Merging potentially inconsistent items of structured text

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

April 11, 2000

Abstract

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 paper, we formalize the concept of structured text, and then focus on how we can identify inconsistency in the logical representation of items of structured text. We then present a new framework for merging logical theories that can be employed to merge inconsistent items of structured text. To illustrate, we consider the problem of merging reports such as weather reports.

Keywords: Merging information, structured text, semi-structured data, knowledge engineering, logic-based inconsistency management techniques.

1 Introduction

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 [Hun00b].

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, and the output from information extraction systems given in templates (see for example [CL96, Gri97, ARP98]). 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. Here we focus on inconsistencies that can arise between items of structured text and how this affects the ways we can merge them. As a simple example, suppose we have two weather reports for London for tomorrow midday. If report1 says `rain` and report2 says `sunny`, we have an inconsistency. Whereas if report2 says `showers`, then we don't have an inconsistency.

In this paper, we formalize the concept of structured text, and then consider how we can identify inconsistency in items of structured text. We have assumed the words and phrases are sufficiently simple and restricted to not require natural language processing. We will represent each word or phrase by a constant symbol in the logic, and each semantic label as a relation symbol. We then present a new framework for merging logical theories that can be employed to merge inconsistent items of structured text. In particular, we consider the problem of merging weather reports in structured text.

Rather than draw on existing techniques for merging inconsistent information in logic (for a review of existing techniques see [CH97]), we argue that application dependent axioms are required for defining how items of structured text should be merged. In our approach, a merging system for structured text incorporates both domain knowledge and merging axioms. The input for a merging system is a set of items of structured text, and the output is a single item of structured text that is consistent with the domain knowledge. We conclude the paper by comparing our approach with other logical approaches to merging information.

2 Formalizing structured text

In this section we will formalize the notions of structured text and of skeletons. The later notion is a way of abstracting the structure of a set of items of structured text. We will then explain how we can translate each item of structured into a set of literals.

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

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

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:*

$$\langle \texttt{city} : \texttt{London} \rangle$$

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

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.*

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

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

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 important. So for example, the following two are assumed to be not equivalent:

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

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

Definition 2.4 *Let $\langle \phi : \sigma_1, .., \sigma_n : \phi \rangle$ be a complex feature. The* **sub** *function is defined as follows:*

$$Sub(\langle \phi : \sigma_1, .., \sigma_n : \phi \rangle) = \{\sigma_1, .., \sigma_n\} \cup Sub(\sigma_1) \cup .. \cup Sub(\sigma_n)$$

$$Sub(\langle \phi : \psi \rangle) = \{\langle \phi : \psi \rangle\}$$

For complex features α, β, α is a complex feature in β iff $\alpha \in Sub(\beta)$.

We can consider a complex feature as a tree where the semantic labels are non-leaf nodes and the text entries are leaves.

Definition 2.5 *Let $\langle \phi : \sigma_1, .., \sigma_n : \phi \rangle$ be a complex feature. The semantic label ϕ is the* **parent** *of the complex features $\sigma_1, .., \sigma_n$. The elements of $Sub(\langle \phi : \sigma_1, .., \sigma_n : \phi \rangle)$ are the* **offspring** *of ϕ.*

Definition 2.6 *Let $\langle \phi : \sigma_1, .., \sigma_n : \phi \rangle$ be a complex feature. The function Root is defined as follows:*

$$Root(\langle \phi : \sigma_1, .., \sigma_n : \phi \rangle) = \phi$$

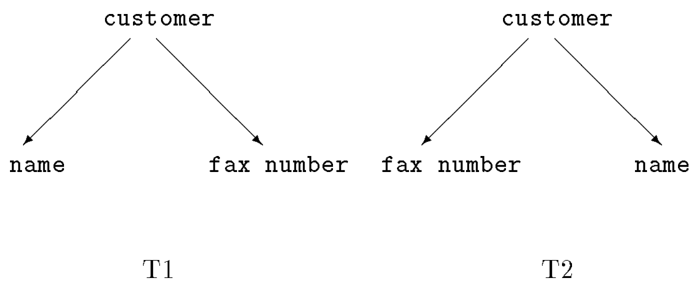

Figure 1: If `customer`, `name`, and `fax number`, are semantic labels, then T1 and T2 are both skeletons. Note they are not equivalent since siblings `name` and `fax number` are such that `name` is left of `fax number` in T1 and right of `fax number` in T2.

Definition 2.7 *The complex features,* $\langle \phi : \sigma_1, .., \sigma_n : \phi \rangle$ *and* $\langle \phi : \psi_1, .., \psi_n : \phi \rangle$ *have the* **same structure** *iff* σ_1 *and* ψ_1 *have the same structure, .., and* σ_n *and* ψ_n *have the same structure. The atomic features* $\langle \phi : \alpha \rangle$ *and* $\langle \phi : \beta \rangle$ *have the same structure.*

We assume that for an application, some complex features will be classified as **structured reports**. These will be complex features with some minimum structure. As an illustration, we may choose to regard Example 2.3 as a structured report since it contains some of the atomic features necessary for a weather report. We do not however assume any general conditions for classifying items of structured text as structured reports.

Definition 2.8 *If* θ *is a structured report, then* θ *has a unique reference label.*

All we require for Definition 2.8 is that if we have a set of structured reports, then we can refer to individual members by their reference label.

2.2 Skeletons

In order to compare items of structured text on the basis of their structure, we will use the following notion of a skeleton.

Definition 2.9 *A* **skeleton** *is a tree* (N, A, S) *defined as follows:* N *is the set of nodes where each node is a semantic label;* A *is a set of arcs represented by pairs of nodes; and* S *is the set of sibling neighbours represented by pairs of nodes such that* $(x, y) \in S$ *iff (i)* x *and* y *are siblings (i.e.* x *and* y *have the same parent) and (ii)* x *is to the left of* y.

According to Definition 2.9, the relative positions of siblings is important in a skeleton. So if x and y are siblings in a skeleton T, such that x is left of y, then we can form a different skeleton T' where y is left of x. As an illustration, see Figure 1.

Since a skeleton is essentially a complex feature without the text entries, a skeleton can be formed from an item by just removing the text entries.

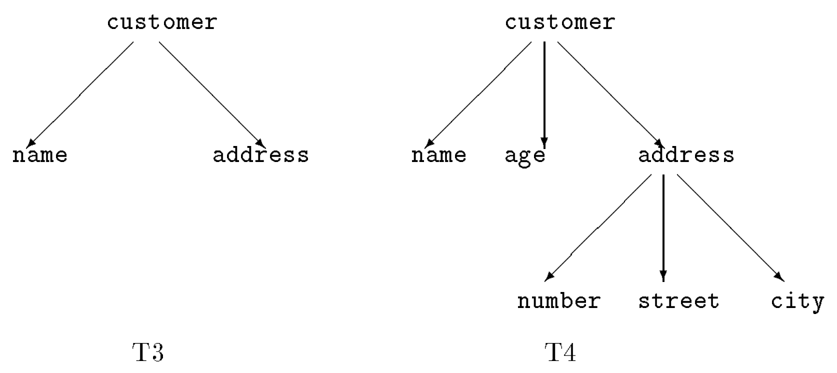

Figure 2: Assume T3 and T4 are both skeletons. Here, $T3 \preceq T4$ holds.

Definition 2.10 *The* **skeleton** *function, denoted Skeleton, is applied to a complex feature θ and returns the skeleton (N, A, S) for θ, where the set of nodes N is the set of semantic labels used in θ, and A and S are defined as follows:*

$$A = \{(\phi, Root(\psi_i)) \mid \langle \phi : \psi_1, ..., \psi_n : \phi \rangle \in Sub(\theta) \textit{ and } i \in \{1, ..., n\}\}$$

$$S = \{(Root(\psi_i), Root(\psi_j)) \mid \langle \phi : \psi_1, ..., \psi_n : \phi \rangle \in Sub(\theta) \textit{ and } i, j \in \{1, ..., n\} \textit{ and } i < j\}$$

So the skeleton function is defined to extract the tree structure discussed in Section 2.1. We also assume the extra constraint on skeletons that for each $\langle \phi : \psi_1, ..., \psi_n : \phi \rangle \in Sub(\theta)$, $Root(\psi_i)$ occurs to the left of $Root(\psi_j)$.

Example 2.4 *Let θ_1 be the following items of structured text:*

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

Then $Skeleton(\theta_1)$ is represented by T_1 in Figure 1.

Definition 2.11 *Let $T_i = (N_i, A_i, S_i)$ and $T_j = (N_j, A_j, S_j)$ be skeletons and let $\preceq$ be a* **pre-ordering over skeletons** *such that: $T_i \preceq T_j$ iff $N_i \subseteq N_j$ and $A_i \subseteq A_j$ and $S_i \subseteq S_j$.*

So for skeletons T_i and T_j, if we have $T_i \preceq T_j$, then the set of arcs in T_i is a subset of the arcs in T_j, and for all sibling nodes x, y in T_i, if x is left of y in T_i, then x is left of y in T_j.

An example of a pre-ordering is given in Figure 2.

Definition 2.12 *Let θ be a complex feature and let S be a skeleton. An* **instantiation** *of S by θ is defined as follows:*

$$\textit{If } Skeleton(\theta) \preceq S \textit{ then } \theta \textit{ is a partial instantiation of } S$$

$$\textit{If } Skeleton(\theta) \preceq S \textit{ and } S \preceq Skeleton(\theta) \textit{ then } \theta \textit{ is a full instantiation of } S$$

If θ is an instantiation of S, this is denoted by $S(\theta)$.

With reference to Definition 2.7, structured reports θ_1 and θ_2 have the same structure iff $Skeleton(\theta_1) \preceq Skeleton(\theta_2)$ and $Skeleton(\theta_2) \preceq Skeleton(\theta_1)$.

Definition 2.13 *Let θ be a structured report and let S be a skeleton such that $S(\theta)$ is an instantiation of θ. Since we have assumed that θ has a unique reference label and S is a tree, we can assume that each node in $S(\theta)$ has a unique* **reference label**.

From Definition 2.13, we can also assume that each occurrence of a semantic label in a structured report also has a unique reference label. In this way, we are using the notions of nodes in skeletons and semantic labels in structured reports interchangeably.

2.3 Representing structured reports as logical formulae

To represent structured reports as logical formulae, we adopt classical first-order logic. 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. We represent the classical logical consequence relation by $\vdash$, and logical inconsistency by $\perp$.

Now we consider how we can represent a structured report as a set of classical logic formulae. To do this, we harness the tree structure implicit in an item of structured text with the semantic labels as non-leaf nodes and the text entries as leaves.

Definition 2.14 *Let $\langle \phi : \sigma_1, .., \sigma_n : \phi \rangle$ be a structured report that we want to represent by logical formulae. We call ϕ the root of the item. We assume each semantic label in the complex feature has a reference label (as described in Definition 2.13). If a semantic label occurs more than once in a complex feature, then each occurrence will have a different reference label. The set of formulae that can be formed from this complex feature is obtained by exhaustively applying the following rules:*

1. *If X is a semantic label of a complex feature, and Y is the reference label for the parent of X, and Z is the reference label for X, then $X(Y, Z)$ is a formula.*
2. *If X is a semantic label of an atomic feature, and Y is the reference label for the parent of X, and Z is the text entry for the atomic feature, 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 reference label for X, then $X(Y)$ is a formula.*
4. *If X is the root of the tree and X is an atomic semantic label, and Y is the reference label for X, and Z is the text entry, then $X(Y, Z)$ is a formula.*

In Definition 2.14, Rule 1 gives the binary relations defining the path from the root to any other semantic label, Rule 2 gives the text entry for each branch, Rule 3 gives the reference label for the tree, and Rule 4 gives the text entry when the structured report is an atomic feature.

Example 2.5 *Consider the following item of structured text with the reference label* `e`.

```
⟨auction report :
      ⟨buyer : ⟨name : ⟨firstname : John⟩, ⟨surname : Smith⟩ : name⟩ : buyer⟩,
      ⟨seller : ⟨name : ⟨firstname : Mary⟩, ⟨surname : Jones⟩ : name⟩ : seller⟩
      ⟨property : Lot37⟩
: auction report⟩
```

From this item, we obtain the following set of formulae using Definition 2.14, where b*,* b1*,* s*,* s1*, and* p *are reference labels.*

$$\begin{gathered}\texttt{auctionreport(e)}, \texttt{buyer(e,b)}, \texttt{seller(e,s)}, \texttt{property(e,p)},\\ \texttt{name(b,b1)}, \texttt{firstname(b1,John)}, \texttt{surname(b1,Smith)},\\ \texttt{name(s,s1)}, \texttt{firstname(s1,Mary)}, \texttt{surname(s1,Jones)},\\ \texttt{property(p,Lot37)}\end{gathered}$$

Definition 2.15 *The* **formulate** *function, denoted $Formulate$, is defined as follows: For a structured report θ, let $Formulate(\theta)$ denote a set of formulae obtained by exhaustive application of Definition 2.14 to θ. For a set of structured reports $\Theta = \{\theta_1, .., \theta_n\}$, let $Formulate(\Theta) = Formulate(\theta_1) \cup .. \cup Formulate(\theta_n)$.*

For a structured report θ, $Formulate(\theta)$ is not unique, since the reference labelling is not unique. If we assume appropriate equality axioms for the labels, then $Formulate(\theta)$ is unique.

In the next section we show how we can use the logical representation of structured reports to identify inconsistency. Then in the subsequent sections, we show how we can merge structured reports using merging axioms.

3 Inconsistency in structured reports

To find inconsistencies in structured reports, we use the logical representation obtained via Definition 2.14. Since this definition only gives a set of positive literals for each structured report, 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 structured text is defined with respect to the domain knowledge.

Definition 3.1 *For an application,* **domain knowledge***, denoted Δ, is a set of classical first-order formulae. We will add some constraints on the language for any Δ in this section.*

The choice of axioms in the domain knowledge depends on the application. We explain in the following definitions and examples the kinds of formulae that are included in such a set.

First, we need to clarify types of inconsistency. An inter-item inconsistency is an inconsistency that arises between two or more structured reports together with the domain knowledge, whereas an intra-item inconsistency is an inconsistency that arises in an individual structured report together with the domain knowledge. For merging, intra-item inconsistency may just be viewed as a special case of inter-item inconsistency. Alternatively, an item that contains intra-item inconsistency might be viewed as generally erroneous and hence ignored. In this paper, we will assume that no items have intra-item inconsistency, and so will just focus our attention on inter-item inconsistency.

The simplest form of inconsistency is between a pair of atomic features. Consider two structured reports, θ_1 and θ_2, where the atomic feature $\langle\alpha : \phi\rangle$ is in item θ_1 and the atomic feature $\langle\alpha : \psi\rangle$ is in item θ_2 and $\phi \neq \psi$. For some semantic labels, this inequality would suggest an inconsistency with the domain knowledge, as illustrated by Example 3.1. Obviously different text entries for the same semantic label do not always suggest an inconsistency, as illustrated by Example 3.2.

Example 3.1 *Let θ_1 and θ_2 be two structured reports. Suppose $\langle\texttt{today} : \texttt{sun}\rangle$ is an atomic feature in θ_1 and $\langle\texttt{today} : \texttt{rain}\rangle$ is an atomic feature in θ_2, and θ_1 and θ_2 are on the topic "weather reports for London on 1 August 1999".*

Example 3.2 *Let θ_1 and θ_2 be two structured reports. Consider $\langle \texttt{city} : \texttt{London} \rangle$ is an atomic feature in θ_1 and $\langle \texttt{city} : \texttt{Paris} \rangle$ is an atomic feature in θ_2, and θ_1 and θ_2 are on the topic "weather reports for 1 August 1999".*

To capture this situation, we can describe some semantic labels as sensitive, such as `today` in Example 3.1, and others as insensitive, such as `city` in Example 3.2. To do this, we add axioms to the domain knowledge to formalize the conditions under which a conflict between text entries arises.

Definition 3.2 *Let* `incoherent` *and* `coherent` *be binary predicates in the language for the domain knowledge with the following being constraints on their behaviour. We assume the following axioms are included in the domain knowledge.*

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

We assume the variables `X` *and* `Y` *range over text entries.*

Definition 3.3 *In the language for the domain knowledge, the* **label conjunction** *function, denoted $+$, is a function symbol denoting conjunction of reference labels. So if* `X` *and* `Y` *are reference labels, then $\texttt{X} + \texttt{Y}$ is a reference label. The domain knowledge includes axioms for associativity, commutativity and idempotence for this function symbol.*

Definition 3.4 *The* **labels** *function, denoted Labels, is defined as follows: For a structured report θ, $Labels(\theta)$ is the set of reference labels in $Formulate(\theta)$. For a set of structured reports $\Theta = \{\theta_1, .., \theta_n\}$,*

$$Labels(\Theta) = \{r_1 + ... + r_n \mid r_1 \in Labels(\theta_1) \text{ and } .... \text{ and } r_n \in Labels(\theta_n)\}$$

We assume axioms are added to the domain knowledge such that if a pair of structured reports have a semantic label α with incoherent text entries, then an inconsistency follows. Adopting this assumption means that there is no classical model of a set of structured text where there are incoherent text entries.

Definition 3.5 *Let* `conflict` *be a monadic predicate in the language with the following axiom in the domain knowledge:*

$$\forall \texttt{X}\ \texttt{conflict}(\texttt{X}) \rightarrow \bot$$

We assume the variable `X` *ranges over conjunctions of reference labels.*

Example 3.3 *Let us assume we have the following domain knowledge for identifying pairwise* *inconsistency in text entries for sensitive semantic labels in weather reports:*

$$\forall \mathtt{A,B,C,D,X,Y}\quad \mathtt{date(X,D)} \wedge \mathtt{date(Y,D)} \wedge \mathtt{city(X,C)} \wedge \mathtt{city(Y,C)}$$
$$\wedge\, \mathtt{today(X,A)} \wedge \mathtt{today(Y,B)} \wedge \mathtt{incoherent(A,B)} \rightarrow \mathtt{conflict(X+Y)}$$

$$\forall \mathtt{A,B,C,D,X,Y}\quad \mathtt{date(X,D)} \wedge \mathtt{date(Y,D)} \wedge \mathtt{city(X,C)} \wedge \mathtt{city(Y,C)}$$
$$\wedge\, \mathtt{tomorrow(X,A)} \wedge \mathtt{tomorrow(Y,B)} \wedge \mathtt{incoherent(A,B)} \rightarrow \mathtt{conflict(X+Y)}$$

```
incoherent(sun,rain)
incoherent(cold,hot)
incoherent(cool,hot)
incoherent(cold,warm)
incoherent(cool,warm)
incoherent(sun,snow)
```

The conditions `date(X,D)`*,* `date(Y,D)`*,* `city(X,C)`*, and* `city(Y,C)` *ensure that structured reports with reference labels* `X` *and* `Y` *are on the same topic.*

Example 3.4 *Now to illustrate inconsistency in structured reports, we will consider the problem of the following two conflicting weather reports.*

```
⟨weather report:                 ⟨weather report:
      ⟨source: TV1⟩                    ⟨source: TV3⟩
      ⟨date: 19.5.1999⟩                ⟨date: 19.5.1999⟩
      ⟨city: London⟩                   ⟨city: London⟩
      ⟨today: sun⟩                     ⟨today: sun⟩
      ⟨tomorrow: sun⟩                  ⟨tomorrow: rain⟩
:weather report⟩                 :weather report⟩
```

The reports can be represented by the following two sets of formulae, where `r1` *is the reference label of the first report, and* `r2` *is the reference label of the second report.*

$$\{\mathtt{weatherreport(r1)}, \mathtt{source(r1,TV1)}, \mathtt{date(r1,19.5.1999)},$$
$$\mathtt{city(r1,London)}, \mathtt{today(r1,sun)}, \mathtt{tomorrow(r1,sun)}\}$$

$$\{\mathtt{weatherreport(r2)}, \mathtt{source(r2,TV3)}, \mathtt{date(r2,19.5.1999)},$$
$$\mathtt{city(r2,London)}, \mathtt{today(r2,sun)}, \mathtt{tomorrow(r2,rain)}\}$$

From this, we get `conflict(r1+r2)` *as an inference, and hence* $\perp$*.*

Definition 3.6 *Let* θ_1 *and* θ_2 *be structured reports. Let* `X` *be the reference label for* θ_1 *and let* `Y` *be the reference label for* θ_2*. There is a* **conflict** *in* Δ *between* θ_1 *and* θ_2 *with respect to semantic label* α *iff* $\exists$ `A,B` *such that* $Formulate(\theta_1) \cup Formulate(\theta_2) \cup \Delta$ *implies* $\alpha(\mathtt{X,A})$*,* $\alpha(\mathtt{Y,B})$*, and* `incoherent(A,B)`*.*

In order to determine whether a pair of structured reports represented as formulae are on the same topic, we need axioms to reason with formulae, such as in Example 3.5. These axioms are needed to identify equivalences between text entries including synonymous text entries. In practice, we would need axioms for text entries to semantic labels such as `name`, `date`, `location`, `source`, etc.

Example 3.5 *Consider the following axioms — we might assume axioms like this for every date in the domain knowledge.*

$$\forall \mathtt{X}\ \mathtt{date(X,12.01.99)} \leftrightarrow \mathtt{day(X,12th)} \wedge \mathtt{month(X,January)} \wedge \mathtt{year(X,1999)}$$

$$\forall \mathtt{X}\ \mathtt{date(X,12\ Jan\ 99)} \leftrightarrow \mathtt{day(X,12th)} \wedge \mathtt{month(X,January)} \wedge \mathtt{year(X,1999)}$$

Now consider the atomic features ⟨`date: 12.01.99`⟩ *and* ⟨`date: 12 Jan 1999`⟩. *Here,* `12.01.99` *and* `12 Jan 1999` *are not identical terms. However, we can infer from the above axioms that they are equivalent.*

So far we have focussed on inconsistency between pairs of atomic features. We can also have an inconsistency between two or more complex features. Consider the complex feature $\langle \alpha : \sigma_1, .., \sigma_n : \alpha\rangle$ in item X and the complex feature $\langle \alpha : \phi_1, .., \phi_n : \alpha\rangle$ in item Y, where $\{\sigma_1, .., \sigma_n\} \neq \{\phi_1, .., \phi_n\}$. For some semantic labels, this inequality would indicate an inconsistency.

Example 3.6 *(Complex feature inconsistency) Consider the following two complex features that have the same structure:*

```
⟨customer :                          ⟨customer :
      ⟨name : Peter Jones⟩                 ⟨name : Peter Jones⟩
      ⟨address :                           ⟨address :
            ⟨street : High Street⟩               ⟨street : Castle Street⟩
            ⟨city : Bath⟩                        ⟨city : Swindon⟩
      : address⟩                           : address⟩
: customer⟩                          : customer⟩
```

Here if we assume a form of "unique names" axiom for `name` *plus assume an integrity constraint that each individual only has one address, then we have an inconsistency between the* `address` *complex features in this pair of items.*

In this paper, most examples involve an inconsistency between a pair of items. However, an inconsistency may involve three or more items.

Example 3.7 *Suppose we have three weather reports θ_1, θ_2, and θ_3, and suppose θ_1 contains the atomic feature* ⟨`morning weather : rain`⟩, *θ_2 contains the atomic feature* ⟨`afternoon weather : rain`⟩, *and θ_3 contains the atomic feature* ⟨`today's weather : sunny periods`⟩. *Here it is straightforward to add axioms to the domain knowledge that would force an inconsistency between these reports.*

Now we have a way of determining how structured reports can be inconsistent with respect to domain knowledge, we can consider ways of merging sets of structured reports. This is the subject of the next section.

4 Merging structured reports

We now present a new framework for logic-based merging. Given a set of structured reports $\{\theta_1, ..., \theta_n\}$, we want to merge them into a single structured report θ. We call θ a merged report. If we have two or more conflicting reports, we can form a merged report by using strategies based on one or more of the following approaches, and which we will explain more fully in subsequent sections:

Combination of atomic features — suppose that $\phi_1, .., \phi_n$ are some of the atomic features in a set of structured reports, and furthermore, suppose that they all have the same semantic label, then we can merge them by taking either the "disjunction" or "conjunction" of the text entries in $\phi_1, .., \phi_n$ as the text entry in the atomic feature in the merged report.

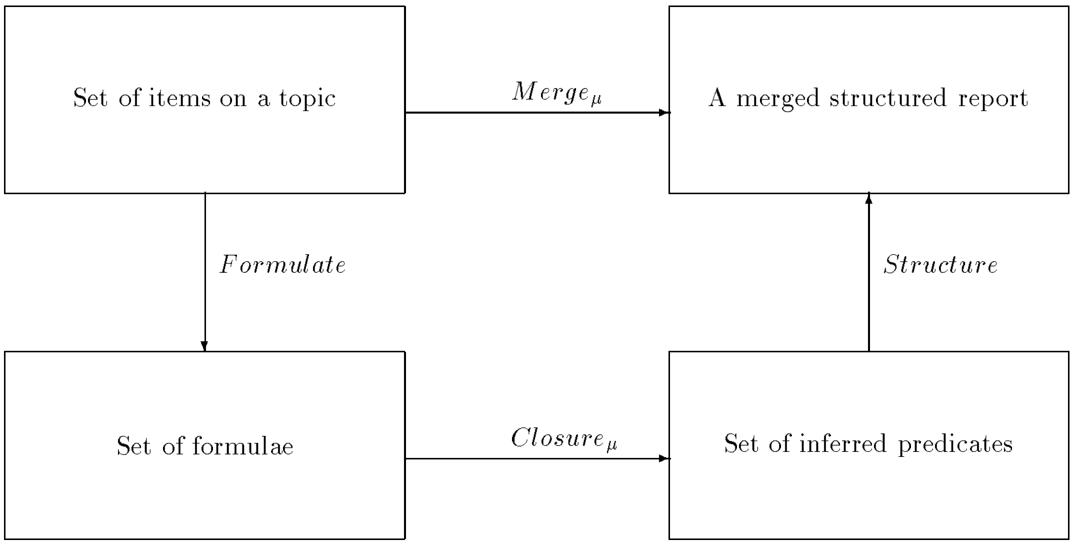

Figure 3: This figure summarizes the merging framework. $Merge_\mu$ is a function that takes a set of structured reports and returns a new structured report that is formed by merging. Rather than define $Merge_\mu$ directly, we define it indirectly in terms of $Formulate$, $Closure_\mu$, and $Structure$. $Formulate$ takes a set of structured reports and returns a set of formulae. $Closure_\mu$ takes a set of formulae and returns all inferred predicates that can be inferred using the domain knowledge and merging axioms in μ. $Structure$ takes a set of inferred predicates and forms a structured report.

Union of complex features — suppose that $\phi_1, .., \phi_n$ are some of the complex features in a set of structured reports, and furthermore, suppose all of $\phi_1, .., \phi_n$ have the same semantic label as root, then we can take the union of them by putting all of them in the merged report.

Selection of some atomic or complex features occuring in a set of structured reports, and rejecting the rest. Selecting atomic or complex features may involve precedence over sources (such as precedence over author, date, provider, or media) or precedence over content.

For combination, union, and selection, we need to adopt some merging axioms. Essentially, for a set of structured reports to be merged, we take the inferences that can be obtained from them by applying the merging axioms. Then from the inferences from the merging axioms, we form a new structured report. In this paper, we will restrict consideration to merging structured reports $\{\theta_1, .., \theta_n\}$ that have the same semantic label as the root, ie., $Root(\theta_1) = ... = Root(\theta_n)$.

Definition 4.1 *A set of* **merging axioms***, denoted Γ, is a set of classical formulae. We will add some constraints on the language for any Γ in this section. The choice of axioms in the set depends on the application. We also assume that the merging axioms can use the language we have introduced so far for the domain knowledge.*

We explain in the following definitions and examples the kinds of formulae that are included in a set of merging axioms.

Definition 4.2 *A* **merging system***, denoted μ, is a pair (Δ, Γ) where Δ is a set of domain knowledge and Γ is a set of merging axioms.*

Definition 4.3 *A* **closure** *function, denoted $Closure_\mu$, is defined as follows. Let Δ be the domain knowledge, let Γ be a set of merging axioms, let $\mu = (\Delta, \Gamma)$, let $\Theta = \{\theta_1, .., \theta_n\}$ be a set of structured reports, and let $\vdash$ be the classical logic consequence relation.*

$$\begin{aligned} Closure_\mu(\Theta) = \{\alpha(\beta,\gamma) \mid \Delta \cup \Gamma \cup Formulate(\Theta) \vdash \alpha(\beta,\gamma) \\ \cup \{\alpha(\beta) \mid \ \Delta \cup \Gamma \cup Formulate(\Theta) \vdash \alpha(\beta)\} \end{aligned}$$

Each formula in $Closure_\mu(\Theta)$ is an inferred predicate.

Definition 4.4 *Let Φ be a set of structured reports on some topic. A* **merging function** *$Merge_\mu$ returns a structured report such that*

$$Merge_\mu(\Phi) = Structure(Closure_\mu(\Phi))$$

where Structure takes a set of inferred predicates and forms a structured report. We define Structure formally in Definition 4.14.

In the following subsections, we explain combination, union, and selection in more detail, and give examples of merging axioms.

4.1 Combination of atomic features

The aim of combination of atomic features is to form atomic features with text entries that are disjunctive if incoherent and conjunctive if coherent.

Definition 4.5 *In the language for merging systems (i.e. the language for domain knowledge and merging axioms), $\oplus$ is a function symbol denoting conjunction of text entries, and $\otimes$ is a function symbol denoting disjunction of text entries. The domain knowledge includes axioms for associativity, commutativity and idempotence for both of these function symbols.*

Example 4.1 *Let the following be a set of merging axioms for combination.*

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

$$\texttt{tomorrow(X,A)} \land \texttt{tomorrow(Y,B)} \land \texttt{coherent(A,B)} \rightarrow \texttt{tomorrow(X} + \texttt{Y,A} \oplus \texttt{B)}$$

$$\texttt{today(X,A)} \land \texttt{today(Y,B)} \land \texttt{incoherent(A,B)} \rightarrow \texttt{today(X} + \texttt{Y,A} \otimes \texttt{B)}$$

$$\texttt{tomorrow(X,A)} \land \texttt{tomorrow(Y,B)} \land \texttt{incoherent(A,B)} \rightarrow \texttt{tomorrow(X} + \texttt{Y,A} \otimes \texttt{B)}$$

$$\texttt{source(X,A)} \land \texttt{source(Y,B)} \rightarrow \texttt{source(X} + \texttt{Y,A} \oplus \texttt{B)}$$

$$\texttt{date(X,D)} \land \texttt{date(Y,D)} \rightarrow \texttt{date(X} + \texttt{Y,D)}$$

$$\texttt{city(X,C)} \land \texttt{city(Y,C)} \rightarrow \texttt{city(X} + \texttt{Y,C)}$$

Example 4.2 *We return to Example 3.4. Using the merging axioms in Example 4.1, we get the following inferred predicates:*

source(r1+r2,TV1⊕TV3)
date(r1+r2,19.5.99)
city(r1+r2,London)
today(r1+r2,sun)
tomorrow(r1+r2,sun⊗rain)

It may also be desirable to include lexical rules in the merging axioms so that for example, if we get $\texttt{weather}(\texttt{r1}+\texttt{r2}, \texttt{sun} \oplus \texttt{sunny})$ as an inferred predicate, we can also get $\texttt{weather}(\texttt{r1}+\texttt{r2}, \texttt{sun})$ as an inferred predicate.

4.2 Union of complex features

We do not necessarily want to combine all complex features by considering the combination of the atomic features within them. Rather, in some cases, we may want to allow for the union of some complex features occurring in the set of structured reports to be merged. We can handle this by including all these complex features in the merged feature.

Example 4.3 *Consider the following two structured reports.*

```
⟨weather report :
      ⟨date : 5 Nov 99⟩
      ⟨regional report :
            ⟨region : South East⟩
            ⟨max temp : 20C⟩
      : regional report⟩
: weather report⟩
```

```
⟨weather report :
      ⟨date : 5 Nov 99⟩
      ⟨regional report :
            ⟨region : North West⟩
            ⟨max temp : 18C⟩
      : regional report⟩
: weather report⟩
```

Here we may wish to take the union of the two `regional report` *complex features in the merged feature, giving the following merged report:*

```
⟨weather report :
      ⟨date : 5Nov99⟩
      ⟨regional report :
            ⟨region : South East⟩
            ⟨max temp : 20C⟩
      : regional report⟩
      ⟨regional report :
            ⟨region : North West⟩
            ⟨max temp : 18C⟩
      : regional report⟩
: weather report⟩
```

To get this, we may assume the following merging axiom:

$$
\begin{array}{l}
\texttt{weatherreport(X1)} \land \texttt{weatherreport(Y1)} \\
\land\texttt{regionalreport(X1,X2)} \land \texttt{regionalreport(Y1,Y2)} \\
\land\texttt{region(X2,SouthEast)} \land \texttt{region(Y2,NorthWest)} \\
\land\texttt{maxtemp(X2,A2)} \land \texttt{maxtemp(Y2,B2)} \\
\rightarrow \\
\texttt{weatherreport(X1} + \texttt{Y1)} \\
\land\texttt{regionalreport(X1} + \texttt{Y1,X2)} \land \texttt{regionalreport(X1} + \texttt{Y1,Y2)} \\
\land\texttt{region(X2,SouthEast)} \land \texttt{region(Y2,NorthWest)} \\
\land\texttt{maxtemp(X2,A2)} \land \texttt{maxtemp(Y2,B2)}
\end{array}
$$

Suppose we have a set of structured reports $\theta_1, .., \theta_n$, and ϕ_1 is a complex feature in θ_1, and ..., and ϕ_n is a complex feature in θ_n. To be able to take the union of $\phi_1, ..\phi_n$ in the merged report, we would need to check that the parent of each ϕ_i in θ_i has the same semantic label. We capture this in the definition of the unionizable property below.

Definition 4.6 *Let $\{\theta_1, ..., \theta_n\}$ be a set of structured reports. If $\phi_1 \in Sub(\theta_1) \cup ... \cup Sub(\theta_n)$ and ... and $\phi_i \in Sub(\theta_1) \cup ... \cup Sub(\theta_n)$ and $Root(\phi_1) = ... = Root(\phi_i)$ then $\{\phi_1, ..., \phi_i\}$ are* **unionizable**.

So merging axioms for taking the union of some complex features will incorporate a condition that checks that the complex features are unionizable. We see this in Example 4.3.

4.3 Combination of nestable reports

Here we consider a generalization of combination given in Section 4.1 by defining a notion of nestable reports, and then extending the form of combination axioms. In particular, we are interested in merging reports on the same topic where some of the reports may be missing some features, such as in Example 4.4.

Definition 4.7 *Let $\{\theta_1, .., \theta_n\}$ be a set of structured reports. A skeleton S is a* **superstructure** *for $\{\theta_1, .., \theta_n\}$ iff S is the minimal skeleton in the $\preceq$ ordering such that*

$$Skeleton(\theta_1) \preceq S \text{ and ... and } Skeleton(\theta_n) \preceq S$$

Definition 4.8 *A set of items Θ is* **nestable** *iff there is a superstructure for Θ.*

Example 4.4 *For nestable items of structured text, we will consider the following two weather reports.*

```
⟨weather report:                 ⟨weather report:
     ⟨source: TV1⟩                     ⟨source: TV4⟩
     ⟨date: 19.5.1999⟩                 ⟨date: 19.5.1999⟩
     ⟨city: London⟩                    ⟨city: London⟩
     ⟨today: sun⟩                      ⟨today: sun⟩
     ⟨tomorrow: sun⟩               :weather report⟩
:weather report⟩
```

Suppose the left report is denoted by θ_1, and the right report is denoted by θ_2. Here, we have $Skeleton(\theta_2) \preceq Skeleton(\theta_1)$, but $Skeleton(\theta_1) \not\preceq Skeleton(\theta_2)$. Furthermore, we can take $Skeleton(\theta_1)$ as the superstructure for this pair of structured reports.

The reports can be represented by the following two sets of formulae, where `r1` *is the reference label of θ_1, and* `r9` *is the reference label of θ_2.*

$$\{\texttt{weatherreport(r1)}, \texttt{source(r1, TV1)}, \texttt{date(r1, 19.5.1999)}, \\ \texttt{city(r1, London)}, \texttt{today(r1, sun)}, \texttt{tomorrow(r1, sun)}\}$$

$$\{\texttt{weatherreport(r9)}, \texttt{source(r9, TV4)}, \texttt{date(r9, 19.5.1999)}, \\ \texttt{city(r9, London)}, \texttt{today(r9, sun)}\}$$

To handle nestable items, we add further axioms to the merging axioms. For a structured report θ, the set $Neg(\theta, S)$ is the set of literals corresponding to nodes on the skeleton S but not occurring in θ. Each formula in $Neg(\theta, S)$ is a negative literal where the predicate symbol is a semantic label in S but not in θ, and the argument is the reference label for the node if it were in θ. We define this more formally as follows.

Definition 4.9 *Let θ be a structured text. The set of missing nodes in θ with respect to S, denoted by $Neg(\theta, S)$, is formed as follows: Let $S(\theta)$ be a partial instantiation of S. If ϕ_n is a node in S that is not instantiated in $S(\theta)$, and the reference label of ϕ_n is p, then $\neg\phi_n(p)$ is in $Neg(\theta, S)$.*

Definition 4.10 *Let $\Theta = \{\theta_1, .., \theta_n\}$ be a set of nestable items, and let S be a superstructure for Θ. For each i such that $1 \leq i \leq n$, the set $Neg(\theta_i, S)$ is added to the merging axioms.*

Example 4.5 *Let the following be a set of merging axioms for combining nestable items. These can be added to the merging axioms in Example 4.1.*

$$\texttt{today}(\texttt{X}, \texttt{A}) \land \neg\texttt{today}(\texttt{Y}) \rightarrow \texttt{today}(\texttt{X} + \texttt{Y}, \texttt{A})$$

$$\texttt{tomorrow}(\texttt{X}, \texttt{A}) \land \neg\texttt{tomorrow}(\texttt{Y}) \rightarrow \texttt{tomorrow}(\texttt{X} + \texttt{Y}, \texttt{A})$$

So returning to Example 4.4, we get $\neg\texttt{tomorrow}(\texttt{r9})$, *and so we can infer* $\texttt{tomorrow}(\texttt{r1} + \texttt{r9}, \texttt{sun})$.

In this way, we have formalized a form of closed world assumption that can be used in a context-sensitive way to form a merged predicate in the case of missing information.

4.4 Selection of atomic features

The aim of selection is to form atomic features with text entries where the choice of text entry depends on the preferences over the original items. We represent preferences using an ordering relation.

Definition 4.11 *A* **selection ordering** *for a semantic label α is a pre-order relation $<_\alpha$ over the set of text entries. Each $<_\alpha$ is in the language for merging axioms.*

We give an illustration of a selection ordering and its use in the following example.

Example 4.6 *Suppose weather reports from different TV channels are given a ranking according to the reliability ordering* $\texttt{TV1} < \texttt{TV2} < \texttt{TV3}$, *where* $\texttt{TV1}$ *is least reliable, and* $\texttt{TV3}$ *is most reliable. To axiomatize this, we assume the merging axioms include the following formulae where $<_{source}$ is the selection ordering:*

$$\texttt{TV1} <_{source} \texttt{TV2} \land \texttt{TV1} <_{source} \texttt{TV3} \land \texttt{TV2} <_{source} \texttt{TV3}$$

$$\begin{gathered}\texttt{today}(\texttt{X}, \texttt{A}) \land \texttt{today}(\texttt{Y}, \texttt{B}) \land \texttt{date}(\texttt{X}, \texttt{D}) \land \texttt{date}(\texttt{Y}, \texttt{D}) \land \texttt{source}(\texttt{X}, \texttt{P}) \land \texttt{source}(\texttt{Y}, \texttt{Q}) \\ \land\texttt{incoherent}(\texttt{A}, \texttt{B}) \land \texttt{P} <_{source} \texttt{Q} \rightarrow \texttt{today}(\texttt{X} + \texttt{Y}, \texttt{B})\end{gathered}$$

$$\begin{gathered}\texttt{tomorrow}(\texttt{X}, \texttt{A}) \land \texttt{tomorrow}(\texttt{Y}, \texttt{B}) \land \texttt{date}(\texttt{X}, \texttt{D}) \land \texttt{date}(\texttt{Y}, \texttt{D}) \land \texttt{source}(\texttt{X}, \texttt{P}) \land \texttt{source}(\texttt{Y}, \texttt{Q}) \\ \land\texttt{incoherent}(\texttt{A}, \texttt{B}) \land \texttt{P} <_{source} \texttt{Q} \rightarrow \texttt{tomorrow}(\texttt{X} + \texttt{Y}, \texttt{B})\end{gathered}$$

$$\texttt{source}(\texttt{X}, \texttt{A}) \land \texttt{source}(\texttt{Y}, \texttt{B}) \rightarrow \texttt{source}(\texttt{X} + \texttt{Y}, \texttt{A} \oplus \texttt{B})$$

$$\texttt{date}(\texttt{X}, \texttt{D}) \land \texttt{date}(\texttt{Y}, \texttt{D}) \rightarrow \texttt{date}(\texttt{X} + \texttt{Y}, \texttt{D})$$

$$\texttt{city}(\texttt{X}, \texttt{C}) \land \texttt{city}(\texttt{Y}, \texttt{C}) \rightarrow \texttt{city}(\texttt{X} + \texttt{Y}, \texttt{C})$$

We give another illustration of a selection ordering in the following example.

Example 4.7 *Let* X *and* Y *be the reference labels for a pair of weather reports that are for the same city and same source but made on different days. Also suppose we prefer more recent weather reports. To do this, we add the following set of formulae to the merging axioms:*

$$\begin{array}{l}\texttt{today(X,A)} \land \texttt{today(Y,B)} \land \texttt{incoherent(A,B)} \\ \land\texttt{source(X,P)} \land \texttt{source(Y,Q)} \\ \land\texttt{P} \not<_{source} \texttt{Q} \land \texttt{Q} \not<_{source} \texttt{P} \\ \land\texttt{date(X,D)} \land \texttt{date(Y,D}') \land \texttt{D} <_{date} \texttt{D}' \\ \rightarrow \texttt{today(X+Y,B)}\end{array}$$

For this, we also assume that the following set of axioms is implicitly in the merging axioms:

$$\{\texttt{D} <_{date} \texttt{D}' \mid \texttt{D} \textit{ is an earlier date than } \texttt{D}'\}.$$

We also need to aggregate preferences as illustrated in the following example.

Example 4.8 *We extend the merging axioms given in Examples 4.6 and 4.7. These extra axioms aggregate the preferences, so that we prefer more recent reports from a less reliable TV station than less recent reports from a more reliable TV station.*

$$\begin{array}{l}\texttt{today(X,A)} \land \texttt{today(Y,B)} \land \texttt{source(X,P)} \land \texttt{source(Y,Q)} \land \texttt{Q} <_{source} \texttt{P} \\ \qquad \land\texttt{incoherent(A,B)} \land \texttt{date(X,D)} \land \texttt{date(Y,D}') \land \texttt{D} <_{date} \texttt{D}' \rightarrow \texttt{today(X+Y,B)}\end{array}$$

$$\begin{array}{l}\texttt{tomorrow(X,A)} \land \texttt{tomorrow(Y,B)} \land \texttt{source(X,P)} \land \texttt{source(Y,Q)} \land \texttt{Q} <_{source} \texttt{P} \\ \qquad \land\texttt{incoherent(A,B)} \land \texttt{date(X,D)} \land \texttt{date(Y,D}') \land \texttt{D} <_{date} \texttt{D}' \rightarrow \texttt{tomorrow(X+Y,B)}\end{array}$$

Example 4.9 *We return to Example 3.4. Using the merging axioms in Example 4.8, we get the following merged predicates:*

$$\begin{array}{c}\texttt{source(r1+r2,TV1}\oplus\texttt{TV3)} \\ \texttt{date(r1+r2,19.5.99)} \\ \texttt{city(r1+r2,London)} \\ \texttt{today(r1+r2,sun)} \\ \texttt{tomorrow(r1+r2,sun)}\end{array}$$

In this way, we use a multi-dimensional preference ordering over the atomic features. In other words, rather than just assuming an ordering over the structured reports, we adopt finer-grained orderings that can be at the level of atomic features.

4.5 Formation of merged structured report

Using the merging axioms, whether combination, union, or selection, we obtain a set of inferred predicates. We now consider constructing a new structured report that incorporates the information in these inferred predicates. Hence, we form an item of structured text from the original set of items of structured text. First, we require a couple of subsidiary definitions.

Definition 4.12 *A* **rewrite rule** *is a formula* $\forall X_1,..,X_n(\alpha \rightarrow \beta)$ *where* α *and* β *are unquantified positive literals that have the same predicate symbol.*

For merging systems, the rewrite rules are formulae that take inferred predicates and rewrite the text entry — either simplifying the text entry or replacing the text entry with a synonym. In other words, using universal instantiation and modus ponens, with a merged predicate and a rewrite rule, gives a new inferred predicate.

Example 4.10 *The following merging axioms are examples of rewrite rules.*

$$\forall \mathtt{X}, \mathtt{Y}, \mathtt{Z}\ \mathtt{weather}(\mathtt{X}, (\mathtt{Y} \oplus \mathtt{Y}) \oplus \mathtt{Z}) \rightarrow \mathtt{weather}(\mathtt{X}, \mathtt{Y} \oplus \mathtt{Z})$$

$$\forall \mathtt{X}\ \mathtt{weather}(\mathtt{X}, \mathtt{sun} \otimes \mathtt{sunny}) \rightarrow \mathtt{weather}(\mathtt{X}, \mathtt{sunny})$$

Definition 4.13 *Let $Closure_\mu$ be a merging function and Θ be a set of structured reports. For all $\phi, \psi \in Closure_\mu(\Theta)$, if $\forall X_1, .., X_n(\alpha \rightarrow \beta)$ is a rewrite rule in the merging axioms in μ, and there is a grounding (instantiation) of the rewrite rule of the form $\phi \rightarrow \psi$, then ψ is* **more preferred** *than ϕ in $Closure_\mu(\Theta)$.*

Example 4.11 *Continuing Example 4.10, if we have* $\mathtt{weather}(\mathtt{r}, \mathtt{sun} \otimes \mathtt{sunny})$ *and* $\mathtt{weather}(\mathtt{r}, \mathtt{sunny})$, *in $Closure_\mu(\Theta)$ then* $\mathtt{weather}(\mathtt{r}, \mathtt{sunny})$ *is preferred to* $\mathtt{weather}(\mathtt{r}, \mathtt{sun} \otimes \mathtt{sunny})$.

Using preference over inferred predicates, we can filter out the less preferred predicates when we form the merged structured report.

Definition 4.14 *The* **structure** *function, denoted Structure, is defined as follows: Let $Closure_\mu(\Theta)$ be a set of inferred predicates. Let $\langle \phi : \sigma_1, .., \sigma_n : \phi \rangle$ be a structured report that we will form from the logical formulae in $Closure_\mu(\Theta)$. If $\Theta = \{\theta_1, .., \theta_n\}$, and r_1 is the reference label of θ_n, and ... and r_n is the reference label of θ_n, then $r_1 + ... + r_n$ is the reference label of the merged report and also the reference of the root of the merged report. The structured report is obtained by exhaustively applying the following rules:*

1. *If $X(Y, Z) \in Closure_\mu(\Theta)$, and X is a semantic label, and $Y \in Labels(\Theta)$, and Z is not a text entry, then Y is the reference label for the parent of X in the merged report, and Z is the reference label for X in the merged report.*
2. *If $X(Y, Z) \in Closure_\mu(\Theta)$, and X is a semantic label, and Y is a reference label for a semantic label in the merged report, then Y is the reference label for the parent of X in the merged report, and Z is the reference label for X in the merged report.*
3. *If $X(Y, Z) \in Closure_\mu(\Theta)$, and there is no literal in $Closure_\mu(\Theta)$ that is more preferred than $X(Y, Z)$, and X is a semantic label, and Y is the reference label for the parent of X in the merged report, then Z is the text entry for X in the merged report.*

We explain these rules as follows: Rule 1 identifies the semantic labels that should be included in the merged report going from the root downwards — these are identified via the reference labels in $Labels(\Theta)$; Rule 2 identifies any branches that are to be included in the merged report as a result of the union of complex features; and Rule 3 identifies the most preferred text entries to include.

The structure function is, in a sense, the reverse process of the formulate function. It is defined so that $Structure(Formulae(\theta)) = \theta$ holds.

Example 4.12 *Continuing Example 4.2, we form the following merged item of structured text using the structure function on the inferred predicates.*

```
⟨weather report:
     ⟨source: TV1⊕TV3⟩
     ⟨date: 19.5.1999⟩
     ⟨city: London⟩
     ⟨today: sun⟩
     ⟨tomorrow: sun⊗rain⟩
:weather report⟩
```

Example 4.13 *Continuing Example 4.9, we form the following merged item of structured text using the structure function on the inferred predicates.*

```
⟨weather report:
     ⟨source: TV1⊕TV3⟩
     ⟨date: 19.5.1999⟩
     ⟨city: London⟩
     ⟨today: sun⟩
     ⟨tomorrow: sun⟩
:weather report⟩
```

Before we consider a bigger example (Example 4.14), we will provide a further definition for the notation used in a merging system. We will illustrate the use of this definition in the bigger example.

Definition 4.15 *In the language for merging systems (i.e. the language for domain knowledge and merging axioms), the* **range function***, denoted* $-$*, is a function symbol where* $A - B$ *denotes a range of values from the value given in the text entry* A *to the value given in the text entry* B*.*

Example 4.14 *Now to illustrate merging on a bigger example, we will consider the problem of the following two conflicting weather reports.*

```
⟨weather report:                             ⟨weather report:
     ⟨source: TV1⟩                                ⟨source: TV3⟩
     ⟨date: 1.8.1999⟩                             ⟨date: 1.8.1999⟩
     ⟨city: Paris⟩                                ⟨city: Paris⟩
     ⟨midday weather:                             ⟨midday weather:
          ⟨precipitation: inclement⟩                   ⟨precipitation: showers⟩
          ⟨temperature: 20C⟩                           ⟨temperature: 18C⟩
     midday weather⟩                              midday weather⟩
:weather report⟩                             :weather report⟩
```

From this we get the following set of predicates:

```
     weatherreport(r1)             weatherreport(r2)
       source(r1,TV1)                source(r2,TV3)
      date(r1,1.8.99)               date(r2,1.8.99)
      city(r1,Paris)                city(r2,Paris)
   middayweather(r1,r1′)         middayweather(r2,r2′)
precipitation(r1′,inclement) precipitation(r2′,showers)
   temperature(r1′,20C)          temperature(r2′,18C)
```

We add the following axiom to the merging axioms given in Examples 4.6, 4.7, and 4.8.

$$\texttt{temperature(X,A)} \wedge \texttt{temperature(Y,B)} \wedge \texttt{A} < \texttt{B} \wedge \texttt{coherent(A,B)} \rightarrow \texttt{temperature(X+Y, A-B)}$$

We also assume facts such as `coherent(18C,20C)` *and* `incoherent(10C,30C)` *are added to the merging axioms. Note, if we are to adopt a logic-based approach, and we do not want to use many-valued logic or fuzzy logic, then we need to allow statements such as* `coherent(18C,20C)` *and* `incoherent(10C,30C)` *to hold. This approach would also necessitate us to commit to statements such as* `coherent(19C,21C)` *and* `incoherent(11C,30C)`*, or maybe we could assume the following more general rule.*

$$(\texttt{0} < \texttt{A}) \wedge (\texttt{A} \leq \texttt{B}) \wedge ((\texttt{B} - \texttt{A}) < \texttt{10}) \leftrightarrow \texttt{coherent(A,B)}$$

Any such knowledge is context-sensitive, and very much depends on who is using the merging system. Different users may choose to adopt different knowledge.

Using this extended set of merging axioms gives the following set of inferred predicates:

```
weatherreport(r1+r2)
source(r1+r2,TV1⊕TV3)
date(r1+r2,1.8.99)
city(r1+r2,Paris)
middayweather(r1+r2,r1′+r2′)
precipitation(r1′+r2′, inclement⊕showers)
temperature(r1′+r2′,18C-20C)
```

This gives the following merged report:

```
⟨weather report:
     ⟨source: TV1⊕TV3⟩
     ⟨date: 1.8.1999⟩
     ⟨city: Paris⟩
     ⟨midday weather:
          ⟨precipitation: inclement⊕showers⟩
          ⟨temperature: 18C-20C⟩
     :midday weather⟩
:weather report⟩
```

For this example, we introduced the range function. There are many other functions and associated axioms that we may choose to introduce in the language of merging systems depending on the application area.

5 Analysing merging systems

Our approach to merging is based on application-dependent axioms. This means the $Merge_\mu$ function depends on the formulae in μ and hence depends on the application. However, we can make some general comments about the desirable behaviour of any $Merge_\mu$ function in an application.

Definition 5.1 *The* **reflexive**, **commutative**, *and* **associative** *properties are defined as follows: Let $\{\theta, \theta_1, \theta_2, \theta_3\}$ be a set of structured reports and let $Merge_\mu$ be a merge function.*

$$Merge_\mu(\{\theta, ..., \theta\}) = \theta \qquad (Reflexive)$$

$$Merge_\mu(\{\theta_1, \theta_2\}) = Merge_\mu(\{\theta_2, \theta_1\}) \qquad (Commutative)$$

$$Merge_\mu(\{Merge_\mu(\{\theta_1, \theta_2\}), \theta_3\}) = Merge_\mu(\{\theta_1, Merge_\mu(\{\theta_2, \theta_3\})\}) \qquad (Associative)$$

Normally a $Merge_{\mu}$ relation should be reflexive and commutative, though not associative. All the properties in Definition 5.1 hold if we assume μ contains a set of merging axioms that has axioms of the following form for every semantic label α:

$$\alpha(X1, X2) \wedge \alpha(Y1, Y2) \rightarrow \alpha(X1 + Y1, X2 + Y2)$$

$$\alpha(X1, A) \wedge \alpha(Y1, B) \rightarrow \alpha(X1 + Y1, A \oplus B)$$

An extreme example of a $Merge_{\mu}$ function where all the properties in Definition 5.1 hold is when the set of merging axioms is inconsistent. In contrast, an extreme example of a $Merge_{\mu}$ function where reflexivity fails is when the set of merging axioms is the empty set.

Further properties that should hold of a $Merge_{\mu}$ function include the transitive and left logical equivalence properties.

Definition 5.2 *The* **transitive** *property holds for $Merge_{\mu}$ under the following condition where $\{\theta_1, ..., \theta_5\}$ is a set of structured reports.*

$Merge_{\mu}(\{\theta_1, \theta_2\}) = \theta_3$ *and* $Merge_{\mu}(\{\theta_3, \theta_4\}) = \theta_5$ *implies* $Merge_{\mu}(\{Merge_{\mu}(\{\theta_1, \theta_2\}), \theta_4\}) = \theta_5$

Definition 5.3 *The* **left logical equivalence** *property holds for $Merge_{\mu}$ under the following condition where $\{\theta_1, \theta_2\}$ is a set of structured reports.*

$$Formulate(\theta_1) = Formulate(\theta_2) \text{ implies } Merge_{\mu}(\theta_1) = Merge_{\mu}(\theta_2)$$

Further properties that may hold of a $Merge_{\mu}$ function include the monotonic, democratic, and dictatorial properties.

Definition 5.4 *The* **monotonic** *property holds for $Merge_{\mu}$ under the following condition where $\{\theta_1, .., \theta_n, \theta_{n+1}\}$ is a set of structured reports.*

$$Formulate(Merge_{\mu}(\{\theta_1, ..., \theta_n\})) \subseteq Formulate(Merge_{\mu}(\{\theta_1, ..., \theta_n, \theta_{n+1}\}))$$

Definition 5.5 *The* **democratic** *property is defined as follows: Let $\{\theta_1, ..., \theta_n\}$ be a set of structured reports and let $Merge_{\mu}$ be a merge function. There is an n such that if α is a complex feature in θ_1, and ....., and α is a complex feature in θ_n, and $Merge_{\mu}(\{\theta_1, ..., \theta_n\} = \psi$, then α is a complex feature in ψ. In other words, if there are enough items that contain α, then α is in the merged report.*

Definition 5.6 *The* **dictatorial** *property is defined as follows: Let Θ be a set of structured reports and let $Merge_{\mu}$ be a merge function. For all Θ, if $\phi \in \Theta$, then $Merge_{\mu}(\Theta) = \phi$.*

However, normally it is not possible for a $Merge_{\mu}$ function to adhere to the monotonic, democratic, or dictatorial properties.

Example 5.1 *Consider $Merge_1$ defined by the merging axioms in Example 4.1. Here $Merge_1$ is reflexive and commutative, but not associative, monotonic, democratic, or dictatorial.*

Example 5.2 *Similarly consider $Merge_2$ defined by the union of the merging axioms in Examples 4.1, 4.6, 4.7 and 4.8. Here $Merge_2$ is reflexive and commutative, but not associative, monotonic, democractic, or dictatorial.*

Further axioms that hold for some merge functions can be obtained by adapting some of the axioms in social choice theory (for a review see for example [Kel88]).

6 Comparison with related techniques

In this paper, we have examined the problem of merging items of structured text. We have seen how structured reports can be viewed as logical formulae, and looked at how we can conceptualize inconsistencies between structured reports and domain knowledge. To merge structured reports, we have considered various kinds of merging by using explicit sets of merging axioms. This now raises the question of the relationships with related logic-based techniques for fusion. For a review of information fusion in logic, see [CH97].

6.1 Overview of techniques for fusion in logic

Formalization of operators for merging information is a topic that impinges upon a number of areas in data and knowledge engineering research. In the following, we briefly review belief revision, database updating, knowledgebase merging and logical inferencing with inconsistent information.

6.1.1 Belief revision theory

In belief revision theory it is assumed that beliefs can be represented by logical formulae, and that operations on beliefs, such as revising beliefs in the light of new information, adhere to rational logical postulates. The initial proposals for postulates for adding, deleting, and revising beliefs were given by Alchurron, Gardenfors, and Makinson [AGM85, Gar88]. There have been a series of developments of belief revision theory including iterated belief revision [DP97, Leh95], and relating belief revision to database updating [KM91]. For a review of belief revision theory see [DP98].

These proposals for belief revision theory offer interesting intuitive abstract constraints for revision. However, there are a number of problems with regard to applicability. We will just focus on the "primacy of updates". This is the assumption that the result of updating a set of beliefs Φ by a new formula α should result in a revised set of beliefs that includes α. This is a central tenet in proposals for belief revision theory, and yet it seems counter to the needs of merging structured reports.

There are some more concrete proposals for knowledgebase merging that adhere to belief revision postulates. In Konieczny and Pino Perez [KP98], there is a proposal for merging beliefs based on semantically characterizing interpretations which are "closest" to the set of formulae to be updated. The approach has been generalized by considering merging with respect to integrity constraints [KP99]. These proposals also maintain the primacy of updates.

Another approach that extends belief revision theory, called arbitration operators, is by Liberatore and Schaerf [LS98]. This form of merging is restricted to merging only two knowledgebases and it forces the result to be the disjunction of the two original knowledgebases. Also it does not use any meta-level information such as preferences.

Less abstract proposals for belief revision that do incorporate priorities include ordered theory presentations [Rya92] and prioritized revision [GR96, GR97]. These approaches aim to keep as much of the original beliefs by adopting a more sophisticated view of refining beliefs. In ordered theory presentations, if a formula is less preferred than another which contradicts it, those aspects of it which are not contradicted are preserved. This is done by adopting an inferentially weaker formula to avoid the contradiction with the more preferred formula. This merging can be undertaken in an arbitrarily large partial ordering of formulae. In prioritized revision, a belief revision operator is defined in terms of selecting the model that satisfies the new belief and is nearest to the existing beliefs. The measure of nearness can be used in iterated belief revision where the more preferred items are used in later revisions. The key criticism with these approaches is the assumption that all the sources can be partially ordered without consideration of context. This does not seem realistic in practice.

6.1.2 Knowledgebase merging techniques

In addition to the techniques of Konieczny and Pino Perez [KP98, KP99] that we discussed in the previous subsection, we need to consider knowledgebase merging, by Baral et al [BKMS92] which offers two approaches to combining prioritized theories. These two approaches assume a linear ordering over a set of sets of formulae. The first technique is top-down merging and the second is bottom-up merging.

In the top-down version of merging, the most preferred set Δ_1 is taken and combined with the next most preferred set Δ_2 by taking the maximal subset of Δ_2 that is consistent with Δ_1. This technique is then iterated by taking the result from the previous cycle as the most preferred set, and Δ_3 is the next most preferred set. This process is repeated until all sets have been considered.

In the bottom-up version of combination, the least preferred set Δ_n is taken and combined with the next least preferred set Δ_{n-1} by taking the maximal subset of Δ_n that is consistent with Δ_{n-1}. This technique is then iterated by taking the result from the previous cycle as the least preferred set, and Δ_{n-2} is the next least preferred set. This process is repeated until all sets have been considered.

The key criticism with these approaches is the assumption that all the sources can be ordered without consideration of context. This does not seem realistic in practice.

6.1.3 Logical inferencing with inconsistent information

So far we have focussed on merging or revising a knowledgebase by rejecting items that cause inconsistency. A closely related approach is to keep all formulae, but to reason with maximally consistent subsets of the knowledgebase (see for example [MR70, BDP93, EGH95]). One of the problems with reasoning with maximally consistent subsets is that there may be many of them. This can be ameliorated by assuming meta-level information such as priorities over formulae. Examples of this kind of reasoning include [Bre89, CRS93, BDP95]. The key criticism with these approaches is the assumption that all the sources can be partially ordered without consideration of context.

Our review here of logical inferencing with inconsistent information has focussed on approaches closely related to knowledgebase merging. However, there are a number of other significant proposals for logical inferencing with inconsistent information (for a review see [Hun98]).

6.1.4 Summary of inadequacies of existing techniques

We regard these other merging proposals as too simplistic for the needs of merging structured reports. They do not incorporate enough consideration of more detailed issues of representation of the application domain. These problems can be summarized as follows:

One-dimensional preference orderings over sources If there is a preference ordering over the sources, then it is a one-dimensional ordering. Yet, for merging items of structured text, we need finer-grained preference over sources. These need to be context-sensitive and they need to be at the level of atomic features.

Primacy of update In belief revision theory, most approaches assume that the latest update takes precedence over the previously acquired information. In some situations, this is a good heuristic, but it is definitely not appropriate in all cases of merging items of structured text.

Separation of domain knowledge Most approaches do not incorporate a special role for domain knowledge — though in some approaches domain knowledge can be equated with integrity constraints.

Weak merging based on a meet operator This form of merging causes an unnecessary loss of useful information. In restricted domains it seems viable to acquire sufficient merging rules to give much more intelligent merging where far less information needs to be lost.

The net conclusion we draw from this analysis of these other proposals is that we require the more sophisticated context-dependent approaches for merging information such as proposed in this paper. We hope that the examples of merging weather reports given in this paper motivate this viewpoint.

6.2 Addressing the need for knowledge

Since we advocate the need for more sophisticated context-dependent approaches for merging information, we need to consider the issue of obtaining and managing the necessary knowledge. As can be seen in this paper, even seemingly simple examples of merging call for a substantial knowledge engineering effort. However, there are a number of research areas that impinge upon this problem, and that offer resources and techniques that could be harnessed in a solution.

Computational linguistics A lot of information for reasoning with structured text could be described as being lexical knowledge. Various very large lexical resources such as WordNet [Mil95] and various kinds of machine readable dictionaries and thesauri [WSG96] address part of this need. But richer formalisms for lexical semantics are also called for. This is an increasingly important area in computational linguistics (see for example [Pus95, MR99]).

Knowlegdebased systems Developing robust large-scale knowledgebases that contain common-sense knowledge and/or general knowledge has been the subject of much research within the knowlege representation and reasoning community. Probably one of the largest and most well-known is CYC which aimed to axiomatize a broad range of common-sense knowledge for use in diverse intelligent systems [Len95]. In addition, a number of knowledge engineering projects have produced tools and formalizations that could be appropriate in this application area (for example CommonKADS [Sea99]).

Inductive logic programming There are number of techniques in machine learning and text mining that could be applicable in developing knowledgebases for focussed example-rich structured text applications. Inductive logic programming is a particularly promising approach that generates logic theories from examples [Mug92].

Clearly, any practical merging system will involve a challenging integration of technologies and resources for constructing and managing the necessary knowledge.

7 Discussion

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 reasoning [Hun00b] and defeasible reasoning [Hun00a], could become a valuable technology. In addition, context-sensitive lexical and world knowledge can be used for reasoning with semantic labels to better answer queries [Hun96, Hun97, Hun99].

Whilst structured text is closely related to XML and associated ideas in semi-structured data, we leave a formal comparison to future work given that XML and related technologies are in such a state of flux [BDF+98, Abi99]. However, we can briefly consider the notion of wrappers, (see for example [HGNY97, Coh98, SA99]), which are of particular relevance to this paper. The approach of wrappers offers a practical level a way of defining how heterogenous information can be merged. However, there is little consideration of problems of conflicts arising between sources. Our approach therefore goes beyond these in terms of formalizing merging with inconsistent information.

References

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

[Abi99] S Abiteboul. On views and XML. In *Proceedings of the ACM SIGMOD/SIGART Conference on Principles of Database Systems (PODS'99)*, 1999.

[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.

[BDF+98] P Buneman, A Deutsch, W Fan, H Liefke, A Sahuguet, and W-C Tan. Beyond XML query languages. In *Proceedings of the Query Language Workshop (QL'98)*, 1998.

[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 *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, volume 956 of *Lecture Notes in Computer Science*, pages 36–43. Springer, 1995.

[BKMS92] C Baral, S Kraus, J Minker, and V Subrahmanian. Combining knowledgebases of consisting 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 *Proceedings 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.

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

[Coh98] W Cohen. A web-based information system that reasons with structured collections of text. In *Proceedings of Autonomous Agents'98*, 1998.

[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.

[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.

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

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

[GR96] D Gabbay and O Rodrigues. A methodology for iterated theory change. In *Practical Reasoning*, volume 1085 of *Lecture Notes in Computer Science*. Springer, 1996.

[GR97] D Gabbay and O Rodrigues. Structured belief bases: A practical approach to prioritized base revision. In *Qualitative and Quantitative Practical Reasoning*, volume 1244 of *Lecture Notes in Computer Science*. Springer, 1997.

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

[HGNY97] J Hammer, H Garcia-Molina, S Nestorov, and R Yerneni. Template-based wrappers in the TSIMMIS system. In *Proceedings of ACM SIGMOD'97*. ACM, 1997.

[Hun96] A Hunter. Intelligent text handling using default logic. In *Proceedings of the IEEE Conference on Tools with Artificial Intelligence (TAI'96)*, pages 34–40. IEEE Computer Society Press, 1996.

[Hun97] A. Hunter. Using default logic for lexical knowledge. In *Qualitative and Quantitative Practical Reasoning*, volume 1244 of *Lecture Notes in Computer Science*, pages 86–95. Springer, 1997.

[Hun98] A Hunter. Paraconsistent logics. In D Gabbay and Ph Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 2, pages 11–36. Kluwer, 1998.

[Hun99] A Hunter. Context-dependent reasoning with lexical knowledge. Technical report, Department of Computer Science, University College London, 1999.

[Hun00a] A Hunter. Ramification analysis using causal mapping. *Data and Knowledge Engineering*, 32:1–27, 2000.

[Hun00b] A Hunter. Reasoning with inconsistency in structured text. *Knowledge Engineering Review*, 2000. (in press).

[Kel88] J Kelly. *Social Choice Theory: An Introduction.* Springer, 1988.

[KM91] H Katsuno and A Mendelzon. On the difference between updating a knowledgebase and revising it. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference (KR'91)*, pages 387–394. Morgan Kaufmann, 1991.

[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 (KR'98)*, pages 488–498. Morgan Kaufmann, 1998.

[KP99] S Konieczny and R Pino Perez. Merging with integrity constraints. In *Qualitative and Quantitative Approaches to Reasoning and Uncertainty*, volume 1638 of *Lecture Notes in Computer Science.* Springer, 1999.

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

[Len95] D Lenat. CYC: a large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38:33–38, 1995.

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

[Mil95] G Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

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

[MR99] C Monz and M De Rijke, editors. *Proceedings of the First International Workshop on Inference in Computational Semantics.* Institute for Logic, Language and Computation, Amsterdam, 1999.

[Mug92] S Muggleton. *Inductive Logic Programming.* Academic Press, 1992.

[Pus95] J Pustejovsky. *The Generative Lexicon.* MIT Press, 1995.

[Rya92] M Ryan. Representing defaults as sentences with reduced priority. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92).* Morgan Kaufmann, 1992.

[SA99] A Sahuguet and F Azavant. Building light-weight wrappers for legacy web data-sources using W4F. In *Proceedings of the International Conference on Very Large Databases (VLDB'99)*, 1999.

[Sea99] G Schreiber and et al. *Knowledge Engineering and Management.* MIT Press, 1999.

[WSG96] Y Wilks, B Slator, and L Guthrie. *Electric Words: Dictionaries, Computers, and Meanings.* MIT Press, 1996.