Logical fusion rules for merging structured news reports

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

January 10, 2002

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. 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 previous papers, we have presented a logic-based framework for merging items of potentially inconsistent structured text [Hun00a, Hun02b]. In this paper, we present fusion rules as a way of implementing logic-based fusion. 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. We give the syntax and mode of execution for fusion rules, and explain how the resulting actions give a merged report. We illustrate the presentation with examples of fusion rules for an application for merging weather reports.

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 an item of structured text as an abstraction of an item of free text.

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

```
⟨bid report :
      ⟨bid date : 30 May 2000⟩
      ⟨buyer :
            ⟨company :
                  ⟨name : France Telecom⟩
                  ⟨capitalization : 150 Billion Euros⟩
                  ⟨headquarters : Paris, France⟩
            : company⟩
      : buyer⟩
      ⟨target :
            ⟨company :
                  ⟨name : Orange⟩
                  ⟨headquarters : Bristol, UK⟩
            : company⟩
      : target⟩
      ⟨bid type : agreed⟩
      ⟨bid value : 40 Billion Euros⟩
      ⟨report info :
            ⟨source : Orange website⟩
            ⟨URL : www.orange.co.uk⟩
            ⟨report date : 31 May 2000⟩
      : report info⟩
: bid report⟩
```

Figure 1: An example of a news report in the form of structured text.

application, the prescribed lexicon delineates the types of states, actions, and attributes, that could be conveyed by the items of structured text. An example of structured text is given in Figure 1.

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. In particular, we are interested in merging potentially inconsistent sets of news reports [Hun00a, Hun02b], and deriving inferences from potentially inconsistent sets of news reports [Hun00b, Hun00c, BH01, Hun02a].

1.1 Our approach to fusion

In order to merge items of structured text, we need to take account of the contents of the structured text. Different kinds of content need to be merged in different ways. To illustrate, consider Examples 1.1 - 1.3 below.

Example 1.1 *Consider the following two conflicting weather reports which are for the same day* *and same city.*

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

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

```
⟨weatherreport:
     ⟨source: TV1 and TV3⟩
     ⟨date: 19.5.1999⟩
     ⟨city: London⟩
     ⟨today: sun⟩
     ⟨tomorrow: sun or rain⟩
: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 the merged report is:*

```
⟨weatherreport:
     ⟨source: TV1 and TV3⟩
     ⟨date: 19.5.1999⟩
     ⟨city: London⟩
     ⟨today: sun⟩
     ⟨tomorrow: rain⟩
:weatherreport⟩
```

Example 1.2 *Consider the following two structured reports which are for the same day but different regions.*

```
⟨weatherreport :                         ⟨weatherreport :
     ⟨date : 5 Nov 99⟩                        ⟨date : 5 Nov 99⟩
     ⟨regionalreport :                        ⟨regionareport :
          ⟨region : South East⟩                    ⟨region : North West⟩
          ⟨maxtemp : 20C⟩                          ⟨maxtemp : 18C⟩
     : regionalreport⟩                        : regionalreport⟩
: weatherreport⟩                         : weatherreport⟩
```

Here we may wish to take the union of the two `regionalreport` *features in the merged report, giving the following merged report,*

```
⟨weatherreport :
     ⟨date : 5 Nov 99⟩
     ⟨regionalreport :
          ⟨region : South East⟩
          ⟨maxtemp : 20C⟩
     : regionalreport⟩
     ⟨regionalreport :
          ⟨region : North West⟩
          ⟨maxtemp : 18C⟩
     : regionalreport⟩
: weatherreport⟩
```

Example 1.3 *Consider the following two weather reports for the same day and same city.*

```
⟨weatherreport:
     ⟨source: TV1⟩
     ⟨date: 1.8.1999⟩
     ⟨city: Paris⟩
     ⟨middayweather:
          ⟨precipitation: inclement⟩
          ⟨temperature: 20C⟩
     middayweather⟩
:weatherreport⟩
```

```
⟨weatherreport:
     ⟨source: TV3⟩
     ⟨date: 1.8.1999⟩
     ⟨city: Paris⟩
     ⟨middayweather:
          ⟨precipitation: showers⟩
          ⟨temperature: 18C⟩
     middayweather⟩
:weatherreport⟩
```

Here we may wish to take the conjunction of `inclement` *and* `showers`*, and range of* `18-20C` *in the merged report.*

```
⟨weatherreport:
     ⟨source: TV1 and TV3⟩
     ⟨date: 1.8.1999⟩
     ⟨city: Paris⟩
     ⟨middayweather:
          ⟨precipitation: inclement and showers⟩
          ⟨temperature: 18C-20C⟩
     :middayweather⟩
:weatherreport⟩
```

There are many further examples we could consider, each with particular features that indicate how the merged report should be formed.

In our approach to merging items of structured text, in particular structured news reports, we draw on domain knowledge to help produce merged reports. The approach is based on fusion rules defined in a logical meta-language. These rules are of the form $\alpha \Rightarrow \beta$, where if α holds, then β is made to hold. So we consider α as a condition to check the information in the structured reports and in the background information, and we consider β as an action to undertake to construct the merged report.

To merge a pair 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 pair of structured news reports and a set of fusion rules, we repeatedly apply the fusion rules until no more rules apply. The application of the fusion rules is then a monotonic process that builds up a set of actions that define how the structured news report should be merged. The information in this fixpoint is then used to construct a merged structured news report. In order to merge more than two reports, we can repeat the merging process iteratively.

1.2 Other approaches to fusion

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]), knowledgebase merging (e.g. [KP98, BKMS92]), and logical inference with inconsistent information (e.g. [MR70, Bre89, CRS93, BCD$^+$93, BDP95]). 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 also could be relevant in merging multiple news reports, 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]).

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]). However, 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.

1.3 The rest of the paper

In the rest of the paper, we develop the approach of logic-based fusion of news reports. In Section 2, we review the definitions for formalizing structured text. This includes coverage of timestamps and sourcestamps. It also includes coverage of some concepts for describing the structure of structured text. In Section 3, we present the syntax for fusion including fusion rules and background knowledge. Then, in Section 4, we define how a set of fusion rules together with background knowledge can be executed to generate a set of actions that can be used to build a merged structured news report. We explain how to use these actions to build a merged structured news report in Section 5. Then, in Section 6, we consider some properties of fusion systems based on fusion rules.

2 Formalizing structured text

In this section, we formalize the composition and structure of structured news reports.

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 examples 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**. *The* **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. The* **type** *of a complex feature is the semantic label ϕ. An* **item of structured text** *is just a complex feature.*

Definition 2.3 *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 2.4 *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. The function Root is defined as: $Root(\langle \phi : \sigma_1, .., \sigma_n : \phi \rangle) = \phi$*

Definition 2.5 *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 such as illustrated in Examples 1.1 - 1.3. We do not however assume any general conditions for classifying items of structured text as structured reports.

2.2 Timestamps and sourcestamps

There are four types of timestamp that we will consider for structured reports, namely an atomic pointbased timestamp, an atomic intervalbased timestamp, a complex pointbased timestamp, and a complex intervalbased timestamp, and we define these below.

Definition 2.6 *The set of* **temporal semantic labels** *is a subset of the set of semantic labels used for structured news reports.*

Example 2.2 *The set of temporal semantic labels includes* `time`*,* `date`*,* `publicationdate`*, and* `year`*.*

Definition 2.7 *The set of* **temporal text entries** *is a subset of the set of text entries used for structured news reports. A temporal text entry may refer to a point or interval in a clock and/or calendar. A temporal text entry is called a* **pointbased text entry** *if it refers to a point in a clock. And a temporal text entry is called an* **intervalbased text entry** *if it refers to an interval in a clock and/or calendar.*

Example 2.3 *Temporal text entries include* `14.00hrs`*,* `19 April 2000`*, and* `19/4/00`*. Temporal text entries may or may not include the units of time used. For example, both* `14.00` *and* `14.00hrs` *are temporal text entries.*

We will look more closely at the nature of points and intervals in the following definitions.

Definition 2.8 *An* **atomic pointbased timestamp** *is an atomic feature* $\langle \alpha : \beta \rangle$ *where* α *is a temporal semantic label referring to a particular clock and/or a particular calendar and* β *is a pointbased text entry with a value denoting a point in that clock and/or calendar.*

Example 2.4 *Examples of atomic pointbased timestamps include:*

$\langle \texttt{time} : \texttt{14.00hrs} \rangle$ $\langle \texttt{date} : \texttt{19 April 2000} \rangle$ $\langle \texttt{publicationdate} : \texttt{19/4/00} \rangle$ $\langle \texttt{year} : \texttt{2004} \rangle$

We assume the background knowledge includes axioms that make different formats for temporal text entries interchangeable, so that for example `19 April 2000` is equivalent to `19/4/2000`.

Definition 2.9 *An* **atomic intervalbased timestamp** *is an atomic feature* $\langle \alpha : \beta \rangle$ *where* α *is a temporal semantic label referring to a particular clock and/or a particular calendar and* β *is a intervalbased text entry with a value denoting an interval in that clock and/or calendar.*

We view time intervals as being either implicitly given as an interval with the start and end points being inferred, or explicitly given in terms of a start point and an end point.

Definition 2.10 *An* **explicit intervalbased text entry** *is of the form* `X-Y`*, where* `X` *and* `Y` *denote points, and an* **implicit intervalbased text entry** *of the form* `X` *where* `X` *describes a period of time without using explicit end points.*

Example 2.5 *Examples of atomic intervalbased timestamps with explicit intervalbased text entries include:*

$\langle \texttt{reportingperiod} : \texttt{1/1/00-31/3/00} \rangle$ $\langle \texttt{openhours} : \texttt{09.00-12.00} \rangle$

Example 2.6 *Examples of atomic intervalbased timestamps with implicit intervalbased text entries include:*

$\langle \texttt{reportingperiod} : \texttt{2004} \rangle$ $\langle \texttt{holidayperiod} : \texttt{Easter} \rangle$

So for `2004`*, the inferred start point in days is* `1/1/2004` *and the inferred endpoint in days is* `31/12/2004`*.*

It can appear difficult to distinguish some implicit intervalbased text entries from pointbased text entries. We address the problem of handling implicit intervals by reducing each intervalbased text entry to pointbased text entries. We assume that the background knowledge includes axioms for this (as discussed in [Hun02b]).

Definition 2.11 *A* **complex pointbased timestamp** *is either an atomic pointbased timestamp or a complex feature $\langle\psi : \phi_1, .., \phi_n : \psi\rangle$ where ψ is a temporal semantic label refering to a particular clock and/or a particular calendar and $\phi_1, .., \phi_n$ are complex pointbased timestamps that describe the point in that clock and/or calendar.*

In this paper, we will assume atomic and complex pointbased timestamps should be interchangeable by appropriate axioms in the background knowledge.

Example 2.7 *An example of a complex pointbased timestamp is:*

```
⟨date:
     ⟨day: 23⟩,
     ⟨month: April⟩,
     ⟨year: 2000⟩,
:date⟩
```

So we can assume this complex pointbased timestamp is equivalent to ⟨`date: 23 April 2000`⟩.

Definition 2.12 *A* **complex intervalbased timestamp** *is either an atomic intervalbased timestamp or a complex feature $\langle\psi : \phi_1, .., \phi_n : \psi\rangle$ where ψ is a temporal semantic label refering to a particular clock and/or a particular calendar and $\phi_1, .., \phi_n$ are complex pointbased timestamps that describe the interval in that clock and/or calendar.*

Also, in this paper, we will assume atomic and complex intervalbased timestamps should be interchangeable by appropriate axioms in the background knowledge.

Example 2.8 *An example of a complex intervalbased timestamp is:*

```
⟨university term:
     ⟨first day of term: 10/1/2000⟩,
     ⟨last day of term: 29/3/2000⟩,
:university term⟩
```

So we assume this complex intervalbased timestamp is equivalent to

⟨`university term: 10/1/2000-29/3/2000`⟩

In the rest of this paper, we assume each news report may have a timestamp which has type `date`, and may be pointbased or intervalbased, and complex or atomic. It may also have a sourcestamp which is an atomic feature of type `source` and text entry that describes what the source of the news report is. For example, for a weather report, we may have the atomic feature ⟨`source` : `BBC TV`⟩. Whilst we take a restricted position on timestamps and sourcestamps in this paper, we believe that the approach presented here can be extended to further types and combinations of timestamps and sourcestamps in structured text. For more information on using temporal knowledge in structured news reports see [Hun02b].

2.3 Structural information about structured news reports

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

Definition 2.13 *A* **skeleton** *is a tree (N, E, S) defined as follows: N is the set of nodes where each node is a semantic label; E is a set of edges 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 2.13, 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.

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. In other words, we use the semantic labels in a item of structured text as the name of the nodes in the skeleton. We call each such name, the **simple name** of the node. However, to obviate any problems arising from multiple occurrences of a semantic label in a complex feature, and hence the same name being used for different nodes, we can adopt the following definition for pedantic names for nodes. This definition uses the sequence of semantic labels used on the path from the root to the particular occurrence of a semantic label in a feature.

Definition 2.14 *The* **pedantic name** *for any node in a structured news report is defined inductively as follows: The pedantic name for the root of a structured news report is the semantic label for the report. For a nested feature $\langle \phi : \psi_1, ..., \psi_n : \phi \rangle$, let the pedantic name for ϕ be α/ϕ, then the pedantic name for the root of each ψ_i is $\alpha/\phi/Root(\psi_i)$. (If there is more than one ψ_i with the same semantic label at the root, then the occurrences can be differentiated by adding a superscript to the semantic label in the pedantic name.)*

Example 2.9 *The pedantic name for* `source` *in Figure 1 is* `bidreport/reportinfo/source`*.*

Essentially, a pedantic name is like a unix file name that can be given by the path of directories from the root. But where we do not have ambiguity, we will just use the simple name.

Definition 2.15 *The* **skeleton** *function, denoted Skeleton, is applied to a complex feature θ and returns the skeleton (N, E, S) for θ, where the set of nodes N is the set of names (simple or pedantic) formed from the semantic labels used in θ, and E and S are defined as follows, where ϕ^*, $Root(\psi_1)^*$,..,$Root(\psi_n)^*$ are either the simple names or pedantic names (as required) for the semantic labels ϕ, and the roots of ψ_1,..,ψ_n, respectively:*

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

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

So the skeleton function is defined to extract the tree structure of each item of structured text.

Definition 2.16 *Let $T_i = (N_i, E_i, S_i)$ and $T_j = (N_j, E_j, S_j)$ 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 $E_i \subseteq E_j$ and $S_i \subseteq S_j$.*

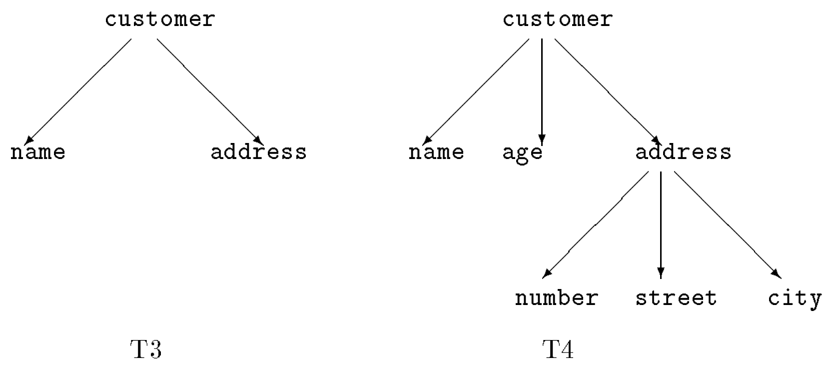

Figure 2: Assume T3 and T4 are both skeletons. Here, $T3 \preceq T4$ holds.

So for skeletons T_i and T_j, if we have $T_i \preceq T_j$, then the set of edges in T_i is a subset of the edges 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 2.17 *Let θ be a complex feature and let S be a skeleton. An* **instantiation** *of S by θ is defined as follows:*

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

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

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

With reference to Definition 2.5, 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 2.18 *Let* `F` *and* `F'` *be features such that* $\texttt{F} \in Sub(\texttt{F'})$ *and let* $Skeleton(\texttt{F'}) = (N, E, S)$*. A* **position** *of* `F` *in* `F'` *is a pedantic name for the root of* `F` *in* $Skeleton(N, E, S)$*.*

Definition 2.19 *Let $\langle \phi : \psi_1, .., \psi_n : \phi \rangle$ be a complex feature. The* **anchor** *for each complex feature ψ_i is ϕ. So each of $\psi_1, .., \psi_n$ is anchored at ϕ.*

The anchor is the position from which one or more complex features hang. In this way, the anchor gives the connection to the rest of structured text.

Example 2.10 *Consider Figure 1. Here we can see* ⟨`bid date: 30 May 2000`⟩ *is at position* `bidreport/biddate` *and it is anchored at* `bidreport`*.*

In this way, we are using the notions of nodes in skeletons and semantic labels in structured reports interchangeably.

3 Syntax for fusion

In this section, we give the syntax for the fusion rules and associated background knowledge for defining fusion systems. We assume fusion is undertaken on pairs of structured news reports, and that these reports are represented as logical terms in the logical fusion rules.

Definition 3.1 *Each complex feature is equivalent to a ground term called a* **feature term**. *If* $\langle \phi : \psi_1, .., \psi_n : \phi \rangle$ *is a complex feature, and* ψ'_1 *is a feature term that represents* ψ_1, *....., and* ψ'_n *is a feature term that represents* ψ_n, *then* $\phi(\psi'_1, .., \psi'_n)$ *is a feature term that represents* $\langle \phi : \psi_1, .., \psi_n : \phi \rangle$. *If* $\langle \phi : \psi \rangle$ *is a complex feature, then* $\phi(\psi)$ *is a feature term that represents it.*

Definition 3.2 *Each text entry is equivalent to a constant symbol called a text entry constant. So if* `T` *is a text entry, then* `T` *is a text entry constant.*

Example 3.1 *Consider the following item of structured text.*

$$\begin{array}{l}\langle \texttt{auctionreport} : \\ \quad \langle \texttt{buyer} : \langle \texttt{name} : \langle \texttt{firstname} : \texttt{John}\rangle, \langle \texttt{surname} : \texttt{Smith}\rangle : \texttt{name}\rangle : \texttt{buyer}\rangle, \\ \quad \langle \texttt{property} : \texttt{Lot37}\rangle \\ : \texttt{auctionreport}\rangle\end{array}$$

This can be represented by the feature term:

$$\texttt{auctionreport(buyer(name(firstname(John), surname(Smith))), property(Lot37))}$$

In defining fusion rules, we require three kinds of atom. These are structural atoms, background atoms, and action atoms. The structural atoms capture information about the structure and type of complex features, and the type and text entries for the atomic features, in individual structured news reports, and between paris of structured news reports. The background atoms relate information in individual structured news reports to the domain knowledge. Finally, action atoms capture instructions for building merged structured news reports.

Definition 3.3 *The* **structural atoms** *are atoms that include the following definitions:*

1. `FeatureType(F,T)` *where* `T` *is the type of feature* `F`.
2. `Report(R)` *where* `R` *is a structured news report.*
3. `TextEntry(A,E)` *where* `E` *is the text entry for the atomic feature* `A`.
4. `IncludeFeature(F,F')` *where* `F'` *is a feature in* `F`, *and so* `F'` $\in Sub($`F`$)$.
5. `AtomicFeature(F,A)` *where* `A` *is an atomic feature in* `F`.
6. `Position(F,P,R)` *where* `P` *is a position of the feature* `F` *in the report* `R`.
7. `Anchor(F,P,R)` *where* `P` *is a position of an anchor of the feature* `F` *in the report* `R`.
8. `SameSkeleton(F,F')` *where the features* `F` *and* `F'` *are such that* $Skeleton($`F`$) = Skeleton($`F'`$)$.
9. `SameTextEntry(A,A')` *where the atomic features* `A` *and* `A'` *have the same text entry.*
10. `NextSibling(P,P',R)` *where position* `P` *is an immediate sibling to the left of* `P'` *in report* `R`.

We denote the set of structural atoms by $\mathcal{S}$.

Example 3.2 *To illustrate some of these atoms, consider the following:*

$$\begin{array}{c}\texttt{TextEntry}(\texttt{city}(\texttt{London}),\texttt{London})\\ \texttt{IncludesFeature}(\texttt{address}(\texttt{street}(\texttt{GowerSt}),\texttt{city}(\texttt{London})),\texttt{city}(\texttt{London}))\\ \texttt{SameTextEntry}(\texttt{city}(\texttt{London}),\texttt{area}(\texttt{London}))\end{array}$$

Example 3.3 *Consider Figure 1. Let this report be denoted* `R`*, and let* $\langle$`bid date: 30 May 2000`$\rangle$ *be denoted* `F`*. For this, we have*

$$\begin{array}{c}\texttt{Position}(\texttt{F},\texttt{bidreport/biddate},\texttt{R})\\ \texttt{Anchor}(\texttt{F},\texttt{bidreport},\texttt{R})\end{array}$$

The structural atoms are evaluated by the underlying implementation for fusion. In other words, for any pair of structured news reports, the set of ground structural atoms that hold is completely determined. These atoms can be viewed as “built-in” predicates by analogy with Prolog.

We also require atoms that relate the contents of structured text to the background knowledge. These are background atoms and a partial list is given below.

Definition 3.4 *The* **background atoms** *are atoms that include the following definitions:*

1. `SameDate(F,F')` *where* `F` *and* `F'` *are timestamps that refer to the same time point.*
2. `SameSource(F,F')` *where* `F` *and* `F'` *are sourcestamps that refer to the same source.*
3. `SameCity(F,F')` *where* `F` *and* `F'` *are features that refer to the same city.*
4. `Source(R,F)` *where* `F` *is the sourcestamp in the report* `R`*.*
5. `Date(R,F)` *where* `F` *is the datestamp in the report* `R`*.*
6. `Coherent(F,F')` *where the features* `F` *and* `F'` *are coherent.*
7. `Prefer(F,F')` *where the feature* `F` *is preferred to the feature* `F'`*.*

We denote the set of background atoms by $\mathcal{B}$.

Example 3.4 *To illustrate background atoms consider the following literals that may be included in the background knowledge:*

$$\begin{array}{c}\texttt{SameDate}(\texttt{date}(\texttt{14Nov01}),\texttt{date}(\texttt{14.11.01}))\\ \texttt{SameCity}(\texttt{city}(\texttt{Mumbai}),\texttt{city}(\texttt{Bombay}))\\ \texttt{SameLog}(\texttt{log}(\texttt{date}(\texttt{day}(\texttt{14}),\texttt{month}(\texttt{11}),\texttt{year}(\texttt{01}))),\texttt{log}(\texttt{date}(\texttt{14.11.01})))\end{array}$$

Background atoms like `SameDate(F,F')`, `SameSource(F,F')`, and `SameLog(F,F')` are useful to determine when two features are equivalent and thereby indicate whether the features they come from are on the same topic. For example, if we have two reports, we can use these atoms to determine that the two reports have `SameCity` and `SameDate` holding, as a precondition before merging. We will use them as conditions in fusion rules below for this purpose.

The coherent relation captures when two features are mutually consistent. 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 3.5. Obviously different text entries for the same semantic label do not always suggest an inconsistency, as illustrated by Example 3.6.

Example 3.5 *Let θ_1 and θ_2 be two structured reports. Suppose* $\langle\texttt{weather} : \texttt{sun}\rangle$ *is an atomic feature in θ_1 and* $\langle\texttt{weather} : \texttt{rain}\rangle$ *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 3.6 *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".*

An example of a definition for the coherent relation is given below.

Example 3.7 *Let us assume we have the following background knowledge for identifying pairwise inconsistency in atomic features of type* `weather` *in weather reports:*

```
∀X Coherent(X,X)

¬Coherent(weather(sun),weather(rain))
¬Coherent(weather(cold),weather(hot))
¬Coherent(weather(cool),weather(hot))
¬Coherent(weather(cold),weather(warm))
¬Coherent(weather(cool),weather(warm))
¬Coherent(weather(sun),weather(snow))

Coherent(weather(snow), weather(sleet))
Coherent(weather(sun), weather(sunny))
Coherent(weather(rain), weather(rainy))
Coherent(weather(heavyrain), weather(storms))
Coherent(weather(rain), weather(showers))
Coherent(weather(showers), weather(unsettled))
```

This is likely to be only a partial list of literals required for this purpose. In addition, we will need various further formulae to define `Coherent` *for other types of atomic feature.*

The background atoms are evaluated by querying background knowledge. In the simplest case, the background knowledge may be just a set of ground background atoms that hold. However, we would expect the background knowledge would include classical quantified formulae that can be handled using automated reasoning. In any case, the background knowledge is defined by a knowledge engineer building a fusion system for an application.

Definition 3.5 *The set of* **check atoms***, denoted $\mathcal{C}$, is the union of the structural atoms and the background atoms.*

We now consider the syntax for action atoms, and this requires the definition for action functions.

Definition 3.6 *The set of* **action functions** *include the following functions that can be used as terms in the action atoms.*

1. `Interval(X,Y)` *where* `X` *and* `Y` *are text entries and the function returns an interval* `X-Y`*.*
2. `Conjunction(X,Y)` *where* `X` *and* `Y` *are text entries and the function returns a text entry* `X and Y`*.*
3. `Disjunction(X,Y)` *where* `X` *and* `Y` *are text entries and the function returns a text entry* `X or Y`*.*

We assume action functions are defined in the underlying implementation that uses the actions as instructions to build a merged report.

Example 3.8 *The following are examples of definitions for action functions:*

```
Interval(18C,25C) = 18-25C
Conjunction(TV1,TV3) = TV1 and TV3
Disjunction(sun,rain) = sun or rain
```

We now consider a basic set of action atoms. In Section 6, we consider extending the set of action atoms.

Definition 3.7 *The* **action atoms** *are atoms that include the following definitions:*

1. `CreateSkeleton(R)` *where* `R` *is a report. The resulting action is to create the skeleton for the merged report. The postcondition of this action is Skeleton*(`R`) *holding.*
2. `AddText(E,P)` *where* `E` *is a text entry, and* `P` *is a tree position. The resulting action is to add the text entry to the tree in position* `P`*. The precondition of this action is that there is no offspring, or text entry, for the semantic label at* `P`*. The postcondition of this action is that* `E` *is the text entry for the semantic label at* `P`*.*
3. `ExtendFeature(F,P)` *where* `F` *is a feature and* `P` *is a position. The resulting action is to extend the tree with* `F` *at position* `P`*. The preconditions for this are that the semantic label for Root*(`F`) *should equal the semantic label at* `P` *and there is no offspring, or text entry, for the semantic label at* `P`*. The postcondition of this action is that Root*(`F`) *is at* `P`*.*
4. `AddFeature(F,P)` *where* `F` *is a feature and* `P` *is a position. The resulting action is to add* `F` *to the tree in position* `P`*. The precondition of this action is that there is no text entry for the semantic label at* `P`*. The postcondition of this action is that the semantic label at position* `P` *is the anchor for* `F`*.*
5. `Populate(F,P)` *where* `F` *is a feature and* `P` *is a position. If there is a Skeleton*(`F`) *attached to position* `P` *in the tree, then the resulting action is to add the text entries in* `F` *to the vacant slots in Skeleton*(`F`) *in the tree. The preconditions of this action are that there are no text entries in the offspring of the semantic label at* `P` *and the skeleton rooted at* `P` *equals Skeleton*(`F`)*. The postcondition of this action is that Root*(`F`) *is at* `P`*.*

We denote the set of action atoms by $\mathcal{A}$*.*

Example 3.9 *Let* `R` *be the report on the left below. Then* `Skeleton(R)` *gives the item on the right below:*

```
⟨weatherreport:                      ⟨weatherreport:
     ⟨source: TV1 and TV3⟩                ⟨source: ⟩
     ⟨date: 19.5.1999⟩                    ⟨date: ⟩
     ⟨city: London⟩                       ⟨city: ⟩
     ⟨today: sun⟩                         ⟨today: ⟩
     ⟨tomorrow: sun or rain⟩              ⟨tomorrow: ⟩
:weatherreport⟩                      :weatherreport⟩
```

Example 3.10 *Consider* `AddText(BBC TV,weatherreport/source)`*. This instruction applied to the item below on the left gives the item below on the right.*

```
⟨weatherreport:         ⟨weatherreport:
     ⟨source: ⟩              ⟨source: BBC TV⟩
     ⟨date: ⟩                ⟨date: ⟩
     ⟨city: London⟩          ⟨city: London⟩
     ⟨today: ⟩               ⟨today: ⟩
     ⟨tomorrow: ⟩            ⟨tomorrow: ⟩
:weatherreport⟩         :weatherreport⟩
```

Example 3.11 *Consider the item on the left, and the feature* `F` *on the right:*

```
⟨weatherreport:           ⟨weathertoday:
     ⟨source: ⟩                ⟨precipitation: snow⟩
     ⟨date: ⟩                  ⟨temp: 0C⟩
     ⟨city: London⟩       :weathertoday⟩
     ⟨weathertoday: ⟩
:weatherreport⟩
```

The instruction `ExtendFeature(F,weatherreport/weathertoday)` *when applied to the item on the left above gives the following item:*

```
⟨weatherreport:
     ⟨source: ⟩
     ⟨date: ⟩
     ⟨city: London⟩
     ⟨weathertoday: ⟩
          ⟨precipitation: snow⟩
          ⟨temp: 0C⟩
     :weathertoday⟩
:weatherreport⟩
```

Implicit with the definition for `ExtendFeature` is the requirement to turn an atomic feature into a complex feature. This is also possible with `AddFeature` though not necessarily.

Example 3.12 *Consider the item on the left, and the feature* `F` *on the right:*

```
⟨weatherreport:         ⟨precipitation:
     ⟨source: ⟩              ⟨type: snow⟩
     ⟨date: ⟩                ⟨amount: 2cm⟩
     ⟨city: London⟩     :precipitation⟩
:weatherreport⟩
```

Then the instruction `AddFeature(F,weatherreport)` *gives the following item:*

```
⟨weatherreport:
      ⟨source: ⟩
      ⟨date: ⟩
      ⟨city: London⟩
      ⟨precipitation: ⟩
            ⟨type: snow⟩
            ⟨amount: 2cm⟩
      :precipitation⟩
:weatherreport⟩
```

Example 3.13 *Consider the item on the left, and the feature* `F` *on the right.*

```
⟨weatherreport:                ⟨weathertoday:
      ⟨source: ⟩                     ⟨precipitation: snow⟩
      ⟨date: ⟩                       ⟨temp: 0C⟩
      ⟨city: London⟩           :weathertoday⟩
      ⟨weathertoday: ⟩
            ⟨precipitation: ⟩
            ⟨temp: ⟩
      :weathertoday⟩
:weatherreport⟩
```

Then the instruction `Populate(F,weatherreport/weathertoday)` *gives the following*

```
⟨weatherreport:
      ⟨source: ⟩
      ⟨date: ⟩
      ⟨city: London⟩
      ⟨weathertoday: ⟩
            ⟨precipitation: snow⟩
            ⟨temp: 0C⟩
      :weathertoday⟩
:weatherreport⟩
```

The action atoms cause a structured news report to be constructed. They define the structure of the report, and the text entries in the report. We will explain how this can be done in Section 5. In the remainder of this section, we explain how we use these atoms in fusion rules.

Definition 3.8 *The set of atoms is $\mathcal{C} \cup \mathcal{A}$. An atom is ground if each term in the atom is ground. If an atom is not ground, then it is unground. The set of ground atoms is denoted $\mathcal{G}$.*

Definition 3.9 *Let $\{\neg, \vee, \wedge\}$ be the set of* **classical logical connectives***. The set of* **ground formulae***, denoted $\mathcal{F}$, is the set of all classical formulae that can be formed from $\mathcal{G}$ and the set of logical connectives using the usual inductive definition for classical logic.*

We leave consideration of quantification to Definitions 3.11 and 3.12. So in the above definition, if a formula contains an unground atom, then the formula will not be a well-formed formula of classical logic, because the free variable(s) will be unbound.

Definition 3.10 *A* **check formula** *is a formula composed from one or more check atoms and zero or more classical logical connectives using the usual inductive definition for classical logic formulae. An* **action formula** *is a formula composed from one or more action atoms and zero or more classical logical connectives using the usual inductive definition for classical logic formulae.*

In the following definition, we introduce a non-classical form of implication that is denoted by the $\Rightarrow$ symbol.

Definition 3.11 *A* **fusion rule** *is a rule of the following form where α is a check formula and β is an action formula.*

$$\alpha \Rightarrow \beta$$

We assume that each variable in each rule is implicitly universally quantified, with the universal quantifiers outermost (i.e. if $X_1, .., X_n$ are the free variables in $\alpha \Rightarrow \beta$, then the explicitly quantified version is $\forall X_1, .., X_n(\alpha \Rightarrow \beta)$).

Normally, β will be an atom or a conjunction of atoms. However, if it incorporates disjunction, then it captures non-determinism in the intended actions, and if it incorporates negation, then the negation captures a form of preclusion in the intended action.

Example 3.14 *The following are four examples of fusion rules.*

$$\begin{array}{l}\texttt{Report(R1)} \wedge \texttt{Report(R2)} \wedge \texttt{SameSkeleton(R1,R2)} \wedge \texttt{SameDate(R1,R2)}\\ \Rightarrow \texttt{CreateSkeleton(R1)}\end{array}$$

$$\begin{array}{l}\texttt{Report(R1)} \wedge \texttt{Report(R2)} \wedge \texttt{AtomicFeature(R1,A1)} \wedge \texttt{AtomicFeature(R2,A2)}\\ \wedge\texttt{Position(A1,P,R1)} \wedge \texttt{FeatureType(A1,rainfall)} \wedge \texttt{FeatureType(A2,rainfall)}\\ \wedge\texttt{Coherent(A1,A2)} \wedge \texttt{TextEntry(A1,X1)} \wedge \texttt{TextEntry(A2,X2)}\\ \Rightarrow \texttt{AddText(Conjunction(X1,X2),P)}\end{array}$$

$$\begin{array}{l}\texttt{Report(R1)} \wedge \texttt{Report(R2)} \wedge \texttt{AtomicFeature(R1,A1)} \wedge \texttt{AtomicFeature(R2,A2)}\\ \wedge\texttt{FeatureType(A1,rainfall)} \wedge \texttt{FeatureType(A2,rainfall)} \wedge \neg\texttt{Coherent(A1,A2)}\\ \wedge\texttt{Source(R1,S1)} \wedge \texttt{Source(R2,S2)} \wedge \texttt{Prefer(S1,S2)}\\ \wedge\texttt{Position(A1,P,R1)} \wedge \texttt{TextEntry(A1,E)}\\ \Rightarrow \texttt{AddText(E,P)}\end{array}$$

$$\begin{array}{l}\texttt{Report(R1)} \wedge \texttt{Report(R2)} \wedge \texttt{IncludeFeature(R1,F1)} \wedge \texttt{IncludeFeature(R2,F2)}\\ \wedge\texttt{FeatureType(F1,regionalreport)} \wedge \texttt{FeatureType(F2,regionalreport)}\\ \wedge\texttt{AtomicFeature(F1,A1)} \wedge \texttt{AtomicFeature(F2,A2)}\\ \wedge\texttt{FeatureType(A1,region)} \wedge \texttt{FeatureType(A2,region)}\\ \wedge\neg\texttt{SameTextEntry(A1,A2)} \wedge \texttt{Position(F1,P,R1)}\\ \Rightarrow \texttt{AddFeature(F1,P)} \wedge \texttt{AddFeature(F2,P)}\end{array}$$

The action formulae give a logical specification that we can reason with. So for example, if we have an action $\neg\alpha$ given by one fusion rule, and we have an action $\alpha \vee \beta$ given by another fusion rule, then taking these together we are obliged to undertake the action β.

Definition 3.12 *The set of* **background formulae** *is formed from the check formulae and the classical universal quantifier, denoted $\forall$, so that any unbound variable is bound by universal quantification. Any subset of the background formulae is called* **background knowledge***.*

Definition 3.13 *A* **fusion system** *is a pair (Δ, Γ) where Γ is a set of fusion rules and Δ is background knowledge.*

We explain how to use a fusion system in the next section.

4 Executing fusion rules

In order to use a set of fusion rules, we need to be able to execute them. We need a fusion system and a pair of structured news report to do this.

Definition 4.1 *A* **fusion call** *is a triple* $(\Delta, \Gamma, \{\texttt{R1}, \texttt{R2}\})$ *where* (Δ, Γ) *is a fusion system, and* R1 *and* R2 *are structured news reports.*

Suppose we want to merge the reports R1 and R2. To do this, we use the background knowledge and the atoms Report(R1) and Report(R2), and then attempt to apply each of the fusion rules by a form of modus ponens, adding the consequent of each applied rule to the current execution state, until no more fusion rules apply.

Definition 4.2 *An* **execution state** *is a subset of* $\mathcal{F}$*.*

An execution state lists the action formulae that hold at each point in an execution of a fusion call.

Definition 4.3 *The* **starting execution state** *for a fusion call* $(\Delta, \Gamma, \{\texttt{R1}, \texttt{R2}\})$ *is* $\{\}$*.*

So at the start of an execution, all we know is the background knowledge, and the two reports. An execution step for a fusion call takes an execution state and a fusion rule and creates a new execution state. The new execution state is the old execution state plus a grounded version of the consequent of one of the fusion rules. For this we need a form of substitution.

Definition 4.4 *A* **substitution** λ *for a fusion rule* $\alpha \Rightarrow \beta$ *is an assignment* λ *of ground terms to variables in* α *and* β *such that* $\lambda(\alpha)$ *and* $\lambda(\beta)$ *are ground formulae.*

Example 4.1 *Consider the first fusion rule in Example 3.14, where* R1 *and* R2 *are grounded by feature terms. A substitution* λ *is*

$$\texttt{R1} = \texttt{weatherreport}(\texttt{date}(\texttt{12.12.01}), \texttt{city}(\texttt{London}), \texttt{weather}(\texttt{rain}))$$
$$\texttt{R2} = \texttt{weatherreport}(\texttt{date}(\texttt{12.12.01}), \texttt{city}(\texttt{London}), \texttt{weather}(\texttt{sleet}))$$

Definition 4.5 *An* **execution step** *for a fusion call* $(\Delta, \Gamma, \{\texttt{R1}, \texttt{R2}\})$ *is a triple* $(X, \alpha \Rightarrow \beta, Y)$ *where* X *is an execution state,* $\alpha \Rightarrow \beta$ *is a fusion rule,* Y *is an execution state, and the following two conditions hold where* λ *is a substitution for* $\alpha \Rightarrow \beta$*:*

1. $\Delta \cup X \cup \{\texttt{Report(R1)}, \texttt{Report(R2)}\} \vdash \lambda(\alpha)$
2. $Y = X \cup \{\lambda(\beta)\}$

Each execution step can be regarded as an application of a form of modus ponens.

Definition 4.6 *An* **execution sequence** *for a fusion call* $(\Delta, \Gamma, \{\texttt{R1}, \texttt{R2}\})$ *is a sequence of execution states* $\langle X_0, .., X_n \rangle$ *where the following conditions hold:*

1. $X_0 = \{\}$

2. *for all* $0 \leq i < n$, $X_i \subset X_{i+1}$

3. *for all* $0 \leq i < n$, *there is an execution step* $(X_i, \alpha \Rightarrow \beta, X_{i+1})$ *for the fusion call* $(\Delta, \Gamma, \{\mathtt{R1}, \mathtt{R2}\})$

4. *there is no execution step* $(X_n, \alpha \Rightarrow \beta, X_{n+1})$ *for the fusion call* $(\Delta, \Gamma, \{\mathtt{R1}, \mathtt{R2}\})$ *such that conditions 1 to 3 hold.*

An execution sequence for a fusion call therefore ensures that: (1) the execution sequence has the starting execution state in the first execution step; (2) each execution step results in an expanded execution state; (3) each execution step follows from the previous step and uses a fusion rule from the fusion system; and (4) the execution sequence is maximal in the sense that it cannot be extended without violating the other conditions.

By definition, an execution sequence is a monotonically increasing sequence of sets. Each X_i in the sequence has one more action formula than the previous set X_{i-1} in the sequence.

Definition 4.7 *An* **action sequence** *for an execution sequence* $\langle X_0, .., X_n\rangle$ *is a sequence of action formulae* $\langle A_1, .., A_n\rangle$ *where for* $0 < i \leq n$, $X_i = X_{i-1} \cup \{A_i\}$.

An action sequence in just the sequence of action formulae that are added to the execution state by each execution step. The action sequence summarizes the actions to be taken to construct the merged news report.

Example 4.2 *Consider the following pair of reports.*

```
⟨weatherreport:                ⟨weatherreport:
     ⟨source: TV1⟩                  ⟨source: TV3⟩
     ⟨date: 19.5.1999⟩              ⟨date: 19.5.1999⟩
     ⟨city: London⟩                 ⟨city: London⟩
     ⟨weather: sun⟩                 ⟨weather: showers⟩
:weatherreport⟩                :weatherreport⟩
```

And a set of fusion rules that includes the following rules:

$\mathtt{Report(R1)} \land \mathtt{Report(R2)} \land \mathtt{SameSkeleton(R1,R2)} \land \mathtt{SameDate(R1,R2)} \land \mathtt{SameCity(R1,R2)}$
$\Rightarrow \mathtt{CreateSkeleton(R1)}$

$\mathtt{Report(R1)} \land \mathtt{Report(R2)} \land \mathtt{AtomicFeature(R1,A1)} \land \mathtt{AtomicFeature(R2,A2)}$
$\land \mathtt{Position(A1,P,R1)} \land \mathtt{FeatureType(A1,source)} \land \mathtt{FeatureType(A2,source)}$
$\land \neg \mathtt{SameSource(R1,R2)} \land \mathtt{TextEntry(A1,E1)} \land \mathtt{TextEntry(A2,E2)}$
$\Rightarrow \mathtt{AddText(Conjunction(E1,E2),P)}$

$\mathtt{Report(R1)} \land \mathtt{Report(R2)} \land \mathtt{AtomicFeature(R1,A1)} \land \mathtt{AtomicFeature(R2,A2)}$
$\land \mathtt{Position(A1,P,R1)} \land \mathtt{FeatureType(A1,date)} \land \mathtt{FeatureType(A2,date)}$
$\land \mathtt{TextEntry(A1,E1)} \land \mathtt{SameTextEntry(A1,A2)}$
$\Rightarrow \mathtt{AddText(E1,P)}$

$$\begin{aligned}&\texttt{Report(R1)} \wedge \texttt{Report(R2)} \wedge \texttt{AtomicFeature(R1,A1)} \wedge \texttt{AtomicFeature(R2,A2)}\\&\wedge \texttt{Position(A1,P,R1)} \wedge \texttt{FeatureType(A1,city)} \wedge \texttt{FeatureType(A2,city)}\\&\wedge \texttt{TextEntry(A1,E1)} \wedge \texttt{SameTextEntry(A1,A2)}\\&\Rightarrow \texttt{AddText(E1,P)}\end{aligned}$$

$$\begin{aligned}&\texttt{Report(R1)} \wedge \texttt{Report(R2)} \wedge \texttt{AtomicFeature(R1,A1)} \wedge \texttt{AtomicFeature(R2,A2)}\\&\wedge \texttt{FeatureType(A1,weather)} \wedge \texttt{FeatureType(A2,weather)} \wedge \neg\texttt{Coherent(A1,A2)}\\&\wedge \texttt{Source(S1,R1)} \wedge \texttt{Source(S2,R2)} \wedge \texttt{Prefer(S1,S2)}\\&\wedge \texttt{Position(A1,P,R1)} \wedge \texttt{TextEntry(A1,E1)}\\&\Rightarrow \texttt{AddText(E1,P)}\end{aligned}$$

A fusion call with these fusion rules and news reports together with appropriate background knowledge can give the following actions:

$$\begin{aligned}A_1 &= \texttt{CreateSkeleton(R1)}\\A_2 &= \texttt{AddText(Conjunction(TV1,TV3),weatherreport/source)}\\A_3 &= \texttt{AddText(19.5.1999,weatherreport/date)}\\A_4 &= \texttt{AddText(London,weatherreport/city)}\\A_5 &= \texttt{AddText(sun,weatherreport/weather)}\end{aligned}$$

The complete action sequence is then given by:

$$\langle A_1, A_2, A_3, A_4, A_5 \rangle$$

In the next section, we consider how we can use an action sequence for constructing a merged structured news report.

5 Acting on fusion rules

Since we are building a merged structured news report in a number of steps, we need to first clarify the nature of the intermediate stages in the construction process. To help, we adopt the following definition of a fusion tree.

Definition 5.1 *A fusion tree is a tree of the form (N, E, S, T, B) where (N, E, S) is a skeleton, T is a set of text entries, and B is a subset of $N \times T$. The set of nodes of the tree is $N \cup T$ and the set of edges of the tree is $E \cup B$.*

So B contains the edges that attach the text entries in T to the skeleton. If a fusion tree (N, E, S, T, B) is a skeleton, then $T = \{\}$ and $B = \{\}$. If a fusion tree is an item of structured text, then T is the set of text entries used in the structured text, and B specifies which atomic features they instantiate. In any case, a fusion tree is a partial instantiation of a skeleton.

Definition 5.2 *A construction sequence $\langle T_1, .., T_n \rangle$ for an action sequence $\langle A_1, .., A_n \rangle$ is a sequence of fusion trees such that*

1. *If $T_i = (N_i, E_i, S_i, T_i, B_i)$, and $T_{i+1} = (N_{i+1}, E_{i+1}, S_{i+1}, T_{i+1}, B_{i+1})$, then $N_i \subseteq N_{i+1}$ and $E_i \subseteq E_{i+1}$ and $S_i \subseteq S_{i+1}$ and $T_i \subseteq T_{i+1}$ and $B_i \subseteq B_{i+1}$.*
2. *T_1 is the result of carrying out A_1 on the fusion tree $(\{\}, \{\}, \{\}, \{\}, \{\})$.*
3. *T_{i+1} is the result of carrying out A_{i+1} on T_i.*

So an action sequence is a sequence of instructions to build a merged structured news report by acting incrementally on a fusion tree. To illustrate, consider the following example.

Example 5.1 *Continuing example 4.2, we have the action sequence where the first instruction is* `CreateSkeleton(R)` *which results in the following fusion tree.*

```
⟨weatherreport:
      ⟨source: ⟩
      ⟨date: ⟩
      ⟨city: ⟩
      ⟨weather: ⟩
:weatherreport⟩
```

The second instruction is `AddText(Conjunction(TV1,TV3),weatherreport/source)` *which updates the above fusion tree to give the following fusion tree.*

```
⟨weatherreport:
      ⟨source: TV1 and TV3⟩
      ⟨date: ⟩
      ⟨city: ⟩
      ⟨weather: ⟩
:weatherreport⟩
```

The third instruction is `AddText(19.5.99,weatherreport/date)` *which updates the above fusion tree to give the following fusion tree.*

```
⟨weatherreport:
      ⟨source: TV1 and TV3⟩
      ⟨date: 19.5.1999⟩
      ⟨city: ⟩
      ⟨weather: ⟩
:weatherreport⟩
```

The fourth instruction is `AddText(London,weatherreport/city)` *which updates the above fusion tree to give the following fusion tree.*

```
⟨weatherreport:
      ⟨source: TV1 and TV3⟩
      ⟨date: 19.5.1999⟩
      ⟨city: London⟩
      ⟨weather: ⟩
:weatherreport⟩
```

The fifth instruction is `AddText(sun,weatherreport/weather)` *which updates the above fusion tree to give the following fusion tree.*

```
⟨weatherreport:
      ⟨source: TV1 and TV3⟩
      ⟨date: 19.5.1999⟩
      ⟨city: London⟩
      ⟨weather: sun⟩
:weatherreport⟩
```

The net result is a merged structured news report.

In Example 5.1, we start by constructing a skeleton, and then adding text entries. So for the fusion tree (N, E, S, T, B), we start by defining (N, E, S), and then incrementally add to T and B until we have a fusion tree that defines an item of structured text. In the next example, we form a merged structured news report from some complex features.

Example 5.2 *Consider* `R1` *being the first report given in Example 1.2. Now consider the action sequence* $\langle A_1, ..., A_4 \rangle$ *where*

$$\begin{aligned}
A_1 &= \mathtt{CreateSkeleton(R1)} \\
A_2 &= \mathtt{AddText(5\,Nov\,99, weatherreport/date)} \\
A_3 &= \mathtt{Populate(F1, weatherreport/regionalreport)} \\
A_4 &= \mathtt{AddFeature(F2, weatherreport)}
\end{aligned}$$

where

$$\begin{aligned}
\mathtt{F1} &= \mathtt{regionalreport(region(South\,East), maxtemp(20C))} \\
\mathtt{F2} &= \mathtt{regionalreport(region(North\,West), maxtemp(18C))}
\end{aligned}$$

For the first instruction, `CreateSkeleton(R1)`*, we get the following fusion tree:*

$$\begin{array}{l}
\langle \mathtt{weatherreport} : \\
\quad \langle \mathtt{date} : \rangle \\
\quad \langle \mathtt{regionalreport} : \\
\quad\quad \langle \mathtt{region} : \rangle \\
\quad\quad \langle \mathtt{maxtemp} : \rangle \\
\quad : \mathtt{regionalreport} \rangle \\
: \mathtt{weatherreport} \rangle
\end{array}$$

For the second instruction, `AddText(5 Nov 99, weatherreport/date)`*, we update the above fusion tree to get the following:*

$$\begin{array}{l}
\langle \mathtt{weatherreport} : \\
\quad \langle \mathtt{date} : \mathtt{5\,Nov\,99} \rangle \\
\quad \langle \mathtt{regionalreport} : \\
\quad\quad \langle \mathtt{region} : \rangle \\
\quad\quad \langle \mathtt{maxtemp} : \rangle \\
\quad : \mathtt{regionalreport} \rangle \\
: \mathtt{weatherreport} \rangle
\end{array}$$

For the third instruction, `Populate(F1,weatherreport/regionalreport)` *we update the above fusion tree to get the following:*

$$\begin{array}{l}
\langle \mathtt{weatherreport} : \\
\quad \langle \mathtt{date} : \mathtt{5\,Nov\,99} \rangle \\
\quad \langle \mathtt{regionalreport} : \\
\quad\quad \langle \mathtt{region} : \mathtt{South\,East} \rangle \\
\quad\quad \langle \mathtt{maxtemp} : \mathtt{20C} \rangle \\
\quad : \mathtt{regionalreport} \rangle \\
: \mathtt{weatherreport} \rangle
\end{array}$$

For the fourth instruction, `AddFeature(F2,weatherreport)` *we update the above fusion tree to* *get the following:*

$$
\begin{array}{l}
\langle \texttt{weatherreport}: \\
\quad \langle \texttt{date}: \texttt{5 Nov 99} \rangle \\
\quad \langle \texttt{regionalreport}: \\
\quad\quad \langle \texttt{region}: \texttt{South East} \rangle \\
\quad\quad \langle \texttt{maxtemp}: \texttt{20C} \rangle \\
\quad : \texttt{regionalreport} \rangle \\
\quad \langle \texttt{regionalreport}: \\
\quad\quad \langle \texttt{region}: \texttt{North West} \rangle \\
\quad\quad \langle \texttt{maxtemp}: \texttt{18C} \rangle \\
\quad : \texttt{regionalreport} \rangle \\
: \texttt{weatherreport} \rangle
\end{array}
$$

If an action sequence is non-conflicting and complete, then the set of instructions can be used to build a fusion tree and they leave no gaps in the text entries in the fusion tree. If the action sequence is incomplete, then the fusion tree will have text entries missing, and if the action sequence is conflicting then there will be instructions for putting more than one text entry into the same position or instructions for putting both a text entry and a complex feature into the same position. Before defining when an action sequence is complete and non-conflicting, we consider when an action sequence is consistent.

Definition 5.3 *An action sequence* $\langle A_1, .., A_n \rangle$ *is* **consistent** *iff* $\{A_1, .., A_n\} \not\vdash \bot$.

This definition takes a direct interpretation of consistent. It just means an action sequence is consistent if there is not an instruction to do both an action α and an action $\neg\alpha$. An action sequence can be checked for consistency before an attempt to construct a merged report is made.

Since, an action sequence is a specification for a merged structured news report, we can determine whether a particular structured news report meets the specification. We define this as follows:

Definition 5.4 *The* **meets** *relation is a binary relation between items of structured text and action sequences, and is defined as follows:*

$$\texttt{R} \textit{ meets } \langle A_1, ..., A_n \rangle \textit{ iff } \texttt{R} \textit{ meets } A_1 \textit{ and .... and } \texttt{R} \textit{ meets } A_n$$

So by recursion, we need to consider the meets relation for action formulae. For action formulae A_i *that are atoms, we require the following rules:*

$$\texttt{R} \textit{ meets } \texttt{CreateSkeleton}(\texttt{R}') \textit{ if } Skeleton(\texttt{R}') \preceq Skeleton(\texttt{R}) \textit{ holds}$$

$$\texttt{R} \textit{ meets } \texttt{AddText}(\texttt{E},\texttt{P}) \textit{ if } \exists \texttt{A} \textit{ s.t. } \texttt{TextEntry(A,E)} \wedge \texttt{Position(A,P,R)} \textit{ holds}$$

$$\texttt{R} \textit{ meets } \texttt{AddFeature}(\texttt{F},\texttt{P}) \textit{ if } \texttt{Anchor(F,P,R)} \textit{ holds}$$

$$\texttt{R} \textit{ meets } \texttt{ExtendFeature}(\texttt{F},\texttt{P}) \textit{ if } \texttt{Position(F,P,R)} \textit{ holds}$$

$$\texttt{R} \textit{ meets } \texttt{Populate}(\texttt{F},\texttt{P}) \textit{ if } \texttt{Position(F,P,R)} \textit{ holds}$$

For action formulae A_i *that are not atoms, we require the following rules:*

$$
\begin{array}{c}
\texttt{R} \textit{ meets } \alpha \vee \beta \textit{ iff } \texttt{R} \textit{ meets } \alpha \textit{ or } \texttt{R} \textit{ meets } \beta \\
\texttt{R} \textit{ meets } \alpha \wedge \beta \textit{ iff } \texttt{R} \textit{ meets } \alpha \textit{ and } \texttt{R} \textit{ meets } \beta \\
\texttt{R} \textit{ meets } \neg\alpha \textit{ iff it is not the case that } \texttt{R} \textit{ meets } \alpha
\end{array}
$$

Definition 5.5 *An action sequence* $\langle A_1, ..., A_n\rangle$ *is* **non-conflicting** *iff there is a construction sequence* $\langle T_1, ..., T_n\rangle$ *for* $\langle A_1, .., A_n\rangle$ *and* T_n *meets* $\langle A_1, .., A_n\rangle$.

However, the meets relation is a little too relaxed in the sense that a report may meet an action sequence but may also include extra information that has not been specified.

Definition 5.6 *The* **matches** *relation is defined as follows where* `R` *is a structured news report and* $\langle A_1, ..., A_n\rangle$ *is an action sequence.*

> `R` *matches* $\langle A_1, ..., A_n\rangle$
> *iff*
> `R` *meets* $\langle A_1, ..., A_n\rangle$ *and* $\nexists$`R'` *s.t.* `R'` *meets* $\langle A_1, ..., A_n\rangle$ *and* $Skeleton(\mathtt{R}') \prec Skeleton(\mathtt{R})$

The matches relation identifies the minimal structured news report(s) that meet(s) the action sequence. In other words, it identifies the news reports that do not include any superfluous information.

Definition 5.7 *An action sequence* $\langle A_1, ..., A_n\rangle$ *is* **unambiguous** *iff there is only one construction sequence* $\langle T_1, .., T_n\rangle$ *for* $\langle A_1, .., A_n\rangle$

If an action sequence $\langle A_1, .., A_n\rangle$ is unambiguous, there is exactly one structured news report `R` such that `R` matches $\langle A_1, .., A_n\rangle$.

Definition 5.8 *An action sequence* $\langle A_1, .., A_n\rangle$ *is* **complete** *iff the construction sequence* $\langle T_1, .., T_n\rangle$ *for* $\langle A_1, .., A_n\rangle$ *is such that* T_n *is a structured news report.*

In other words, an action sequence is a complete if it is not the case that the fusion tree that results has missing text entries.

6 Properties of fusion rules

We now consider a few properties of fusion rules to clarify the nature of the syntax and execution.

Proposition 6.1 *Assuming the actions are only composed from the action atoms defined in Definition 3.7. An action sequence* $\langle A_1, .., A_n\rangle$ *is non-conflicting implies the following conditions:*

1. $\neg\exists$ `P` *s.t.* $\{A_1, ..., A_n\} \vdash$ `AddText(E,P)` $\wedge$ `AddText(E',P)` $\wedge$ `E` $\neq$ `E'`
2. $\neg\exists$ `P` *s.t.* $\{A_1, ..., A_n\} \vdash$ `AddText(E,P)` $\wedge$ `ExtendFeature(F,P)`
3. $\neg\exists$ `P` *s.t.* $\{A_1, ..., A_n\} \vdash$ `AddText(E,P)` $\wedge$ `AddFeature(F,P)`

Proof: Consider condition 1. The result of `AddText(E,P)` is a fusion tree with `E` being the text entry for the atomic feature at `P`. By the definition of atomic features, there can only be one text entry at `P`. So it is not possible to have both `E` and `E'` at `P` when $\mathtt{E} \neq \mathtt{E}'$. So there is no construction sequence $\langle T_1, .., T_n\rangle$ for $\langle A_1, .., A_n\rangle$ where T_n meets $\langle A_1, .., A_n\rangle$. The cases for Conditions 2 and 3 are essentially the same.

A fusion call does not necessarily produce a unique action sequence. In other words, normally there is some non-determinism in which fusion rules are applied.

Proposition 6.2 *Let* $(\Delta, \Gamma, \{\mathtt{R1}, \mathtt{R2}\})$ *be a fusion call. It is not necessarily the case that there is a unique action sequence* $\langle A_1, .., A_n\rangle$ *that is generated.*

Proof: Consider Example 4.2, in which the action sequence $\langle A_1, .., A_5\rangle$ is generated. This fusion call could equally generate the following action sequence $\langle A'_1, .., A'_5\rangle$, where:

$$A'_1 = \mathtt{CreateSkeleton(R1)}$$
$$A'_2 = \mathtt{AddText(sun, weatherreport/weather)}$$
$$A'_3 = \mathtt{AddText(London, weatherreport/city)}$$
$$A'_4 = \mathtt{AddText(19.5.1999, weatherreport/date)}$$
$$A'_5 = \mathtt{AddText(Conjunction(TV1, TV3), weatherreport/source)}$$

There can also be some non-determinism in an action sequence.

Proposition 6.3 *Let* $\langle A_1, .., A_n\rangle$ *be an action sequence, and let* A_i *be an action formula in that sequence. If* A_i *can be rewritten into disjunctive normal form, so* A_i *is equivalent to a formula* $\alpha_1 \vee .. \vee \alpha_k$*, then there may be more than one structured news report* R *such that* R *matches* $\langle A_1, .., A_n\rangle$.

Proof: Consider a report R1 where R1 matches $\langle A_1, .., A_n\rangle$ and a report R2 where R2 matches $\langle A_1, .., A_n\rangle$. So for each A_i, R1 meets A_i. Now consider an A_i of the form of $\alpha \vee \beta$. Clearly R1 meets $\alpha \vee \beta$ and R2 meets $\alpha \vee \beta$. But suppose, R1 meets α and R1 does not meet β. Also suppose, R2 does not meet α and R2 meets β. So, $\mathtt{R1} \neq \mathtt{R2}$.

The length of an execution sequence, i.e. the number of execution steps undertaken for a fusion call is constrained by the number of fusion rules, and the nature of the inferences from the domain knowledge, and the size of the structured news reports. In order to get a useful boundary on the length of an execution sequence, we adopt the following definition.

Definition 6.1 *A fusion rule* $\alpha \Rightarrow \beta$ *is capped iff the only possible substitutions* λ *are such that* λ *assigns feature terms to the free variables in* β.

All the fusion rules in this paper are capped.

Proposition 6.4 *For any fusion call* $(\Delta, \Gamma, \{\mathtt{R1}, \mathtt{R2}\})$*, if* $|\Gamma|$ *is finite, and each fusion rule is capped, then the execution sequence* $\langle X_1, .., X_n\rangle$ *is finite.*

Proof: The constraints in Definition 4.6 ensure that there are no cycles in an execution sequence. So there is no execution sequence $\langle X_1, .., X_n\rangle$ such that there is an X_i and X_j where $X_i = X_j$ unless $i = j$. Hence, there is no sequence of execution steps where the same instantiated form of a fusion rule is used twice. The only way that we can get an infinite sequence $\langle X_1, .., X_n\rangle$ is if there are infinitely many $\lambda(\beta)$ generated for some rule $\alpha \Rightarrow \beta \in \Gamma$. However, if each fusion rule in Γ is capped, then there are only finitely many $\lambda(\beta)$ that can be generated for each fusion rule, since the only substitutions for the variables in β come from the feature terms generated from the structured news reports in the fusion call, and there are only finitely many feature terms that can be generated from each fusion call. So it is not possible to generate an infinite execution sequence $\langle A_1, .., A_n\rangle$.

The assumption that the fusion rules are capped seems quite reasonable if the aim of fusion is to only include information from the original reports being merged. Indeed if we assume the rules are capped, we can identify a tighter bound based directly on the size of the structured reports being merged.

However, the computational viability of a fusion system depends on more than the number of execution steps taken. Indeed there are a number of factors that need to be considered:

- executing fusion rules
- reasoning with background knowledge
- acting on fusion rules

We can regard each of these activities being a form of classical logic inferencing, and hence the computational viability is bounded by the computational viability of classical logic. Whilst in general, reasoning with classical logic is difficult to automate, implementation based on Prolog is feasible.

Another practical question is whether the syntax can express everything that we want or need to express. This includes:

- Location completeness of structural atoms. This is the ability to describe any structural relationship in a report in terms of the nesting and sequence of semantic labels and text entries.
- Comparison completeness of background atoms. This is the ability to compare any combination of text entries with respect to the background knowledge. Clearly the background atoms presented are only indicative of the possible atoms that may be defined for an application.
- Fusion completeness of action atoms. This is the ability to describe how any structured news report can be constructed. In one sense, the current set of action atoms is sufficient for this. However, further actions atoms would allow reports to be constructed with fewer instructions. For example, currently the action atoms cannot directly specify the sequence in which siblings occur.

The notion of structured news reports that we reviewed in Section 2 provides a rich structural representation. The check and action atoms that we define in Section 3, do not draw on the full expressivity of structured new report. However, it is straightforward to add further check and action atoms to extend the basic fusion framework that we have presented here. To illustrate, we could introduce the following actions.

Definition 6.2 *Further action atoms include:*

1. `LeftAdd(F,P)` *where* `F` *is a feature and* `P` *is a position. The resulting action is to add* `F` *to the fusion tree to the left of the existing feature at* `P`*.*
2. `RightAdd(F,P)` *where* `F` *is a feature and* `P` *is a position. The resulting action is to add* `F` *to the fusion tree to the right of the existing feature at* `P`*.*

To use these actions, we also introduce the following structural atoms.

Definition 6.3 *Further structural atoms include:*

1. `LeftNeighbour(F',F,P,R)` *where* `F'` *and* `F` *are features with the anchor at position* `P` *in* `R` *and* `F'` *is immediately to the left of* `F`.

2. `RightNeighbour(F,F',P,R)` *where* `F'` *and* `F` *are features with the anchor at position* `P` *in* `R` *and* `F'` *is immediately to the right of* `F`.

We illustrate the first of these structural atoms below.

Example 6.1 *Let* `R` *be the news report given in Figure 1. Also consider the following:*

$$\begin{array}{c}\texttt{F'} = \texttt{source}(\texttt{Orange website})\\ \texttt{F} = \texttt{URL}(\texttt{www.orange.co.uk})\\ \texttt{P} = \texttt{bidreport/reportinfo}\end{array}$$

For this report, `LeftNeighbour(F',F,P,R)` *holds.*

Now we illustrate the action atoms given in Definition 6.2.

Example 6.2 *Consider the following fusion tree* `T`*.*

$$\begin{array}{l}\langle\texttt{weatherreport}:\\ \quad\quad\langle\texttt{date}:\texttt{5 Nov 99}\rangle\\ \quad\quad\langle\texttt{regionalreport}:\\ \quad\quad\quad\quad\langle\texttt{region}:\texttt{South East}\rangle\\ \quad\quad\quad\quad\langle\texttt{maxtemp}:\texttt{20C}\rangle\\ \quad\quad:\texttt{regionalreport}\rangle\\ :\texttt{weatherreport}\rangle\end{array}$$

Now consider the following instructions:

$$\begin{array}{l}A = \texttt{LeftAdd}(\texttt{F}, \texttt{weatherreport/regionalreport})\\ A' = \texttt{RightAdd}(\texttt{F}, \texttt{weatherreport/regionalreport})\end{array}$$

where

$$\texttt{F} = \texttt{regionalreport}(\texttt{region}(\texttt{North West}), \texttt{maxtemp}(\texttt{18C}))$$

For the instruction, `LeftAdd(F,weatherreport/regionalreport)`*, applied to the fusion tree* `T`*, we get the following:*

$$\begin{array}{l}\langle\texttt{weatherreport}:\\ \quad\quad\langle\texttt{date}:\texttt{5 Nov 99}\rangle\\ \quad\quad\langle\texttt{regionalreport}:\\ \quad\quad\quad\quad\langle\texttt{region}:\texttt{North West}\rangle\\ \quad\quad\quad\quad\langle\texttt{maxtemp}:\texttt{18C}\rangle\\ \quad\quad:\texttt{regionalreport}\rangle\\ \quad\quad\langle\texttt{regionalreport}:\\ \quad\quad\quad\quad\langle\texttt{region}:\texttt{South East}\rangle\\ \quad\quad\quad\quad\langle\texttt{maxtemp}:\texttt{20C}\rangle\\ \quad\quad:\texttt{regionalreport}\rangle\\ :\texttt{weatherreport}\rangle\end{array}$$

But suppose we ignore the previous instruction, and return to the original state `T` *of the fusion tree. For the instruction* `RightAdd(F,weatherreport/regionalreport)`*, applied to the fusion tree* `T`*,* *we get the following:*

$$
\begin{array}{l}
\langle \texttt{weatherreport} : \\
\quad \langle \texttt{date} : \texttt{5 Nov 99} \rangle \\
\quad \langle \texttt{regionalreport} : \\
\quad\quad \langle \texttt{region} : \texttt{South East} \rangle \\
\quad\quad \langle \texttt{maxtemp} : \texttt{20C} \rangle \\
\quad : \texttt{regionalreport} \rangle \\
\quad \langle \texttt{regionalreport} : \\
\quad\quad \langle \texttt{region} : \texttt{North West} \rangle \\
\quad\quad \langle \texttt{maxtemp} : \texttt{18C} \rangle \\
\quad : \texttt{regionalreport} \rangle \\
: \texttt{weatherreport} \rangle
\end{array}
$$

It is straightforward to extend the framework that we have presented in this paper to accommodate these atoms.

Clearly, we can greatly extend the set of background atoms depending on the application. Some further discussion of formulae for the background knowledge including discussion of inconsistency, temporal knowledge, and domain knowledge is given in [Hun00a, Hun02b], and for further discussion of lexical and world knowledge see also [Hun01, HM99, Hun96]. Also of relevance are the options of using ontologies for structured text (see for example [ES01]) and using comprehensive semantic networks such as WordNet [Mil95]. More generally, it may be appropriate to harness typed-feature structures [Car92] and machine readable dictionaries [WSG96] for representing and reasoning with lexical 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.

We have proposed fusion rules as a scripting language for defining how to merge news reports. It may be appropriate to develop syntactic sugar and other notational conveniences to enhance this proposal. This may include using symbols such as AND, OR, and NOT. It may also include priority ordering over fusion rules to dictate the preferred ordering in which they should apply so as to allow for simpler antecedents. Further assumptions could also be used about the process to simplify the notation. For example, each rule has $\texttt{Report(R1)} \wedge \texttt{Report(R2)}$ in the antecedent, and yet we may assume two reports which are always referred to as `R1` and `R2`, and thereby not need $\texttt{Report(R1)} \wedge \texttt{Report(R2)}$ in the antecedent.

The definition for a fusion call suggests an implementation based on existing automated reasoning technology and on XML programming technology. The most obvious route for representing each structured news report is to represent it as an XML document. Once information is in the form of XML documents, a number of technologies for managing and manipulating information in XML are available [Bra00]. Possibilities for representing and reasoning with background knowledge include relational databases, Datalog, and Prolog. Possibilities for implementing an inference engine for executing fusion rules include a meta-level program in Prolog, or an implementation in a imperative programming language such as Java. Another possibility is to present fusion rules in RuleML and harness one of the Java rule engines that are currently being proposed[1]. Finally, an action engine for acting on the instructions given by the fusion rules, could be implemented in an imperative programming language such as Java that can manipulate XML. An action engine would need to take each instruction in an action sequence and construct a construction sequence.

We have not formalized the relationship between the structural atoms and XML technology. However, there is clearly an overlap in functionality with technologies including the XPath language and proposals for the XML Query Algebra[2]. However, the main points we want to stress in any comparison is that: (1) fusion rules offer a logical bridge between logical reasoning with background knowledge, structural information about news reports to be merged, and logical specifications of the instructions for producing the merged report; and (2) fusion rules offer a higher-level scripting language for handling structured text than available with XPath or XML Query Algebra, and so fusion rules can be used on top of XML technology. In this sense, fusion rules and XML technology are complementary.

Given that information extraction may be the technology for providing structured news reports for merging, integration of a fusion system with information extraction technology may be appropriate. The GATE System provides an implemented architecture for managing textual data storage and exchange, visualization of textual data structures, and plug-in modularity of text processing components [GCW+96]. The text processing components includes LaSIE which performs information extraction tasks including named entity recognition, coreference resolution, template element filling, and scenario template filling.

References

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

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

[BCD+93] S Benferhat, C Cayrol, D Dubois, J Lang, and H Prade. Inconsistency management and prioritized syntax-based entailment. In *Proceedings of the 13th International Joint Conference on AI (IJCAI'93)*, 1993.

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

[BDP95] S Benferhat, D Dubois, and H Prade. How to infer from inconsistent beliefs without revising. In *Proceedings of the 14th International Joint Conference on AI (IJCAI'95)*, 1995.

[BH01] Ph Besnard and A Hunter. A logic-based theory of deductive arguments. *Artificial Intelligence*, 128:203–235, 2001.

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

[Bra00] N Bradley. *The XML Companion*. AddisonWesley, 2000.

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

[1] www.dfki.uni-kl.de/ruleml
[2] www.w3.org

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

[Car92] B Carpenter. *The Logic of Typed Feature Structures.* Cambridge University Press, 1992.

[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.

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

[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.

[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.

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

[ES01] M Erdmann and R Studer. How to structure and access XML documents with ontologies. *Data and Knowledge Engineering*, 36:317–335, 2001.

[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.

[GCW+96] R Gaizaukas, H Cunningham, Y Wilks, P Rodgers, and K Humphreys. GATE:an environment to support research and development in natural language engineering. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence.* IEEE Computer Society Press, 1996.

[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.

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

[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.

[HM99] A Hunter and L Marten. Context-sensitive reasoning with lexical and world knowledge. In *SOAS Working Papers in Linguisitcs*, volume 9, pages 80–95, 1999.

[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.

[Hun00a] A Hunter. Merging potentially inconsistent items of structured text. *Data and Knowledge Engineering*, 34:305–332, 2000.

[Hun00b] A Hunter. Ramification analysis using causal mapping. *Data and Knowledge Engineering*, 32:1–27, 2000.

[Hun00c] A Hunter. Reasoning with inconsistency in structured text. *Knowledge Engineering Review*, 15:317–337, 2000.

[Hun01] A Hunter. A default logic-based framework for context-dependent reasoning with lexical knowledge. *Journal of Intelligent Information Systems*, 16:65–87, 2001.

[Hun02a] A Hunter. Hybrid argumentation systems for structured news reports. *Knowledge Engineering Review*, 2002. (in press).

[Hun02b] A Hunter. Merging structured text using temporal knowledge. *Data and Knowledge Engineering*, 2002. (in press).

[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.

[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.

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

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

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

[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 datasources 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.

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