NewsFusion Systems: Logic-based merging of heterogeneous news reports

Anthony Hunter and Rupert Summerton
Department of Computer Science
University College London
Gower Street, London WC1E 6BT, UK

May 28, 2003

Abstract

A NewsFusion System is a logic-based system for merging heterogeneous structured news reports. Structured news reports are XML documents, where the text entries are restricted to individual words or simple phrases, such as names and domain-specific terminology, and numbers and units. We assume structured news reports do not require natural language processing. In previous papers [Hun00, Hun02b, Hun02a, HS03b, HS03a], we have presented a logic-based framework for merging structured news reports based on fusion rules. Fusion rules are a form of scripting language that define how structured news reports should be merged. The antecedent of a fusion rule is a call to investigate the information in the structured news reports and the background knowledge, and the consequent of a fusion rule is a formula specifying an action to be undertaken to form a merged report. It is expected that a set of fusion rules is defined for any given application. In this paper, we extend the syntax for fusion rules to support variables in queries to the background knowledge in Prolog, and we outline our implementation for executing fusion rules, together with a case study on merging weather reports.

1 Introduction

Syntactically, a structured news report is a data structure containing a number of grammatically simple phrases together with a tag (giving semantic information) for each phrase. Each phrase that is tagged is a textentry. The set of tags in a structured news report is meant to parameterize a stereotypical situation, and so a particular structured news report is an instance of that stereotypical situation. For example, news reports on corporate acquisitions can be represented as structured news reports using tags including `buyer`, `seller`, `acquisition`, `value`, and `date`. Each phrase in structured news report 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 structured news report.

In order to merge structured news reports, we need to take account of the contents of the structured news reports. Different kinds of content need to be merged in different ways as illustrated below.

Example 1.1 *Consider the following two conflicting weather reports which are for the same day and same city. There are many further examples we could consider, each with particular features that indicate how the merged report should be formed.*

```
⟨weatherreport⟩
        ⟨source⟩ TV1 ⟨/source⟩
        ⟨date⟩ 19/3/02 ⟨/date⟩
        ⟨city⟩ London ⟨/city⟩
        ⟨today⟩ showers ⟨/today⟩
        ⟨temp⟩
                ⟨max⟩ 9C ⟨/max⟩
                ⟨min⟩ 3C ⟨/min⟩
        ⟨/temp⟩
        ⟨tomorrow⟩ sun ⟨/tomorrow⟩
        ⟨dewpoint⟩ 2C ⟨/dewpoint⟩
⟨/weatherreport⟩
```

```
⟨weatherreport⟩
        ⟨source⟩ TV3 ⟨/source⟩
        ⟨date⟩ 19 March 2002 ⟨/date⟩
        ⟨city⟩ London ⟨/city⟩
        ⟨today⟩ inclement ⟨/today⟩
        ⟨temp⟩
                ⟨max⟩ 45F ⟨/max⟩
                ⟨min⟩ 36F ⟨/min⟩
        ⟨/temp⟩
        ⟨tomorrow⟩ rain ⟨/tomorrow⟩
⟨/weatherreport⟩
```

We can merge them so the `source` *is* `TV1 and TV3`*, and the weather for* `today` *is* `showers and inclement`*, and the weather for* `tomorrow` *is* `sun or rain`*.*

```
⟨weatherreport⟩
        ⟨source⟩ TV1 and TV3 ⟨/source⟩
        ⟨date⟩ 19.03.02 ⟨/date⟩
        ⟨city⟩ London ⟨/city⟩
        ⟨today⟩ showers and inclement ⟨/today⟩
        ⟨temp⟩
                ⟨max⟩ 7 to 9C ⟨/max⟩
                ⟨min⟩ 2 to 3C ⟨/min⟩
        ⟨/temp⟩
        ⟨tomorrow⟩ sun or rain ⟨/tomorrow⟩
        ⟨dewpoint⟩ 2C ⟨/dewpoint⟩
⟨/weatherreport⟩
```

An alternative way of merging these reports may be possible if we have a preference for one source over the other. Suppose we have a preference for `TV3` *in the case of conflict, then we may prefer the text entry* `rain` *for the semantic label* `tomorrow`*. We consider various further aggregation functions for handling conflicting information in Section 4.*

In our approach to merging structured news reports, we draw on domain knowledge to help produce merged reports. The approach is based on fusion rules defined in an XML file[1] (using FusionRuleML[2]). These rules are of the form $\alpha \Rightarrow \beta$, where if α is true in the knowledgebase, then β is an instruction that needs to be undertaken in the process of building a merged news report.

To merge a set of structured news reports, we start with the background knowledge and the information in the news reports to be merged, and apply the fusion rules to this information. For a set of structured news reports and a set of fusion rules, we attempt to ground each fusion rule with textentries from the structured news reports, and then check whether all the conditions of each ground fusion rule are implied by the background knowledge, and if they are, then the ground actions of the rule are added to the actionlist (a list of actions that specify how the merged report should be constructed).

A NewsFusion System is a system for merging structured news reports for a particular domain. It incorporates some modules for executing fusion rules and for executing the resulting actions. It also incorporates a set of fusion rules that has been defined for the application domain together with an appropriate background knowledgebase. The basic architecture for a NewsFusion System is given in Figure 1.

[1]Examples of fusion rules together with XML representation are available at www.cs.ucl.ac.uk/research/newstools

[2]See www.cs.ucl.ac.uk/staff/a.hunter/newstools/frt/frml.dtd for the FusionRuleML DTD.

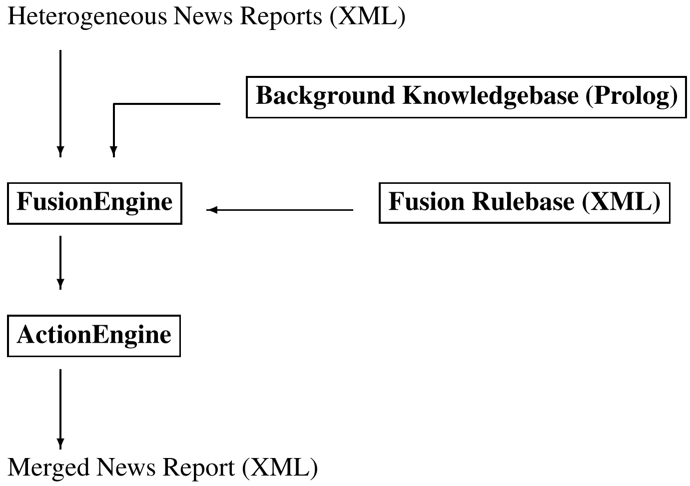

Figure 1: The basic architecture for a NewsFusion System. The two key modules are a FusionEngine and an ActionEngine implemented in Java. The fusion engine executes a rulefile (an XML file containing the fusion rules marked up in FusionRuleML), by grounding the fusion rules with textentries from the structured news reports, and then checks whether all the conditions of each ground fusion rule are implied by the background knowledge and, if they are, then the ground actions of the rule are added to the actionlist (a list of actions that specify how the merged report should be constructed). The ActionEngine executes the actionlist to build a merged report

2 Structured news reports

We use XML to represent structured news reports. So each structured news report is an XML document, but not vice versa, as defined below.

Definition 2.1 *If ϕ is a tagname (i.e. an element name), and ψ is textentry (i.e. a phrase represented by a string), then $\langle\phi\rangle\psi\langle/\phi\rangle$ is a structured news report. If ϕ is an tagname and $\sigma_1, ..., \sigma_n$ are structured news reports, then $\langle\phi\rangle\sigma_1...\sigma_n\langle/\phi\rangle$ is a structured news report.*

The definition for a structured news report is very general. In practice, we would expect a DTD for a given domain. So for example, we would expect that for an implemented system that merges weather reports, there would be a corresponding DTD. One of the roles of a DTD, say for weather reports, would be to specify the minimum constellation of tags that would be expected of a weather report. We may also expect integrity constraints represented in classical logic to further restrict appropriate structured news reports for a domain.

Clearly each structured news report is isomorphic to a tree with the non-leaf nodes being the tagnames and the leaf nodes being the textentries.

Definition 2.2 *Let $\sigma_1, .., \sigma_n$ be tagnames occurring in the structured news report $\langle\phi\rangle\psi_1, .., \psi_n\langle\phi\rangle$. For this, $\sigma_1/../\sigma_n$ is a* **branch** *of this structured news report iff σ_1 is the root, and for all i, if $1 \leq i < n$, then σ_i is the parent of σ_{i+1}.*

A skeleton is equivalent to a structured news report without text entries. It is the underlying structure without the content. We need this definition later.

Definition 2.3 *A* **skeleton** *is of the form $\phi(\psi_1, .., \psi_n)$ where ϕ is a tagname and $\psi_1, .., \psi_n$ are skeletons. If ψ_i is a tagname, then it is a skeleton. A skeleton $\phi(\psi_1, .., \psi_n)$ can be regarded as a tree.*

In a NewsFusion System, we assume a naming of the input reports with names from a set of report names. In this paper we use $\{\mathtt{in_1}, .., \mathtt{in_n}\}$ as the set of report names. When a set of reports is given as input, we need a process of **registration** to assign each report with a report name. If all reports are treated equally by a set of fusion rules, we can make an arbitrary assignment. But in some cases, we may want to treat them differently. For example, we may want to assign the weather report from the BBC website to $\mathtt{in_1}$, and the weather report from the CNN website to $\mathtt{in_2}$. Registration is therefore a process that needs to be done before executing the fusion rules. Once a structured news report has been registered, we can refer to it by name.

3 Fusion rules

In this section, we present the framework for specifying and using a set of fusion rules.[3]

3.1 Schema variables

When merging a set of reports, a set of fusion rules is instantiated with the information (textentries) from the reports. Schema variables specify what to instantiate and from which source. There are two types of schema variable, namely textentry variable and set variable.

Definition 3.1 *Let $\phi_1/../\phi_n$ be a branch, and let $\mathtt{in_i}$ be a report. A* **textentry variable** *is denoted $\mathtt{in_i}//\phi_1/.../\phi_n$. A* **set variable** *is denoted $//\phi_1/.../\phi_n$. A* **schema variable** *is either a set variable or a textentry variable. Let the set of schema variables be denoted $\mathcal{S}$.*

1. *For a report $\mathtt{in_i}$, $\mathtt{T}$ is a* **valid grounding for a textentry variable** *$\mathtt{in_i}//\phi_1/.../\phi_k$ iff $\mathtt{T}$ is a textentry and $\mathtt{T}$ is the child of ϕ_k in report $\mathtt{in_i}$.*
2. *For a set of input reports $\{\mathtt{in_1}, .., \mathtt{in_n}\}$, let $\{\mathtt{in'_1}, .., \mathtt{in'_m}\}$ be the largest subset of $\{\mathtt{in_1}, .., \mathtt{in_n}\}$ where for each $\mathtt{in'_i} \in \{\mathtt{in'_1}, .., \mathtt{in'_m}\}$, there is a valid grounding for the textentry variable $\mathtt{in'_i}//\phi_1/.../\phi_k$. Using $\{\mathtt{in'_1}, .., \mathtt{in'_m}\}$, the list $[\mathtt{T'_1}, .., \mathtt{T'_m}]$ is a* **valid grounding for a set variable** *$//\phi_1/.../\phi_k$ iff $\mathtt{T'_1}$ is a* **valid grounding for a textentry variable** *$\mathtt{in'_1}//\phi_1/.../\phi_k$ and ... and $\mathtt{T'_m}$ is a* **valid grounding for a textentry variable** *$\mathtt{in'_m}//\phi_1/.../\phi_k$*

Example 3.1 *The valid grounding for the variable $\mathtt{in_1}//\mathtt{weatherreport/today}$ is* `showers` *if we let $\mathtt{in_1}$ be the top left report in Example 1.1. Similarly, the valid grounding for $\mathtt{in_2}//\mathtt{weatherreport/today}$ is* `inclement` *if we let $\mathtt{in_2}$ be the top right report in Example 1.1.*

Example 3.2 *The valid grounding for the set variable $//\mathtt{weatherreport/today}$ is $[\mathtt{showers}, \mathtt{inclement}]$ if we let $\mathtt{in_1}$ and $\mathtt{in_2}$ be the two top structured news reports in Example 1.1.*

[3]The definitions in this section extend the syntax for fusion rules given in [HS03b] by introducing logical variables. Due to space pressures, we have omitted consideration of subtrees in this paper. As explained in [HS03b], these allow schema variables to take subtrees from the structured news reports. Since each structured news report is isomorphic to a ground logic term, these subtrees are treated as ground logical terms in fusion rules.

3.2 Syntax for fusion rules

In the following definition, we augment the usual definition for a classical logic language with notation for schema variables which are just placeholders to be instantiated with textentries before logical reasoning. In effect they provide the "input" for a fusion rule. Logical variables are the other kind of variables that we use in fusion rules. They are just the usual classical variables. As we explain below, after we ground the schema variables, the literals in the antecedent of a fusion rule are conditions to be evaluated in a Prolog knowledgebase, and the logical variables are ground by the Prolog knowledgebase if the respective conditions can be evaluated to "true".

Definition 3.2 *We assume the following sets of symbols: (1) $\mathcal{C}$ is a set of constants; (2) $\mathcal{V}$ is a set of logical variables; (3) $\mathcal{S}$ is a set of schema variables; (4) $\mathcal{F}$ is a set of functions; and (5) $\mathcal{P}$ is a set of predicates.*

1. *The set of simple terms $\mathcal{T}_1$ is $\mathcal{C} \cup \{f(d_1, .., d_k) \mid f \in \mathcal{F}$ and $d_1, .., d_k \in \mathcal{C} \cup \mathcal{V} \cup \mathcal{S}\}$.*
2. *The set of schemafree terms $\mathcal{T}_2$ is $\mathcal{C} \cup \{f(d_1, .., d_k) \mid f \in \mathcal{F}$ and $d_1, .., d_k \in \mathcal{C} \cup \mathcal{V}\}$.*
3. *The set of ground terms $\mathcal{T}_3$ is $\mathcal{C} \cup \{f(c_1, .., c_k) \mid f \in \mathcal{F}$ and $c_1, .., c_k \in \mathcal{C}\}$.*
4. *The set of simple atoms $\mathcal{A}_1$ is $\{p(t_1, .., t_k) \mid p \in \mathcal{P}$ and $t_1, .., t_k \in \mathcal{T}_1\}$.*
5. *The set of schemafree atoms $\mathcal{A}_2$ is $\{p(t_1, .., t_k) \mid p \in \mathcal{P}$ and $t_1, .., t_k \in \mathcal{T}_2\}$.*
6. *The set of ground atoms $\mathcal{A}_3$ is $\{p(t_1, .., t_k) \mid p \in \mathcal{P}$ and $t_1, .., t_k \in \mathcal{T}_3\}$.*

We assume that all possible textentries are constant symbols in $\mathcal{C}$. We also assume that if $\phi_1/../\phi_n$ is a branch, then it is a constant symbol in $\mathcal{C}$. Similarly, if $\phi(\psi_1, .., \psi_n)$ is a skeleton, then it is a constant symbol in $\mathcal{C}$. We also assume that if α is an atom, then α is a literal, and $\neg\alpha$ is a literal. Let $\mathcal{L}_1$ be the literals formed from $\mathcal{A}_1$, $\mathcal{L}_2$ be the literals from $\mathcal{A}_2$, and $\mathcal{L}_3$ be the literals from $\mathcal{A}_3$.

We now consider the definition for fusion rules which may contain both logical variables and schema variables.

Definition 3.3 *A* **fusion rule** *is of the following form where $\alpha_1, .., \alpha_n \in \mathcal{L}_1$ and $\beta_1, .., \beta_m \in \mathcal{A}_1$.*

$$\alpha_1 \wedge .. \wedge \alpha_n \Rightarrow \beta_1 \wedge .. \wedge \beta_m$$

We call $\alpha_1, .., \alpha_n$ the condition literals and $\beta_1, .., \beta_m$ the action atoms.

We regard a fusion rule that incorporates schema variables, as a scheme for one or more classical formulae. These classical formulae are obtained by grounding schema variables as we explain below. We discuss condition literals in Section 3.3 and action atoms in Section 3.4.

Example 3.3 *Consider the following condition literal with the set of input reports* $\{\mathtt{in}_1, \mathtt{in}_2, \mathtt{in}_3\}$

$$\mathtt{SameCity}(//\mathtt{weatherreport/city})$$

Now suppose that `London` *is the textentry on the* `weatherreport/city` *branch for reports* $\mathtt{in}_1$ *to* $\mathtt{in}_3$. *So after grounding the schema variable, we have the following ground condition literal.*

$$\mathtt{SameCity}([\mathtt{London}, \mathtt{London}, \mathtt{London}])$$

Definition 3.4 *A* **schemafree fusion rule** *is a fusion rule with every schema variable systematically replaced by a valid grounding according to Definition 3.1.*

Example 3.4 *The following is a propositional fusion rule that could be used for Example 1.1 where the set of news reports is given by the two top reports in Example 1.1.*

$$\texttt{Samecity}(//\texttt{weatherreport/city}) \Rightarrow \texttt{AddText}(\texttt{in}_1//\texttt{weatherreport/city}, \texttt{weatherreport/city})$$

The schemafree fusion rule obtained with the fusion rule is the following.

$$\texttt{Samecity}([\texttt{London}, \texttt{London}]) \Rightarrow \texttt{AddText}(\texttt{London}, \texttt{weatherreport/city})$$

So if $\alpha'_1 \wedge .. \wedge \alpha'_n \Rightarrow \beta'_1 \wedge .. \wedge \beta'_m$ is a fusion rule, then it is a schemafree fusion rule iff $\alpha'_1, .., \alpha'_n \in \mathcal{L}_2$ and $\beta'_1, .., \beta'_m \in \mathcal{A}_2$. A schemafree fusion rule is a formula of classical predicate logic where the logical variables are implicitly universally quantified, and the universal quantifiers are assumed to be outermost. In other words, if $\alpha'_1 \wedge .. \wedge \alpha'_n \Rightarrow \beta'_1 \wedge .. \wedge \beta'_m$ is a schemafree fusion rule, and $X_1, .., X_m$ are the free variables in $\alpha'_1, .., \alpha'_n, \beta'_1, .., \beta'_m$, then $\forall X_1, .., X_m\ \alpha'_1 \wedge .. \wedge \alpha'_n \Rightarrow \beta'_1 \wedge .. \wedge \beta'_m$ is a classical predicate formula.

3.3 Condition literals

The condition literals in fusion rules relate the contents of structured news reports to the background knowledge. There are many possible condition literals that we could define that relate one or more features from one or more structured news reports to the background knowledge. To illustrate, these literals may include the following kinds: $\texttt{SameDate}(\texttt{L})$ where the textentries in the list $\texttt{L}$ refer to the same date; $\texttt{SameCity}(\texttt{L})$ where the textentries in $\texttt{L}$ refer to the same city; $\texttt{Synonymous}(\texttt{L})$ where the textentries in $\texttt{L}$ are synonyms; $\texttt{Coherent}(\texttt{L})$ where the textentries in $\texttt{L}$ are mutually coherent; and $\texttt{WithinRange}(\texttt{L}, \texttt{S})$ where the values represented by the textentries in $\texttt{L}$ are in the range specified by $\texttt{S}$.

Example 3.5 *Some examples of condition literals (when ground) may include the following.*

$$\begin{gathered}
\texttt{SameDate}([\texttt{14April2003}, \texttt{14/04/03}, \texttt{14.4.03}, \texttt{14/04/03}]) \\
\texttt{SameCity}([\texttt{Mumbai}, \texttt{Bombay}]) \\
\texttt{Coherent}([\texttt{sun}, \texttt{sunny}, \texttt{sun}]) \\
\texttt{Coherent}([\texttt{showers}, \texttt{inclement}]) \\
\texttt{Coherent}([\texttt{sun}, \texttt{rain}]) \\
\texttt{Synonymous}([\texttt{showers}, \texttt{rain}]) \\
\texttt{WithinRange}([\texttt{11C}, \texttt{12C}, \texttt{11C}, \texttt{10C}, \texttt{12C}, \texttt{15C}, \texttt{14C}], 10 - \texttt{15C})
\end{gathered}$$

The condition literals are evaluated by querying the Prolog knowledgebase.

Example 3.6 *In the following condition,* $\texttt{X}$ *is a logical variable, and* $\texttt{Lub}$ *is a predicate that captures the least upper bound of the textentries on the* $\texttt{weatherreport/precipitation}$ *branch of the structured news reports. In other words,* $\texttt{X}$ *will be ground by the Prolog knowledgebase with a value that according to the background knowledge is more general than or equal to the values in the input reports.*

$$\texttt{Lub}(//\texttt{weatherreport/precipitation}, \texttt{X})$$

If $//\texttt{weatherreport/precipitation}$ *is ground as* $[\texttt{rain}, \texttt{showers}, \texttt{intermittent rain}, \texttt{rainy}]$*, then the query sent to the Prolog knowledgebase is the following.*

$$\texttt{Lub}([\texttt{rain}, \texttt{showers}, \texttt{intermittent rain}, \texttt{rainy}], \texttt{X})$$

The grounding for X *returned by the Prolog knowledgebase is* rain*. To get this outcome, the Prolog knowledgebase incorporates an inheritance hierarchy capturing a set of textentries and their relative generalization.*

Now we consider grounding logical variables.

Definition 3.5 *A* **ground fusion rule** *is a fusion rule formed from a schemafree fusion rule with every logical variable systematically replaced by a valid grounding by the Prolog knowledgebase.*

So if $\alpha''_1 \wedge .. \wedge \alpha''_n \Rightarrow \beta''_1 \wedge .. \wedge \beta''_m$ is a fusion rule, then it is a ground fusion rule iff $\alpha''_1, .., \alpha''_n \in \mathcal{L}_3$ and $\beta''_1, .., \beta''_m \in \mathcal{A}_3$.

In our Java prototype for executing fusion rules, conditions are evaluated from left to right. So for a schemafree fusion rule $\alpha'_1 \wedge .. \wedge \alpha'_n \Rightarrow \beta'_1 \wedge .. \wedge \beta'_m$, first the condition literal α'_1 is processed as a prolog query, then the the condition literal α'_2 is processed as a prolog query, and so on. This means the query variables are evaluated and the result instantiated systematically from left to right. In the current implementation, negation is treated as negation-as-failure in Prolog.

We assume the background knowledge is defined by a knowledge engineer building a NewsFusion System for an application. Since the background knowledge is handled in Prolog, there is the possibility of multiple instantiations for query variables. So far we have constrained the knowledgebase to provide unique instantiations, but an alternative strategy is to take the first answer only. As will be apparent in Section 3.5, we do not support backtracking of fusion rules.

3.4 Action atoms

Action atoms specify the structure and content for a merged report. In a ground fusion rule $\alpha'' \Rightarrow \beta''$, if the ground literals in the antecedent α'' hold, then the merged report should meet the specification represented by the ground atoms in β''. We look at this in more detail in the next section. Here we consider the syntax for action literals. In Definition 3.6, we define a basic set of action atoms. A number of further definitions for action atoms are possible.

Definition 3.6 *The* **action atoms** *are literals that include the following specifying how the merged report should be constructed.*

1. $\texttt{Initialize}(\phi(\psi_1, .., \psi_n))$ *where* $\phi(\psi_1, .., \psi_n)$ *is a skeleton constant. The intended action is to start the construction of the merged structured news report with* $\phi(\psi_1, .., \psi_n)$ *defining the basic structure. So the root of the merged report is* ϕ.
2. $\texttt{AddText}(\texttt{T}, \phi_1/../\phi_n)$ *where* T *is a textentry, and* $\phi_1/../\phi_n$ *is a branch constant. The intended action is to add the textentry* T *as the child to the tagname* ϕ_n *in the merged report on the branch* $\phi_1/../\phi_n$.
3. $\texttt{AddNode}(\texttt{N}, \phi_1/../\phi_n)$ *where* N *is a tagname for a node, and* $\phi_1/../\phi_n$ *is a branch constant. The intended action is to add* N *as the child to the tagname* ϕ_n *in the merged report on the branch* $\phi_1/../\phi_n$.

The action atoms are specifications that are intended to be made to hold by producing a merged report that satisfies the specification. For further analysis of the logic of action atoms, see [Hun02a].

Example 3.7 *Consider the action atom in the consequent of Example 3.4. After grounding the schema variable, the action atom is now the following:*

$$\texttt{AddText}(\texttt{London}, \texttt{weatherreport/city})$$

This specifies that the textentry should be `London` *in the merged report for the tagname* `city` *on the branch* `weatherreport/city`.

3.5 Rule execution

In order to use a set of fusion rules, we need to be able to execute them with background knowledge and a set of reports. Essentially each rule in turn is ground with the information extracted from the input set of structured news reports. Then for each ground fusion rule $\alpha'' \Rightarrow \beta''$, if each of the conditions in α'' is true, then all of the actions in β'' are added to the actionlist, and so the merged report should meet the specification β''. In the Java implementation of the modules for NewsFusion Systems, this is the basis of the key algorithm in the FusionEngine. The ground actions in the actionlist specify the structure and content for a merged report. The minimum we expect of the ground actions in an actionlist is that a merged report is produced that meets the specification. In the Java implementation, this is the basis of the key algorithm in the ActionEngine.

Example 3.8 *The pair of reports given in the top of Example 1.1, together with an appropriate set of fusion rules and an appropriate background knowledge would be executed by a NewsFusion System to give the following actionlist which specifies the merged report given in the bottom of Example 1.1.*

```
Initialize(weatherreport(source, date, country, today, temp(max, min), tomorrow))
AddText(TV1 and TV3, weatherreport/source)
AddText(19.03.02, weatherreport/date)
AddText(London, weatherreport/city)
AddText(showers and inclement, weatherreport/today)
AddText(7 to 9C, weatherreport/temp/max)
AddText(2 to 3C, weatherreport/temp/min)
AddText(sun or rain, weatherreport/tomorrow)
AddNode(dewpoint, weatherreport)
AddText(2C, weatherreport/dewpoint)
```

The fusion rules in a NewsFusion System need to be engineered so that exactly one `Initialize` atom is obtained for any set of reports that are to be covered by the system, and no `Initialize` atom is to be obtained for any set of reports that are not to be covered by the NewsFusion System.

The set of action atoms that we have defined in this paper is only part of the range of possible action atoms. We have implemented others including $\texttt{ExtendTree}(\texttt{T}, \phi_1/../\phi_n)$ where `T` is a ground term representing a tree (which is isomorphic to part of a structured news report), and $\phi_1/../\phi_n$ is a branch constant, and the intended action is to extend the merged report (which is also a tree) with `T` so that the tagname for the root of `T` is ϕ_n on the branch $\phi_1/../\phi_n$.

4 Aggregation functions as conditions

A common problem in merging reports arises when they do not agree on a particular textentry. For example, for `weatherreport/precipitation`, we may have the first report with textentry `rain`, the second with `rain`, and the third with `sun`, which we can represent by the list [`rain`, `rain`, `sun`]. In order to select a

Condition literal with logical variable	Instantiation of logical variable
$\texttt{Majority}([\texttt{rain}, \texttt{rain}, \texttt{snow}], \texttt{X})$	rain
$\texttt{Majority}([\texttt{rain}, \texttt{rain}, \texttt{snow}, \texttt{showers}, \texttt{sleet}], \texttt{X})$	NoMajority
$\texttt{Majority}([\texttt{rain}, \texttt{rain}, \texttt{snow}, \texttt{snow}], \texttt{X})$	NoMajority
$\texttt{FirstPastThePost}([\texttt{rain}, \texttt{rain}, \texttt{snow}], \texttt{X})$	rain
$\texttt{FirstPastThePost}([\texttt{rain}, \texttt{rain}, \texttt{snow}, \texttt{showers}, \texttt{sleet}], \texttt{X})$	rain
$\texttt{FirstPastThePost}([\texttt{rain}, \texttt{rain}, \texttt{snow}, \texttt{snow}], \texttt{X})$	NoFirstPastThePost
$\texttt{PopularSharing}([\texttt{rain}, \texttt{rain}, \texttt{snow}], \texttt{X})$	rain
$\texttt{PopularSharing}([\texttt{rain}, \texttt{rain}, \texttt{snow}, \texttt{showers}, \texttt{sleet}], \texttt{X})$	rain
$\texttt{PopularSharing}([\texttt{rain}, \texttt{rain}, \texttt{snow}, \texttt{snow}], \texttt{X})$	rain or snow

Table 1: Examples of condition literals with instantiation of query variables

textentry for the merged report, we need to apply an aggregation function to the list. There are a number of different approaches to aggregation functions that we may consider, but each of them can be captured as a condition literal. Note, the key advantage of using aggregation functions within fusion rules is that the choice of aggregation function can be dependent on the context of the input reports.

One approach, given in Example 3.6, is for the textentry in the merged report to be sufficiently general as to be more general than all the input textentries, but no more general than necessary. Here generality is semantic generality and it is defined in terms of a semantic network encoded in Prolog. Another approach is to take the logical generalization of the input textentries that is obtained by taking the disjunction of the input textentries. However, the approach we will consider in this section is to apply a form of voting. Each report "votes" for the merged textentry and based on this idea some aggregation functions, defined below, have been implemented in Prolog.

Definition 4.1 *Let Ω be a list of textentries of the form $[\psi_1, .., \psi_n]$. Since this is a tuple, there may be multiple occurrences of one or more textentries in the tuple. Let the set of textentries used in the tuple be the set of candidates Θ. Let S be a scoring function from Θ to $\{1, .., n\}$ such that $S(\psi)$ is the number of occurrences of ψ in Ω. Let $\mathsf{MaxVote}(\Omega) = \{\psi \in \Theta \mid$ for all $\psi' \in \Theta$ $S(\psi) \geq S(\psi')\}$. This gives the following schemafree atoms, as conditions, where* X *is a logical variable to be ground by the Prolog knowledgebase:*

1. *For* $\texttt{Majority}(\Omega, \texttt{X})$*, if* $\mathsf{MaxVote}(\Omega) = \{\psi_i\}$ *and* $S(\psi_i) > n/2$*, then* X *is ground to the textentry* ψ_i*, otherwise* X *is ground to the textentry* NoMajority*.*
2. *For* $\texttt{FirstPastThePost}(\Omega, \texttt{X})$*, if* $\mathsf{MaxVote}(\Omega) = \{\psi_i\}$*, then* X *is ground to the textentry* ψ_i*, otherwise* X *is ground to the textentry* NoFirstPastThePost*.*
3. *For* $\texttt{PopularSharing}(\Omega, \texttt{X})$*,* X *is ground to the text entry* ψ'_1 or...or ψ'_k *where* $1 \leq k \leq n$ *and* $\mathsf{MaxVote}(\Omega) = \{\psi'_1, ..., \psi'_k\}$*.*

Here we just merge textentries with no consideration of background knowledge. For examples, see Table 1. However, we can introduce background knowledge. A simple development is just to consider synonyms.

Definition 4.2 *Let Π be the set of all textentries to be used in a NewsFusion System, and let $\Pi_2 \subseteq \Pi$ be the set of preferred terms, where P is a mapping from Π to Π_2 such that ψ_1 and ψ_2 are synonyms iff $P(\psi_1) = P(\psi_2)$. Let Ω be a tuple of textentries of the form $[\psi_1, .., \psi_n]$. Since this is a tuple, there may be multiple occurrences of one or more textentries in the tuple. Let the set of textentries used in the tuple be the set of candidates Θ. Let S be a scoring function from Θ to $\{1, .., n\}$ such that $S(\psi)$ is the number* *of occurrences of ψ in Ω. Let S_2 be a scoring function from $\Theta \cap \Pi_2$ to $\{1,..,n\}$ such that $S_2(\psi)$ is the number of occurrences of ψ, and of synonyms of ψ, in Ω, defined as follows.*

$$S_2(\psi) = \sum_{\psi' \in \Theta \text{ s.t. } P(\psi')=\psi} S(\psi')$$

Let $\mathsf{MaxVote2}(\Omega) = \{\psi \in \Theta \cap \Pi_2 \mid \textit{for all } \psi' \in \Theta \cap \Pi_2 \ S_2(\psi) \geq S_2(\psi')\}$. *This gives the following schemafree atoms, as conditions, where* `X` *is a logical variable to be ground by the Prolog knowledgebase:*

1. *For* $\texttt{Majority2}(\Omega, \texttt{X})$*, if* $\mathsf{MaxVote2}(\Omega) = \{\psi_i\}$ *and* $S(\psi_i) > n/2$*, then* `X` *is ground to the textentry* ψ_i*, otherwise* `X` *is ground to the textentry* `NoMajority`.
2. *For* $\texttt{FirstPastThePost2}(\Omega, \texttt{X})$*, if* $\mathsf{MaxVote2}(\Omega) = \{\psi_i\}$*, then* `X` *is ground to the textentry* ψ_i*, otherwise* `X` *is ground to the textentry* `NoFirstPastThePost`.
3. *For* $\texttt{PopularSharing2}(\Omega, \texttt{X})$*,* `X` *is ground to the textentry* ψ'_1 `or`...`or` ψ'_k *where* $1 \leq k \leq n$ *and* $\mathsf{MaxVote2}(\Omega) = \{\psi'_1, ..., \psi'_k\}$*, otherwise* `X` *is ground to the textentry* `NoPopularSharing`.

Example 4.1 *Let* Ω *be* $[\texttt{rain}, \texttt{rain}, \texttt{rainy}, \texttt{snow}]$*, and so* Θ *is* $\{\texttt{rain}, \texttt{rainy}, \texttt{snow}\}$*. Suppose* $\Theta \cap \Pi_2$ *is* $\{\texttt{rain}, \texttt{snow}\}$*, where* $P(\texttt{rainy}) = \texttt{rain}$*. So for the condition literal* $\texttt{Majority}([\texttt{rain}, \texttt{rain}, \texttt{rainy}, \texttt{snow}], \texttt{X})$*,* `X` *is ground to* `rain`.

The condition literals given in Definitions 4.1 and 4.2 can be used in fusion rules with the corresponding definition in the Prolog knowledgebase. We can show these condition literals are equivalent to some of the aggregation functions defined in voting theory (social choice theory) (see for example [Kel88]). Variations on these definitions allow formalization of aggregation functions that adhere to the Alchourron-Gardenfors-Makinson axioms [KP98]. A further enhancement can include the weighting of sources to allow weighted voting [Lin96]. These are just indicative of the range of further aggregation functions that we can capture as condition literals.

5 Weather reports case study

The weather reports case study is based on merging weather reports obtained from websites including www.bbc.co.uk and www.cnn.com. These weather reports are already in the form of structured news reports though not in the form of XML. They were marked up in XML by hand, though it would be straightforward to automate the construction of each XML file from the HTML file. The DTD for the weather reports contained 20 elements. A condensed version of the DTD is as follows where α is instantiated with any of `source`, `date`, `city`, `today`, `max`, `min`, `windspeed`, `relativehumidity`, `sunrise`, `sunset`, `absolutevalue`, `directionofchange`, `visibility`, `airpollutionindex`, `sunindex`, and `dewpoint`.

$$\begin{array}{l}
\langle \texttt{!ELEMENT weatherreport(source, date, city, today, temp,} \\
\qquad\qquad \texttt{windspeed, relativehumidity, daylighthours,} \\
\qquad\qquad \texttt{pressure, visibility, visibilitydistance,} \\
\qquad\qquad \texttt{airpollutionindex, sunindex, sunindexrating,} \\
\qquad\qquad \texttt{dewpoint)} \rangle \\
\langle \texttt{!ELEMENT temp(max, min)} \rangle \\
\langle \texttt{!ELEMENT daylighthours(sunrise, sunset)} \rangle \\
\langle \texttt{!ELEMENT pressure(absolutevalue, directionofchange)} \rangle \\
\langle \texttt{!ELEMENT } \alpha(\#\texttt{PCDATA}) \rangle
\end{array}$$

In addition to the DTD, the textentries for the structured news reports were delineated as follows: **Cities** Major UK cities; **Accepted Sources** Main UK TV and radio broadcasters, and national newspapers; **Today** Today's weather conditions in the form of a word or phrase from a selection of 66 possibilities (e.g. cloudy, scattered showers, mostly wet, prolonged rain, etc); **Date** Acceptable date formats for day (e.g. 1, 1st, etc.), month (e.g. 1, Jan, January, etc.), year (e.g. 1998, 98, etc.), and these can be separated by ",", ". ", "- ", "/ ", "\ " and " "; **Visibility** Quantitative (e.g. 1 Mile, 1 km, etc.); **Visibility** Qualitative from the selection of good, fair, very good, excellent, unlimited, moderate, and poor; **Pressure** Absolute value (e.g. 1021.0 mb, etc.); **Pressure** Direction of change from the selection of rising, falling, and steady; **Max and Min Temperatures** (e.g. -1C, 23.9F, etc.); **Windspeed** (e.g. 15mph, 23kph, etc.); **Humidity** (e.g. 80%, etc.); **Time** (e.g. 10.13, 22.13, etc.); **SunindexRating** from the selection of low, high, and medium; **Sunindex** and **Airpollutionindex** 1, 2, 3, etc.

To merge information in reports of this kind we required 31 fusion rules and a Prolog knowledgebase with 425 clauses. The NewsFusion System developed had no problem merging the information from ten of these reports at any one time, and there is no reason not to suppose that it could merge substantially more (say, 20-30) in a timely fashion.

The fusion rules fall into one of two classes, foundational and optional, depending upon the nature of the conditions involved. For most domains of interest, certain conditions enjoy a privileged position in that if these conditions are not met, the reports in question should not be merged. These key conditions are combined into a single rule, whose status is then flagged as being foundational in the FusionRuleML: If any condition fails (causing the rule to fail), no other rule is executed and no merged report is constructed. In the weather report case study, for example, if the reports are not for the same date, location, and from accepted sources, no merged report should be constructed.

The case is different for all other report elements (whether those elements are expected to appear in all reports, or only expected to occur in some reports): The fusion rules flagged as optional handle these other report elements. Furthermore, for these report elements, we need a fusion rule telling us what to do if a certain condition concerning one of those elements holds; we also need one or more fusion rules telling us what to do if that condition *fails*. Thus, in the case study, rules that deal with elements that are not foundational occur in pairs. For example, if the dewpoints forecast in the reports in Example 1.1 all fall within the accepted range, then an interval consisting of the minimum and maximum values, together with the unit suffix, is constructed. If, on the other hand, not all forecast dewpoints fall within that range, then we have a second rule that discards outlying values, and constructs a similar interval from the remaining values.

Example 5.1 *A pair of rules (displayed in logical form) from the case study dealing with the entry for today's weather. The rule code indicates that these have been registered as the twelfth and thirteenth rules in the rulebase. The status of both is optional, indicating that even if either fails, other rules may be executed to produce a merged report. Rule 12 says that if all the textentries for today's weather are from the same equivalence class, the preferred term from that class is selected and that text is added to the* `today` *tag in the merged report. If the first condition of rule 12 fails, then the first condition of rule 13 will succeed. The second condition of rule 13 constructs a list, substituting for each textentry for today's weather from the set of input reports the preferred term from the equivalence class to which it belongs. The disjunction of the most frequently occurring textentries is then chosen as the textentry in the merged report.*

The conditions literals in rule 13 therefore implement PopularSharing as in Definition 4.2

$\mathtt{Rulecode\ is\ 12\ (Status = optional)}$

$\mathtt{EquivalentTerms(//weatherreport/today)}$
$\mathtt{AND\ PreferredTerm(//weatherreport/today, X)}$
$\mathtt{IMPLIES\ AddText(X, weatherreport/today)}$

$\mathtt{Rulecode\ is\ 13\ (Status = optional)}$

$\mathtt{NOT\ EquivalentTerms(//weatherreport/today)}$
$\mathtt{AND\ PreferredTerms(//weatherreport/today, X)}$
$\mathtt{AND\ PopularSharing(X, Y)}$
$\mathtt{IMPLIES\ AddText(Y, weatherreport/today)}$

Many predicates in the knowledgebase fall into one of two categories: those that deal with qualitative textentries, and those that deal with quantitative textentries. Textentries for `today`, `visibility`, `sunindexrating`, and `pressure` (direction of change), are qualitative (`rain`, `sun`, `moderate`, etc.). These are tested to see if they are members of the same equivalence class. (See Table 2 for equivalence classes for the textentries of `today`.) If they are, the preferred term for that class is used to construct the merged report. If they are not, then the goal `equivalentterms`(...) will fail, but another goal, `NOT equivalentterms`(...), will succeed in a subsequent rule. In this case each textentry is substituted by the preferred term for the equivalence class to which it belongs, and the disjunction of the most frequently occurring terms is used. (See example 5.1 above.)

Where textentries denote quantitative values (`temperature`, `windpseed`, `humidity`, `visibilitydistance`, `sunindex`, `airpollutionindex`, `time`, and `pressure`) the unit suffix, if present, is stripped from the textentry and any required unit conversion is carried out (e.g kph to mph, degrees F to C, etc.). The maximum and minimum values are then found and, if they are within the accepted range (as determined by the knowledge engineer), those values are returned as part of a textentry including the unit suffix. If, for example, the goal is $\mathtt{similarwindspeeds([10mph, 12mph, 11mph, 11mph, 24kph], X)}$, the grounding for $\mathtt{X}$ would be $\mathtt{10 - 14.9136mph}$. On the other hand, if the minimum and maximum values are not within the accepted range, this goal would fail, and another goal, `notsimilarwindspeeds`(...), would succeed. In this case, the converted values would be ordered and, provided they form a majority, the greatest number within the accepted range would be selected. A similar interval textentry would then be constructed from this subset of the original values.

The accepted ranges used are as follows: `visibilitydistance` (quantitative) accepted range is 5 miles; `pressure` (absolute value) accepted range is 5mb; `max` and `min` temperatures accepted range is 5 degrees; `daylighthours` accepted range is 5 minutes; `sunindex` accepted range is 2; `airpollution` accepted range is 2; and `dewpoint` accepted range is 2.

Another group of predicates deals with dates. By checking for common seperator characters ("/", "-", etc.), dates are segmented into day, month, and year. Corresponding segments can then be compared to see if they are members of an equivalence class (Jan, January, 1, etc.). At present, date strings must be in the order day-month-year, although it may be possible to use context (e.g location or source) as a reason for judging a string such as "`3/1/03`" to be in UK or US format.

From this case study, it is clear that the weather reports DTD allows a diverse range of potential conflicts to arise in sets of input reports. Yet many of these conflicts can be addressed systematically, correctly, and efficiently by a modest sized combination of a fusion rulebase and a Prolog knowledgebase within a NewsFusion System. Furthermore, it is straightforward to encode both the rulesbase and the knowledgebase. It is also straightforward to extend the fusion rules and the knowledgebase to handle any extensions or changes to the range of input reports.

Preferred term	Equivalence class
`cloud`	`cloud`, `cloudy`, `mostly cloud`, `mostly cloudy`, `overcast`, `heavy cloud`.
`sun`	`sun`, `sunshine`, `sunny`, `mostly sunny`, `fair`.
`sun and cloud`	`scattered cloud`, `scattered clouds`, `partly cloudy`, `patchy cloud`, `p/cloudy`, `sunny spells`, `sunny intervals`, `patchy sunshine`.
`haze`	`haze`, `hazy`.
`showers`	`showers`, `showery`, `scattered showers`, `patchy rain`, `intermittant rain`, `drizzle`, `heavy showers`, `blustery showers`.
`storms`	`storm`, `storms`, `stormy`, `thunder`, `lightning`, `thunder storms`, `thundery storms`, `thunder and lightning`, `severe storms`.
`ice`	`ice`, `icy`, `frost`, `frosty`, `ground frost`, `black ice`, `hoar frost`.
`snow`	`snow`, `snow showers`, `sleet`, `snow flurries`, `snowy`, `snowing`, `heavy snow`, `drifting snow`.
`rain`	`rain`, `rainy`, `wet`, `precipitation`, `inclement`, `mostly wet`, `mostly rain`, `heavy rain`, `prolonged rain`, `downpour`, `downpours`, `frequent downpours`, `rain/snow`.

Table 2: For the `today` tagname (representing today's conditions), the above are used as equivalence classes of textentry. If the entries for today from all input reports are in the same equivalence class, the preferred term is used in the merged report. If the entries are not all similar terms, each is converted into its preferred form, and the most frequently occurring term is used.

6 Discussion

Our logic-based approach differs from other logic-based approaches for handling inconsistent information such as belief revision theory (e.g. [Gar88, DP98, KM91, LS98]) and knowledgebase merging (e.g. [KP98, BKMS92]). These proposals are too simplistic in certain respects for handling news reports. Each of them has one or more of the following weaknesses: (1) One-dimensional preference ordering over sources of information — for news reports we require finer-grained preference orderings; (2) Primacy of updates in belief revision — for news reports, the newest reports are not necessarily the best reports; and (3) Weak merging based on a meet operator — this causes unnecessary loss of information. Furthermore, none of these proposals incorporate actions on inconsistency or context-dependent rules specifying the information that is to be incorporated in the merged information, nor do they offer a route for specifying how merged reports should be composed.

Other logic-based approaches to fusion of knowledge include the KRAFT system and the use of Belnap's four-valued logic. The KRAFT system uses constraints to check whether information from heterogeneous sources can be merged [PHG$^+$99, HG00]. If knowledge satisfies the constraints, then the knowledge can be used. Failure to satisfy a constraint can be viewed as an inconsistency, but there are no actions on inconsistency. In contrast, Belnap's four-valued logic uses the values "true", "false", "unknown" and "inconsistent" to label logical combinations of information (see for example [LSS00]). However, this approach does not provide actions in case of inconsistency.

Merging information is also an important topic in database systems. A number of proposals have been made for approaches based in schema integration (e.g. [PM98]), the use of global schema (e.g. [GM99]), and conceptual modelling for information integration based on description logics [CGL$^+$98b, CGL$^+$98a, FS99, PSB$^+$99, BCVB01]. These differ from our approach in that they do not seek an automated approach that uses domain knowledge for identifying and acting on inconsistencies. Heterogeneous and federated database systems are relevant, but they do not identify and act on inconsistency in a context-sensitive way [SL90, Mot96, CM01], though there is increasing interest in bringing domain knowledge into the process (e.g. [Cho98, SO99]). Also relevant is revision programming, a logic-based framework for describing and enforcing database constratints [MT98].

Our approach also goes beyond other technologies for handling news reports. The approach of wrappers offers a practical way of defining how heterogeneous information can be merged (see for example [HGNY97, Coh98, SA99]) and NLP/NLG offer some technqiues for identifying information in input reports and generating a natural langauge summary [BME99]. However, in these approaches there is little consideration of problems of conflicts arising between sources. Our approach therefore goes beyond these in terms of formalizing reasoning with inconsistent information and using this to analyse the nature of the news report and for formalizing how we can act on inconsistency.

Acknowledgements

The authors wish to thank Weiru Liu for helpful feedback on the paper.

References

[BCVB01] S Bergamaschi, S Castano, M Vincini, and D Beneventano. Semantic integration of heterogeneous information sources. *Data and Knowledge Engineering*, 36:215–249, 2001.

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

[BME99] R Barzilay, K McKeown, and M Elhadad. Information fusion in the context of multi-document summarization. In *Proceedings of 37th Annual Meeting of the Association of Computational Linguistics*, 1999.

[CGL+98a] D Calvanese, G De Giacomo, M Lenzerini, D Nardi, and R Rosati. Description logic framework for information integration. In *Proceedings of the 6th Conference on the Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13. Morgan Kaufmann, 1998.

[CGL+98b] D Calvanese, G De Giacomo, M Lenzerini, D Nardi, and R Rosati. Source integration in data warehousing. In *Proceedings of the 9th International Workshop on Database and Expert Systems (DEXA'98)*, pages 192–197. IEEE Computer Society Press, 1998.

[Cho98] L Cholvy. Reasoning with data provided by federated databases. *Journal of Intelligent Information Systems*, 10:49–80, 1998.

[CM01] L Cholvy and S Moral. Merging databases: Problems and examples. *International Journal of Intelligent Systems*, 10:1193–1221, 2001.

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

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

[FS99] E Franconi and U Sattler. A data warehouse conceptual data model for multidimensional aggregation. In S Gatziu, M Jeusfeld, M Staudt, and Y Vassiliou, editors, *Proceedings of the Workshop in Design and Management of Data Warehouses*, 1999.

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

[GM99] G Grahne and A Mendelzon. Tableau techniques for querying information sources through global schemas. In *Proceedings of the 7th International Conference on Database Theory (ICDT'99)*, Lecture Notes in Computer Science. Springer, 1999.

[HG00] K Hui and P Gray. Developing finite domain constraints – a data model approach. In *Proceedings of Computation Logic 2000 Conference*, pages 448–462. Springer, 2000.

[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.

[HS03a] A Hunter and R Summerton. FusionRuleML: Representing and executing fusion rules. Technical report, UCL Department of Computer Science, 2003.

[HS03b] A Hunter and R Summerton. Propositional fusion rules. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Lecture Notes in Computer Science. Springer, 2003. (in press).

[Hun00] A Hunter. Merging potentially inconsistent items of structured text. *Data and Knowledge Engineering*, 34:305–332, 2000.

[Hun02a] A Hunter. Logical fusion rules for merging structured news reports. *Data and Knowledge Engineering*, 42:23–56, 2002.

[Hun02b] A Hunter. Merging structured text using temporal knowledge. *Data and Knowledge Engineering*, 41:29–66, 2002.

[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.

[Lin96] J Lin. Integration of weighted knowledge bases. *Artificial Intelligence*, 83:363–378, 1996.

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

[LSS00] Y Loyer, N Spyratos, and D Stamate. Integration of information in four-valued logics under non-uniform assumptions. In *Proceedings of 30th IEEE International Symposium on Multiple-Valued Logic (ISMVL2000)*. IEEE Press, 2000.

[Mot96] A Motro. Cooperative database systems. *International Journal of Intelligent Systems*, 11:717–732, 1996.

[MT98] V Marek and M Truszczynski. Revision programming. *Theoretical Computer Science*, 190:241–277, 1998.

[PHG+99] A Preece, K Hui, A Gray, P Marti, T Bench-Capon, D Jeans, and Z Cui. The KRAFT architecture for knowledge fusion and transformation. In *Expert Systems*. Springer, 1999.

[PM98] A Poulovassilis and P McBrien. A general formal framework for schema transformation. *Data and Knowledge Engineering*, 28:47–71, 1998.

[PSB+99] N Paton, R Stevens, P Baker, C Goble, S Bechhofer, and A Brass. Query processing in the TAMBIS bioinformatics source integration system. In *Proceedings of the 11th International Conference on Scientific and Statistical Databases*, 1999.

[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.

[SL90] A Sheth and J Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22:183–236, 1990.

[SO99] K Smith and L Obrst. Unpacking the semantics of source and usage to perform semantic reconciliation in large-scale information systems. In *ACM SIGMOD RECORD*, volume 28, pages 26–31, 1999.