences from the conjunction of the two formulae (as an example see Example 4.8). This is not the case with any of the weakly-negative, four-valued or quasi-classical logics. Another significant drawback is the computational complexity of reasoning with argumentative logics since the reasoning is based on consistency checking.

Example 4.8 *Consider* $\{\alpha \wedge \beta, \neg\alpha \wedge (\beta \rightarrow \gamma\}$*. There is no consistent subset which gives the inference* γ*. In a sense,* γ *is locked into the inconsistency.*

The concept of an argumentative structure, with the two notions of argument and acceptability, are a convenient framework for developing practical reasoning tools. Although, they are based on simple definitions of arguments and acceptability, the concepts carry many possibilities for further refinement. It remains to be seen whether there is a general taxonomy of argumentative structures, such as suggested by Pinkas and Loui [PL92], and universal properties of the logics that they induce.

There are also a number of other argument-based systems that have been proposed, including by Vreeswijk [Vre91, Vre97], Wagner [Wag91], Prakken [Pra93], Roos [Roo93], Fox *et al* [FKA92, KAEF94, DFK96], Simari and Loui [SL92], Lin [Lin94], and Parsons [Par96]. For a review of modelling argumentation in non-classical logics see [PV99]. These differ from argumentative logics in that they focus on defeasible reasoning: They incorporate defeasible, or probabilistic, connectives into their languages, together with associated machinery.

Another approach to acceptability of arguments is by Dung [Dun95]. This approach assumes a set of arguments, and a binary "attacks" relation between pairs of subsets of arguments. A hierarchy of arguments is then defined in terms of the relative attacks "for" and "against" each argument in each subset of arguments. In this way, for example, the plausibility of an argument could be defended by another argument in its subset.

The common feature in argument systems is that they incorporate formal representation of individual arguments and techniques for comparing conflicting arguments. In these frameworks, if there are a number of arguments for and against a particular conclusion, an aggregation function determines whether the conclusion holds in the framework. These aggregation functions can be described as being binary since they just consider the existence of arguments for and against, and so they are not sensitive to the number of arguments for or against.

4.4 Quasi-classical logic

As we have seen with weakly-negative logics and with four-valued logics, the weakening of the proof theory means that the connectives do not behave in a classical fashion. To address this, an alternative called quasi-classical logic has been proposed by Besnard and Hunter [BH95, Hun99a]. In this, the proof theory is restricted so that the application of introduction rules cannot be followed by the application of elimination rules. The proof theory allows any formula to be rewritten into CNF (using distributivity, associativity, de Morgan's laws, double negation elimination, and arrow elimination[2]), and for conjunction elimination and resolution to be applied. From the resulting resolvents, the following rules can be applied to obtain an inference: distributivity, associativity, de Morgan's laws, double negation introduction, conjunction introduction, disjunction introduction, and arrow introduction[3].

Example 4.9 *For $\Delta = \{\alpha \vee \beta, \alpha \vee \neg\beta, \neg\alpha \wedge \delta\}$, consequences of Δ include $\alpha \vee \beta$, $\alpha \vee \neg\beta$, α, $\neg\alpha$, and δ, but do not include $\neg\delta$, γ, $\gamma \vee \phi$, or $\neg\psi \wedge \neg\phi$. For $\Delta = \{\alpha \vee (\beta \wedge \gamma), \neg\beta\}$, consequences of Δ include $\alpha \vee \beta$, $\alpha \vee \gamma$, α, and $\neg\beta$. For $\Delta = \{\alpha \vee \beta, \neg\beta, \alpha \rightarrow \gamma, \delta \rightarrow \neg\gamma\}$, consequences of Δ include α,γ, δ, and $\alpha \wedge \gamma \wedge \delta$.*

QC logic is motivated by the need to handle beliefs rather than the need to address issues of verisimilitude for given propositions. The aim is for a logic of beliefs in the "real-world" rather than a logic of truths in the real-world. In this logic, we can regard each formula as a belief.

The heart of the QC proof theory is resolution. Resolution can be regarded as a process of focusing beliefs. So a resolvent γ is more focused than the clause $\neg\alpha \vee \gamma$. Similarly, for the pair of beliefs $\alpha \vee \beta$ and $\neg\alpha \vee \gamma$, the resolvent $\beta \vee \gamma$ is more focused. In general, a clause α is more focused than a clause β if $Atoms(\alpha) \subset Atoms(\beta)$. Hence, as one or more applications of resolution decomposes a set of assumptions, it focuses the beliefs from the assumptions.

A useful property of resolution is that α is a resolvent only if all the literals used in α are literals used in the set of assumptions (assuming, of course, that resolution is the only proof rule used). This means that any resolvent, and hence any belief from the assumptions, is a non-trivial inference from the assumptions. This holds even if the set of assumptions is classically inconsistent. As a result, resolution can constitute the basis of useful paraconsistent reasoning. Restricting the proofs so that the introduction proof rules are not followed by the elimination proof rules, as discussed above, means that the paraconsistent reasoning is preserved.

Now, we introduce the following definition for a model.

[2] Arrow elimination applied to a formula $\alpha \rightarrow \beta$ gives $\neg\alpha \vee \beta$.
[3] Arrow introduction applied to a formula $\neg\alpha \vee \beta$ gives $\alpha \rightarrow \beta$.

Definition 4.12 *Let $\mathcal{A}$ be the set of atoms in the language for QC logic. Let $\mathcal{O}$ be the set of objects defined as follows, where $+\alpha$ is a positive object, and $-\alpha$ is a negative object.*

$$\mathcal{O} = \{+\alpha \mid \alpha \in \mathcal{A}\} \cup \{-\alpha \mid \alpha \in \mathcal{A}\}$$

Any $X \in \wp(\mathcal{O})$ is a model. So $+\alpha \in X$ means that in X there is **a reason for** *the belief α and that in X there is* **a reason against** *the belief $\neg\alpha$. Similarly, $-\alpha \in X$ means that in X there is* **a reason against** *the belief α and that in X there is* **a reason for** *the belief $\neg\alpha$.*

A model just contains reasons for/against beliefs — it incorporates no notion of truth or falsity. Since we can allow both an atom and its complement to be satisfiable, we have decoupled, at the level of the model, the link between a formula and its complement. We now consider a key part of the definition for the satisfiability relation for the logic.

Definition 4.13 *Let $\models_s$ be a satisfiability relation, called strong satisfaction. For $X \in \wp(\mathcal{O})$, we define $\models_s$ as follows, and $\alpha_1, ..., \alpha_n$ are literals in $\mathcal{L}$, and α is an atom in $\mathcal{L}$.*

$X \models_s \alpha$ *if* $+\alpha \in X$

$X \models_s \neg\alpha$ *if* $-\alpha \in X$

$X \models_s \alpha_1 \vee ... \vee \alpha_n$
iff $[X \models_s \alpha_1$ *or ... or* $X \models_s \alpha_n]$
and $\forall i$ *s.t.* $1 \leq i \leq n$ $[X \models_s \sim\alpha_i$ *implies* $X \models_s Focus(\alpha_1 \vee ... \vee \alpha_n, \alpha_i)]$

where $\sim\alpha_i$ is the complementary literal of α_i, and $Focus(\alpha_1 \vee ... \vee \alpha_n, \alpha_i)$ is just the original formula $\alpha_1 \vee ... \vee \alpha_n$ without the disjunct α_i.

The first two parts of this definition covers literals. The third part covers disjunction. This definition for disjunction is more restricted than the classical definition. In addition to at least one disjunct being satisfiable, there is also a notion of focusing incorporated into the definition. Essentially, for each disjunct α_i in the formula, $\alpha_1 \vee ... \vee \alpha_n$, if the model satisfies the complement $\sim\alpha_i$ of that disjunct, then the model must also satisfy the focused formula $Focus(\alpha_1 \vee ... \vee \alpha_n, \alpha_i)$, where the focused formula is just the original formula without the disjunct α_i. The definition for satisfaction is easily extended to any formula of the language, and we can use it for a definition for entailment.

The reason we need this definition for disjunction that is more restricted than the classical version, is that we have decoupled the link between a formula and its negation in the model. Therefore, in order to provide a meaning for resolution, we need to put the link between each disjunct, and its complement, into the definition for disjunction. As a result, to ensure a clause is satisfiable, we need to ensure that if necessary, every more focused clause is also satisfiable.

Example 4.10 *Let $\Delta = \{(\neg\alpha \vee \neg\beta) \vee \gamma, \neg\alpha \vee \gamma, \neg\gamma\}$, where $\alpha, \beta, \gamma \in \mathcal{A}$, and let $X = \{-\alpha, -\beta, -\gamma\}$. So $X \models_s \neg\alpha$, $X \models_s \neg\beta$ and $X \models_s \neg\gamma$. Hence, $X \models_s \neg\beta \vee \gamma$, $X \models_s \neg\alpha \vee \gamma$, and $X \models_s \neg\alpha \vee \neg\beta$. Finally, $X \models_s (\neg\alpha \vee \neg\beta) \vee \gamma$. So every formula in Δ is satisfiable in X.*

The QC consequence relation offers many more non-tautological inferences from data than either the weakly-negative or four-valued logics. For example, via disjunctive syllogism, QC logic gives β from $\{\neg\alpha, \alpha \vee \beta\}$, whereas neither the weakly-negative logic C_ω nor the four-valued logic gives β.

In addition, there is a form of relevancy in the reasoning. If there is an inference of α from some assumption, then there is at least one propositional atom in common between α and the assumptions.

Developing a non-trivializable, or paraconsistent logic, necessitates some compromise, or weakening, of classical logic. The compromises imposed to give QC logic seem to be more appropriate than other paraconsistent logics for applications in computing. QC logic provides a means to obtain all the non-trivial resolvents from a set of formulae, without the problem of trivial clauses also following. Furthermore, QC logic is tractable unlike argumentative logics. Though the constraints on QC logic result in tautologies from an empty set of assumptions being non-derivable, this is not usually a problem for applications.

Whilst all paraconsistent logics fail to adhere to all the properties of the classical consequence relation, QC logic does fail a particularly significant combination of these properties. In particular, the following properties do not hold for the QC consequence relation:

$$
\begin{array}{rl}
\Delta \vdash \alpha \text{ and } \Delta \vdash \alpha \rightarrow \beta \text{ implies } \Delta \vdash_Q \beta & \text{right modus ponens} \\
\Delta \cup \{\alpha\} \vdash_Q \beta \text{ implies } \Delta \vdash_Q \alpha \rightarrow \beta & \text{conditionalization} \\
\Delta \vdash_Q \alpha \rightarrow \beta \text{ implies } \Delta \cup \{\alpha\} \vdash_Q \beta & \text{deduction} \\
\Delta \cup \{\alpha\} \vdash_Q \beta \text{ and } \Gamma \vdash_Q \alpha \text{ implies } \Delta \cup \Gamma \vdash_Q \beta & \text{cut} \\
\Delta \vdash_Q \alpha \text{ and } \vdash \alpha \rightarrow \beta \text{ implies } \Delta \vdash_Q \beta & \text{right weakening}
\end{array}
$$

Failure of these properties can be detrimental for some applications. However, for applications where inferences are obtained as a one-step process, for example in a system that just provides a user with inferences from an item of structured text, these properties are often not required.

5 Paraconsistent reasoning with structured text

Now we have reviewed key approaches to paraconsistent reasoning, we return to the question of paraconsistent reasoning with structured text. First, we summarize the application requirements. We assume we have a set of items of structured text, and that we will represent these by a set of positive literals. We also assume that we have domain knowledge represented by a set of classical formulae. In addition, let us assume that we are using paraconsistent reasoning to look at the structured text, to identify possible inferences from the structured text and domain knowledge, and let us also assume the domain knowledge is consistent.

To illustrate, we will consider weather reports in the form of structured text. In this application we may assume some scientific knowledge about the nature of weather. We will regard this scientific knowledge as the domain knowledge. We will also assume that the aim of paraconsistent reasoning with the weather reports is to find useful inferences from each weather report and perhaps to merge information from the reports. We look at this example in more detail below.

We have considered four options for paraconsistent reasoning, namely C_ω logic, four-valued logic, quasi-classical logic, and argumentative logics. We summarize the pros and cons of each of these options for handling structured text as follows, taking into account the application requirements we have assumed:

- C_ω could be too weak for reasoning with domain knowledge. In particular key classical proof rules such as modus tollens and disjunctive syllogism do not hold and various classical equivalences do not hold. This means that unless there is a full appreciation of the intricacies of the proof theory, it will be difficult to engineer a knowledgebase to behave as expected. As a simple example, if a clause is inserted into the knowledgebase using disjunction as opposed to implication, then obvious inferences that would follow by modus ponens would not follow by resolution. Another problem with this logic is that there is no qualification of inferences.

- Four-valued logic could be useful for aggregating information about literals, but the proof theory is too weak for reasoning with domain knowledge. In particular the notion of implication is much weaker than the classical notion, and there are no proof rules such as modus ponens or resolution. Whilst there is some qualification of inferences, via truth values, it only says whether or not there is a conflict.

- Argumentative logics give useful inferences that are intuitive plus there is a qualification of these inferences. The proof theory is based on classical logic, and in many respects these logics behave as classical logic. There is no need to restrict the language of the domain knowledge, and so long as the domain knowledge is consistent, then there is no problem with knowledge being locked into formulae. In other words, under these conditions argumentative logics are ideal[4]. Furthermore, the inferences are qualified.

- Quasi-classical is in some respects more powerful[5] than argumentative logics and qualification can be incorporated as discussed in [HN98]. However, if we assume that the domain knowledge is consistent and correct, then quasi-classical logic does not seem necessary. Though, the computational complexity of QC logic is significantly better than argumentative logics. If we were to allow inconsistent domain knowledgebases, then QC logic would be ideal.

The net conclusion we can draw from this is that argumentative logics are ideal for reasoning with structured text if we assume that the domain knowledge is consistent.

Since each item of structured text is represented by a set of positive literals, we can only obtain an inconsistency in a structured report, or in a set of structured reports, by reasoning with domain knowledge. So each inconsistency in a structured report is defined with respect to the domain knowledge, and as we will see we need to axiomatize this inconsistency. Some of the axiomatization may involve unique name axioms and domain closure axioms that are defined using the appropriate stereotypes.

Now to illustrate using argumentative logics for reasoning with inconsistency in structured text, we will consider the problem of the following two conflicting weather reports.

```
⟨weather report:
      ⟨source: TV⟩,
      ⟨date: 19.5.1999⟩,
      ⟨city: London⟩,
      ⟨today's weather: sun⟩,
      ⟨tomorrow's weather: sun⟩
:weather report⟩
```

[4] As we saw in Section 4.3, there is a range of argumentative logics. However, they are all based on reasoning with maximally consistent subsets. So we can use the argumentative logics in parallel by finding these subsets, drawing all existential arguments, and then qualifying them according to whether they are also universal, backed, etc.

[5] By "more powerful", we mean that amongest some of the advantageous properties of quasi-classical logic, information does not get locked into inconsistent subsets of formulae (cf. argmentative logics with Example 4.8).

```
⟨weather report:
     ⟨source: Radio⟩,
     ⟨date: 19.5.1999⟩,
     ⟨city: London⟩,
     ⟨today's weather: sun⟩,
     ⟨tomorrow's weather: rain⟩
:weather report⟩
```

The reports can be represented by the following two formulae.

$$\texttt{report(r1)} \wedge \texttt{date(r1, 19.5.1999)} \wedge \texttt{city(r1, London)} \wedge \texttt{today(r1, sun)} \wedge \texttt{tomorrow(r1, sun)}$$

$$\texttt{report(r2)} \wedge \texttt{date(r2, 19.5.1999)} \wedge \texttt{city(r2, London)} \wedge \texttt{today(r2, sun)} \wedge \texttt{tomorrow(r2, rain)}$$

Let us assume we have the following domain knowledge for identifying inconsistency in weather reports:

$$\begin{array}{c} \texttt{date(X, D)} \wedge \texttt{date(Y, D)} \wedge \texttt{city(X, C)} \wedge \texttt{city(Y, C)} \\ \wedge\, \texttt{today(X, A)} \wedge \texttt{today(Y, B)} \wedge \texttt{incoherent(A, B)} \rightarrow \bot \end{array}$$

$$\begin{array}{c} \texttt{date(X, D)} \wedge \texttt{date(Y, D)} \wedge \texttt{city(X, C)} \wedge \texttt{city(Y, C)} \\ \wedge\, \texttt{today(X, A)} \wedge \texttt{tomorrow(Y, B)} \wedge \texttt{incoherent(A, B)} \rightarrow \bot \end{array}$$

$$\begin{array}{c} \texttt{incoherent(sun, rain)} \\ \texttt{incoherent(cold, hot)} \\ \texttt{incoherent(cool, hot)} \\ \texttt{incoherent(cold, warm)} \\ \texttt{incoherent(cool, warm)} \\ \texttt{incoherent(sun, snow)} \end{array}$$

$$\begin{array}{c} \forall \texttt{X, Y incoherent(X, Y)} \rightarrow \texttt{incoherent(Y, X)} \\ \forall \texttt{X coherent(X, X)} \\ \forall \texttt{X, Y coherent(X, Y)} \leftrightarrow \neg \texttt{incoherent(X, Y)} \end{array}$$

Let us also assume we have domain knowledge for merging weather reports where + is a function denoting a form of conjunction. This domain knowledge includes axioms for associativity, commutativity and idempotence for the + function symbol:

$$\texttt{today(X, A)} \wedge \texttt{today(Y, B)} \wedge \texttt{X} \neq \texttt{Y} \wedge \texttt{coherent(A, B)} \rightarrow \texttt{today(X + Y, A + B)}$$

$$\begin{array}{c} \texttt{today(X, (A + B) + C)} \leftrightarrow \texttt{today(X, A + (B + C))} \\ \texttt{today(X, A + B)} \leftrightarrow \texttt{today(X, A + B)} \\ \texttt{today(X, A + (B + B))} \leftrightarrow \texttt{today(X, A + B)} \end{array}$$

So for example from `today(r1,sun)` and `today(r2,sun)`, we get `today(r1+r2,sun+sun)`, and hence `today(r1+r2,sun)`.

Now we return to qualifying inferences. Using this example we obtain `today(r1+r2,sun)` as a free inference from Δ, whereas `tomorrow(r1,sun)` and `tomorrow(r2,rain)` are existential inferences from Δ.

Now suppose we have the followng further formulae in the domain knowledge:

$$\forall \mathtt{X}(\mathtt{tomorrow}(\mathtt{X}, \mathtt{sun}) \rightarrow \neg \mathtt{snow\text{-}is\text{-}forecast\text{-}tomorrow})$$
$$\forall \mathtt{X}(\mathtt{tomorrow}(\mathtt{X}, \mathtt{rain}) \rightarrow \neg \mathtt{snow\text{-}is\text{-}forecast\text{-}tomorrow})$$

Also suppose we restrict universal inferences to maximally consistent subsets that contain all domain knowledge, then all maximally consistent subsets imply $\neg$**snow-is-forcast-tomorrow**, and so this inference is a universal inference. The reason we need to restrict the universal inferences is that we are assuming the domain knowledge is consistent and "correct". So if the union of the domain knowledge and structured text is inconsistent, then we assume it is because the structured text contains inaccuracies. Hence, we adopt the following definition of univeral inference: Let Δ be the domain knowledge, and let Γ be the formulae representing the structured text, then α is a universal inference from (Δ, Γ) iff α is an inference from every set in $\{\Delta \cup \Gamma' \mid \Gamma'$ is a maximal subset of Γ such that $\Gamma' \cup \Delta$ is consistent$\}$.

6 Analysing and acting on inconsistency

Paraconsistent reasoning is only really useful as part of a wider framework for inconsistency handling. Here we consider two key aspects of inconsistency handling that incorporate paraconsistent reasoning. These are analysing inconsistency and acting on inconsistency.

6.1 Analysing inconsistency

Analysis of inconsistency is needed to make better informed decisions on how to handle an inconsistency. In [HN98], we propose the use of *labelled* QC logic that records and tracks information used in reasoning. Proof rules of labelled QC logic can be used to track inconsistent information by propagating these labels (and their associated information) during reasoning. Using this approach we can provide a "logical analysis" of inconsistent information. We can identify the likely sources of the problem, and use this to suggest appropriate actions. This "auditing" is essential if we are to facilitate further development in the presence of inconsistency.

In [GH98, Hun99b], we explore further techniques for analysing sets of inconsistent information. In particular, we give measures of inconsistency for a set of formulae Δ and use this for defining ways to say "a set (of classical formulae) Φ is more inconsistent than a set Ψ" and to say "a set Φ is more inconsistent with a set Γ than a set Ψ is with Γ". These measures are based on features of a set of formulae include: the number of minimally inconsistent subsets; the number of maximally inconsistent subsets; the number of atomic symbols occuring in the minimally inconsistent subsets; the proportion of formulae occurring in a minimally inconsistent subset; and the proportion of formulae in more than one minimally inconsistent subset. These measures can be useful in describing inconsistency in prioritizing resolution tasks, and in particular in selecting appropriate actions.

In the context of structured text, such as weather reports, analysis might include trying to identify certain factors that could be the root of the inconsistencies. For example, we may find that one newspaper may frequently conflict with domain knowledge, and so should be regarded with increased doubt. Or we may find that a number of newspapers may largely agree for four out of the next five days, but on one day they significantly disagree — in this case we may regard that predicting the weather for that day is subject to a high degree of uncertainty and so prone to error.

6.2 Acting on inconsistency

There are various proposals for resolving inconsistency including truth maintenance systems [Doy79, Kle86], knowledgebase merging [BKMS92, CH97, KP98, KP99, LS98, CH99], and belief revision (starting with [AGM85, Gar88] and with developments including iterated belief revision [DP97, Leh95], and relationships with database updating [KM92]. For a review of belief revision theory see [DP98]. Yet we may not always want to immediately restore inconsistency. Since by restoring inconsistency, we may lose valuable information. We may want to harbour the inconsistency until we are better able to resolve it. This may involve seeking further information or advice.

In [GH91, GH93] we argue that dealing with inconsistencies is not necessarily done by restoring consistency but by supplying rules telling one how to act when the inconsistency arises. To illustrate, we can continue with the example of weather reports. For acting on inconsistency in a set of weather reports, we can choose to ignore some reports because they come from less reliable sources, or we can seek more up to date information by say switching on the radio, or we can seek more authoritative information by phoning a government weather service, or if there is a high degree of conflict we may just proceed in the basis that the worst case weather will occur.

Effective action in the presence of inconsistencies requires gathering a wider appreciation of the nature and context of these inconsistencies. If we accept that acting in the presence of inconsistency requires external input, we can adopt a meta-level approach to prescribe inconsistency handling rules of the form: *Inconsistency implies Action*.

One approach is to deploy an action-based temporal logic that allows us to specify the past context and source of an inconsistency in order to prescribe future actions to handle the inconsistency [GH93, FGH+94]. We can consider actions as being: (1) Ignoring the inconsistency completely and continuing development regardless; (2) Circumventing the inconsistent parts of the specification being developed and continuing development; (3) Removing the inconsistency altogether by correcting any mistakes or resolving conflicts; and (4) Ameliorating the inconsistent situations by performing actions that "improve" these situations and increase the possibility of future resolution.

7 Discussion

In this paper, we have focused on paraconsistent reasoning with structured text. There are many proposals for logics for paraconsistent reasoning. Here we have reviewed some of the main approaches and compared them with respect to an application in technology for structured text. From this comparison, we see that there are some useful logics for reasoning with inconsistency in structured text. This offers a formal basis for addressing the conflicts that normally arise in information such as news reports.

- Argumentative logics give useful inferences that are intuitive plus there is a qualification of these inferences.
- Four-valued logic is too weak for reasoning with domain knowledge — in particular the notion of implication is much weaker than the classical notion — and inferences are only qualified according to whether or not there is a conflict.
- C_ω could be too weak for reasoning with domain knowledge — in particular key classical proof rules such as modus tollens and disjunctive syllogism do not hold — and there is no qualification of inferences.

- Quasi-classical logic is in some respects more powerful than argumentative logics — in particular if the domain knowledge is inconsistent — and qualification can be incorporated as discussed in [HN98].

Structured text is a general concept implicit in many approaches to handling textual information in computing — including tagged text in XML, text in relational and object-oriented databases, and output from information extraction systems. Structured text can be naturally viewed in logic. Each item of structured text can be represented by a formula of classical logic. This means that consistency checking and inferencing can be undertaken with structured text using domain knowledge. Given the vast amount of information that is potentially available in the form of structured text, in particular news reports, it is possible that analysis and alerting tools based on paraconsistent logics, together with tools based on defeasible reasoning [Hun00] and tools for merging reports [Hun98], could become a valuable technology.

Acknowledgements

The author would like to thank Sanjay Modgil and Simon Parsons for helpful feedback on a draft of this paper. The author is also very grateful to the anonymous referees who made a number of suggestions for improving this paper.

References

[AB75] A Anderson and N Belnap. *Entailment: The Logic of Relevance and Necessity*. Princeton University Press, 1975.

[Abi97] S Abiteboul. Querying semi-structured data. In *International Conference on Database Theory*, pages 1–18, 1997.

[AGM85] C Alchourrón, P Gärdenfors, and D Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[ARP98] ARPA. *Message Understanding Conference: Proceedings of the Seventh Conference*. Morgan Kaufmann, 1998.

[Arr77] A Arruda. On the imaginary logic of NA Vasilev. In A Arruda, N Da Costa, and R Chuaqui, editors, *Non-classical Logics, Model Theory and Computability*. North Holland, 1977.

[Bat80] D Batens. Paraconsistent extensional propositional logics. *Logique et Analyse*, 90–91:195–234, 1980.

[BDP93] S Benferhat, D Dubois, and H Prade. Argumentative inference in uncertain and inconsistent knowledge bases. In *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, pages 411–419. Morgan Kaufmann, 1993.

[BDP95] S Benferhat, D Dubois, and H Prade. A logical approach to reasoning under inconsistency in stratified knowledge bases. In Ch Froidevaux and J Kohlas, editors, *Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'95)*, volume 956 of *Lecture Notes in Computer Science*, pages 36–43. Springer, 1995.

[Bel77] N Belnap. A useful four valued logic. In G Epstein, editor, *Modern Uses of Multiple-valued Logic*. Reidel, 1977.

[Bes91] Ph Besnard. Paraconsistent logic approach to knowledge representation. In M de Glas and D Gabbay, editors, *Proceedings of the First World Conference on Fundamentals of Artificial Intelligence*, pages 107–114. Angkor, 1991.

[BH95] Ph Besnard and A Hunter. Quasi-classical logic: Non-trivializable classical reasoning from inconsistent information. In C Froidevaux and J Kohlas, editors, *Symbolic and Quantitative Approaches to Uncertainty*, volume 946 of *Lecture Notes in Computer Science*, pages 44–51, 1995.

[BKMS92] C Baral, S Kraus, J Minker, and V Subrahmanian. Combining knowledgebases of first-order theories. *Computational Intelligence*, 8:45–71, 1992.

[Bre89] G Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *Proceedings of the Eleventh International Conference on Artificial Intelligence (IJCAI'89)*, pages 1043–1048, 1989.

[Bun97] P Buneman. Semistructured data. In *Proceeings of the ACM Symposium on Principles of Database Systems*, 1997.

[CH97] L Cholvy and A Hunter. Information fusion in logic: A brief overview. In D Gabbay, R Kruse, A Nonnengart, and H-J Ohlbach, editors, *Symbolic and Quantitative Approaches to Uncertainty (ECSQARU'97)*, Lecture Notes in Computer Science, pages 86–95. Springer, 1997.

[CH99] L Cholvy and A Hunter. Merging requirements from ranked agents. Technical report, ONERA-CERT Toulouse, 1999.

[CL96] J Cowie and W Lehnert. Information extraction. *Communications of the ACM*, 39:81–91, 1996.

[CRS93] C Cayrol, V Royer, and C Saurel. Management of preferences in assumption based reasoning. In *Information Processing and the Management of Uncertainty in Knowledge based Systems (IPMU'92)*, volume 682 of *Lecture Notes in Computer Science*. Springer, 1993.

[dC74] N C da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 15:497–510, 1974.

[DFK96] S Das, J Fox, and P Kraus. A unified framework for hypothetical and practical reasoning: (1) theoretical foundations. In D Gabbay and H-J Ohlbach, editors, *Practical Reasoning*, volume 1085 of *Lecture Notes in Computer Science*, pages 58–72, 1996.

[Doy79] J Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[DP97] A Darwiche and J Pearl. On the logic of iterated belief revision. *Artificial Intelligence*, 89:1–29, 1997.

[DP98] D Dubois and H Prade, editors. *Handbook of Defeasible Resoning and Uncertainty Management Systems*, volume 3. Kluwer, 1998.

[Dun95] P. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[EGH95] M Elvang-Göransson and A Hunter. Argumentative logics: Reasoning from classically inconsistent information. *Data and Knowledge Engineering*, 16:125–145, 1995.

[FGH+94] A Finkelstein, D Gabbay, A Hunter, J Kramer, and B Nuseibeh. Inconsistency handling in multi-perspective specifications. *Transactions on Software Engineering*, 20(8):569–578, 1994.

[FKA92] J Fox, P Krause, and S Ambler. Arguments, contradictions and proactical reasoning. In *Proceedings of the European Conference on Artificial Intelligence*, 1992.

[Gar88] P Gardenfors. *Knowledge in Flux.* MIT Press, 1988.

[GH91] D Gabbay and A Hunter. Making inconsistency respectable 1: A logical framework for inconsistency in reasoning. In Ph Jorrand and J Keleman, editors, *Fundamentals of Artificial Intelligence*, volume 535 of *Lecture Notes in Computer Science*, pages 19–32. Springer, 1991.

[GH93] D Gabbay and A Hunter. Making inconsistency respectable 2: Meta-level handling of inconsistent data. In M Clarke, R Kruse, and S Moral, editors, *Symbolic and Qualitative Approaches to Reasoning and Uncertainty (ECSQARU'93)*, volume 747 of *Lecture Notes in Computer Science*, pages 129–136. Springer, 1993.

[GH98] D Gabbay and A Hunter. Negation and contradiction. In D Gabbay and H Wansing, editors, *What is negation?*, volume 13 of *Applied Logic Series.* Kluwer, 1998.

[GQ99] I Graham and L Quin. *XML Specification Guide.* Wiley, 1999.

[Gri97] R Grishman. Information extraction techniques and challenges. In M Pazienza, editor, *Information Extraction.* Springer, 1997.

[Haa78] S Haack. *The Philosophy of Logics.* Cambridge University Press, 1978.

[HN98] A Hunter and B Nuseibeh. Managing inconsistent specifications: Reasoning, analysis and action. *ACM Transactions on Software Engineering Methodology*, 7:335–367, 1998.

[Hun98] A Hunter. Merging inconsistent items of structured text. Technical report, Department of Computer Science, University College London, 1998.

[Hun99a] A Hunter. Reasoning with contradictory information using quasi-classical logic. *Journal of Logic and Computation*, 1999. (in press).

[Hun99b] A Hunter. Techniques for analysing sets of inconsistent formulae. Technical report, Department of Computer Science, University College London, 1999.

[Hun00] A Hunter. Ramification analysis using causal mapping. *Data and Knowledge Engineering*, 32:1–27, 2000.

[KAEF94] P Krause, S Ambler, M Elvang-Gorannson, and J Fox. A logic of argumentation for reasoning under uncertainty. *Computational Intelligence*, 11:113–131, 1994.

[Kle86] J De Kleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[KM92] H Katsuno and A Mendelzon. On the difference between updating a knowledgebase and revising it. *Belief Revision*, pages 183–203, 1992.

[KP98] S Konieczny and R Pino Perez. On the logic of merging. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR98)*, pages 488–498. Morgan Kaufmann, 1998.

[KP99] S Konieczny and R Pino Perez. Merging with integrity constraints. In Anthony Hunter and Simon Parsons, editors, *Qualitative and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'99)*, volume 1638 of *Lecture Notes in Computer Science.* Springer, 1999.

[Leh95] D Lehmann. Belief revision, revised. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1534–1540, 1995.

[Lin94] J Lin. A logic for reasoning consistently in the presence of inconsistency. In *Proceedings of the Fifth Conference on Theoretical Aspects of Reasoning about Knowledge*. Morgan Kaufmann, 1994.

[LS98] P Liberatore and M Schaerf. Arbitration (or how to merge knowledgebases). *IEEE Transactions on Knowledge and Data Engineering*, 10:76–90, 1998.

[MR70] R Manor and N Rescher. On inferences from inconsistent information. *Theory and Decision*, 1:179–219, 1970.

[MS88] J Martins and S Shapiro. A model of belief revision. *Artificial Intelligence*, 35:25–79, 1988.

[Par96] S Parsons. Defining normative systems for qualitative argumentation. In D Gabbay and H-J Ohlbach, editors, *Practical Reasoning*, volume 1085 of *Lecture Notes in Computer Science*, pages 449–463, 1996.

[Pfa99] B Pfaffenberger. *Web Publishing with XML in 6 easy steps.* AP Professional, 1999.

[PL92] G Pinkas and R Loui. Reasoning from inconsistency: A taxonomy of principles for resolving conflict. In B Nebel, C Rich, and W Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference.* Morgan Kaufmann, 1992.

[Poo85] D Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1985.

[Pra93] H Prakken. An argument framework for default reasoning. In *Annals of Mathematics and Artificial Intelligence*, volume 9, 1993.

[PV99] H Prakken and G Vreeswijk. Modelling argumentation in logic. In D Gabbay, editor, *Handbook of Philosophical Logic*. Kluwer, 1999.

[Roo93] N Roos. A logic for reasoning with inconsistent knowledge. *Artificial Intelligence*, 57(1):69–104, 1993.

[SL92] G. Simari and R. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, 53:125–157, 1992.

[VM98] C Viegas Damasio and L Moniz Pereira. A survey of paraconsistent semantics for logic programs. In D Gabbay and Ph Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 2, pages 241–320. Kluwer, 1998.

[Vre91] G Vreeswijk. Abstract argumentation systems. In M de Glas and D Gabbay, editors, *Proceedings of the First World Conference on Fundamentals of Artificial Intelligence.* Angkor, 1991.

[Vre97] G Vreeswijk. Abstract argumentation systems. *Artificial Intelligence*, 90:225–279, 1997.

[Wag91] G Wagner. Ex contradictione nihil sequitur. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'91)*, 1991.