

Man Bites Dog: Looking for Interesting Inconsistencies in Structured News Reports*

Emma Byrne and Anthony Hunter[†]
Department of Computer Science
University College London
Gower Street
London WC1E 6BT, UK
[e.byrne | a.hunter] @cs.ucl.ac.uk

May 29, 2003

Keywords: Analysing inconsistency; Measuring inconsistency; Expectation;
Contradiction; Argumentation; Defeasible knowledge; Default knowledge

Abstract

Much useful information in news reports is often that which is surprising or unexpected. In other words, we harbour many expectations about the world, and when any of these expectation are violated (i.e. made inconsistent) by news, we have a strong indicator of some information that is interesting for us. In this paper we present a framework for identifying interesting information in news reports by finding interesting inconsistencies. An implemented system based on this framework (1) accepts structured news reports as inputs, (2) translates each report to a logical literal, (3) identifies the story of which the report is a part, (4) looks for inconsistencies between the report, the background knowledge, and a set of expectations, (5) classifies and evaluates those inconsistencies, and (6) outputs news reports of interest to the user together with associated explanations of why they are interesting.

1 Introduction

As in the adage, “Man Bites Dog” is interesting news; “Dog Bites Man” is not. However, the majority of information published is of the “Dog Bites Man” variety, particularly in the case of business, technical, and scientific news. This information continues existing trends or confirms expected information. Unexpected information is often more useful to the reader. Identifying such information is beyond the current capabilities of traditional information management technologies.

We look for interesting news based on the expectations we have about the world. If a news report violates an expectation, this indicates the news is interesting. Furthermore, if we represent each expectation as a classical logic formula, and the information in a news report as a classical logic formula, together with

*This work is supported by EPSRC grant (GR/R22551/01) Logic-based Technology for Measuring Inconsistency of News Reports.

[†]Corresponding author

various kinds of background knowledge as classical logic formulae, then the violation of an expectation by a news report can be characterized as a type of logical inconsistency.

It is straightforward to find expectations for a focussed domain. For example, data mining is a way of generating general rules from databases, and some of these can be exploited as expectations. As another example, in the business domain, investments analysts regularly publish expectations about the activities and performance of companies listed on stock markets, and these expectations can be rephrased as logical formulae. Expectations will come from a variety of sources including knowledge engineering, data mining, and expert heuristics.

In order to manipulate news reports in this framework, we assume each report is provided in the form of structured text. This is an idea implicit in a number of approaches to handling information such as news reports. An item of structured text is a set of semantic labels together with a word, phrase, sentence, null value, or a nested item of structured text, associated with each semantic label. As a simple example, a report on a corporate acquisition could use semantic labels such as “buyer”, “seller”, “acquisition”, “value”, and “date”. Some news agencies store news reports as structured text. In addition technologies such as information extraction [CL96] will massively increase the amount of structured text available.

Unexpected information is not the only cause of inconsistency in news reports so it is necessary to differentiate between inconsistencies that are indicators of interesting news and inconsistencies that arise through errors or other “uninteresting” conflicts. Inconsistencies can be classified as one of the following categories: (1) Typographical errors; (2) Superficial errors arising from different words or phrases used for the same thing; (3) Content errors in the report; (4) Errors in the domain knowledge; and (5) Violations of expectations.

Whilst violations of expectations are interesting to the client, most inconsistencies arise through errors in the report or at the information extraction stage, or are superficial inconsistencies. Even those inconsistencies that are of no interest to the user may be of interest to the system. They may provide the trigger to correct a report, update the knowledge base or reassess expectations. In addition, uninteresting inconsistencies may mask more interesting ones. We believe that no inconsistency should be automatically ignored; it must be classified and then acted upon (even if that action is to make the decision to undertake no further action)¹. However, in this report we assume that the only inconsistencies arising are violations of expectations. This is a reasonable assumption if we eliminate the other kinds of inconsistency by harnessing spell checkers, integrity checkers, and various other modules to preprocess the structured news reports, and to maintain the background knowledge.

Clearly, news reports do not normally exist in isolation. They are usually part of “narratives” which relate them to other articles that deal with the same story. For example, in the mergers and acquisitions domain, we may find a news report announcing a takeover bid followed by a news report of the bid being accepted by the board, followed by a report on the shareholders voting whether to accept the bid, and so on. All news reports belong to at least one narrative. Equivalently, each narrative consists of a set of one or more reports. To address this need, we use event models based on a variant of event calculus to represent and reason with such information.

In this paper we present a logic-based framework called the “Man Bites Dog” (MBD) framework for characterizing violations of expectations by unexpected information in structured news reports. An implemented system based on the framework is called an MBD system.

The input to an MBD system is a report in structured text format. As such, all issues concerning information extraction are handled upstream by an external information extraction system ([CL96]) or by the manual creation of structured texts. The process of an MBD system is then: (1) Translate each incoming news report into logic; (2) Identify the events featured in each report; (3) Identify inconsistencies between each report, the relevant background knowledge, and a repository of expectations; (4) Evaluate each inconsistency for

¹We are therefore adopting the approach of “Inconsistency Implies Action” as proposed in [GH91].

significance; and (5) Output each report highlighting violations of expectations. The activities will be the same for a wide variety of domains but the background knowledgebase needs to be tailored for each domain.

The rest of the paper is laid out as follows. In Section 2, we review some basic definitions for classical logic. In Section 3, we formalize the notion of structured news reports, define how they can be represented as logical literals, and then present two kinds of background knowledge represented as classical logic formulae: (1) Access rules for accessing information encoded in news reports; and (2) Domain facts. In Section 4, we present a framework for representing and reasoning with events referred to in news reports. Then in Section 5, we formalize the notion of expectations, and the notion of violating expectations presented as a form of inconsistency. In Section 6, we present a framework for evaluating these inconsistencies. This is a confirmation theoretic approach to assessing each expectation. Violations of better confirmed expectations are interpreted as more significant, and therefore more interesting. Of course this is only one of a number of possible interpretations, but it is a potentially viable and useful interpretation. Finally, in Section 7, we address the computational complexity problem of consistency checking by presenting an approach to compilation of consistency checking.

2 Basic definitions

We assume the usual language of classical first-order logic using the usual symbols \forall and \exists for quantification and the usual symbols \wedge , \vee , \rightarrow and \neg for logical connectives and sets of predicate symbols, function symbols, constant symbols and variable symbols.

Definition 2.1 Let \mathcal{F} be a set of function symbols, \mathcal{C} be a set of constant symbols and \mathcal{V} be a set of variable symbols. If $c \in \mathcal{C}$ then c is a term. If $v \in \mathcal{V}$ then v is a term. If $f \in \mathcal{F}$ and t_1, \dots, t_n are terms then $f(t_1, \dots, t_n)$ is a term. For all terms t , if t incorporates a variable, then t is an **unground term**, otherwise t is **ground term**. If $f(t_1, \dots, t_n)$ is a term, then t_1 is a subterm, ... and t_n is a subterm. If t is a term, then $t \in \text{Subterms}(t)$, and for any term $t' \in \text{Subterms}(t)$, the subterms of t' are in $\text{Subterms}(t)$

Definition 2.2 Let \mathcal{P} be a set of predicate symbols, and \mathcal{T} be a set of terms. If $t_1, \dots, t_n \in \mathcal{T}$ and $p \in \mathcal{P}$ then $p(t_1, \dots, t_n)$ is a **predicate**. If $p(t_1, \dots, t_n)$ is a predicate and t_1 is a ground term and ... and t_n is a ground term, then $p(t_1, \dots, t_n)$ is a **ground predicate**, otherwise it is an **unground predicate**. An **atom** is a ground predicate. A **literal** is either a predicate (i.e. a **positive literal**) or the negation of a predicate (i.e. a **negative literal**).

Definition 2.3 If α is a literal, then $\text{Terms}(\alpha)$ is the set of terms in α as follows.

$$\begin{aligned} \text{Terms}(p(t_1, \dots, t_n)) &= \text{Subterms}(t_1) \cup \dots \cup \text{Subterms}(t_n) \\ \text{Terms}(\neg p(t_1, \dots, t_n)) &= \text{Subterms}(t_1) \cup \dots \cup \text{Subterms}(t_n) \\ \text{Terms}(\{\alpha_1, \dots, \alpha_n\}) &= \text{Terms}(\alpha_1) \cup \dots \cup \text{Terms}(\alpha_n) \end{aligned}$$

Definition 2.4 A **grounding** is an equality predicate where the first argument is a variable and the second argument is a ground term. A **grounding set** is a set of groundings which can be substituted into an unground term to give a grounded term. Let α be a formula and let Φ be a grounding set. $\text{Ground}(\alpha, \Phi)$ gives the results of substituting each variable X in α with term t where the grounding $X = t$ is in Φ .

Example 2.1 Let $a(b(X), c(Y), d(Z))$ be a formula where X, Y and Z are variables. Let the grounding set Φ be $\{Y = \text{john}, Z = \text{betty}\}$.

$$\text{Ground}(a(b(X), c(Y), d(Z)), \Phi) = a(b(X), c(\text{john}), d(\text{betty}))$$

We assume \vdash denotes the classical consequence relation.

```

<weatherreport>
  <date> 23 April 1999 </date>
  <location>
    <city> London </city>
    <country> UK </country>
  </location>
  <today> cold </today>
  <tomorrow> sunny </tomorrow>
  <log> 23 April 1999 </log>
</weatherreport>

```

Figure 1: An example of a structured news report.

3 News reports and background knowledge

First we consider structured news reports and then discuss how they can be represented as logical literals. In order to support reasoning with information in news reports, we require two kinds of background knowledge: (1) Access rules for accessing information encoded in structured news reports; and (2) Domain facts.

3.1 Structured news reports

We use XML to represent structured news reports. An example is given in Figure 1. Each structured news report is an XML document, but not vice versa, as defined below.

Definition 3.1 *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 $\sigma_1, \dots, \sigma_n$ are structured news reports, then $\langle\phi\rangle\sigma_1\dots\sigma_n\langle/\phi\rangle$ is a structured news report.*

Clearly each structured news report is isomorphic to a tree with each tagname being represented by a non-leaf node and each textentry being represented by a leaf node. Furthermore, each subtree in a structured news report is isomorphic to a ground term where each tagname is represented by a function symbol and each textentry is represented by a constant symbol. Hence, we can represent each structured news report by a ground logical atom in classical logic as follows.

Definition 3.2 *Let $\langle\phi\rangle\sigma_1, \dots, \sigma_n\langle/\phi\rangle$ be a structured news report. A **report atom** is ground atom of the form $\phi(t_1, \dots, t_n)$ where t_1 is a ground term that represents σ_1, \dots , and t_n is a ground term that represents σ_n .*

Example 3.1 *Consider the following structured news report.*

```

<auctionreport>
  <buyer><name><firstname>John</firstname><surname>Smith</surname></name></buyer>,
  <property>Lot37</property>
</auctionreport>

```

This can be represented by the following report atom:

```

auctionreport(buyer(name(firstname(John), surname(Smith))), property(Lot37))

```

Each structured news report is isomorphic to a report atom and the semantic label of the root of a structured news report is the predicate symbol of the corresponding report atom. In the rest of this paper, the root of the structured news report in each example, and the predicate symbol of the corresponding report atom, is the symbol report.

3.2 News atoms and access rules

News atoms are atoms that capture information encoded in report atoms. These predicates can then be directly used with other information in the knowledgebase.

Definition 3.3 Let $\phi(t_1, \dots, t_n)$ be a report atom. A **news atom** for this report is a literal of the form $\beta(s_1, \dots, s_m)$ where β is a predicate symbol and s_1, \dots, s_m are in $\text{Subterms}(t_1) \cup \dots \cup \text{Subterms}(t_n)$.

The news atoms that we obtain for a given report atom are defined by the access rules as follows.

Definition 3.4 An **access rule** is a first order formula of the form where: (1) X_1, \dots, X_k are the variables in α ; (2) the variables in β_1, \dots, β_n are a subset of, or equal to, X_1, \dots, X_k ; (3) all of $\alpha, \beta_1, \dots, \beta_n$ are unquantified; and (4) the predicate symbol for α is not used as a predicate symbol for any of β_1, \dots, β_n .

$$\forall X_1, \dots, X_k; \alpha \rightarrow \beta_1 \wedge \dots \wedge \beta_n$$

If there is a grounding set Φ s.t. $\text{Ground}(\alpha, \Phi)$ is a report atom then $\text{Ground}(\beta_1, \Phi), \dots, \text{Ground}(\beta_n, \Phi)$ are news atoms.

Definition 3.5 Let Δ be a set of access rules and let R be a report. For this, $\text{Access}(\Delta, R)$ is the smallest set of news atoms obtained by exhaustively applying the news atom for report R to the access rules in Δ using generalized modus ponens and conjunction elimination as defined below.

$$\text{Access}(\Delta, R) = \{ \text{Ground}(\beta_1, \Phi), \dots, \text{Ground}(\beta_n, \Phi) \mid \forall X_1, \dots, X_k; \alpha \rightarrow \beta_1 \wedge \dots \wedge \beta_n \in \Delta \text{ and } \text{Ground}(\alpha, \Phi) \text{ is } R \}$$

Example 3.2 Let R be the following report.

report(forecast(date(25July01), city(Malaga), today(blizzard)))

Suppose the set of access rules, Δ , contains the following.

$$\forall X, Y, Z; \text{report}(\text{forecast}(\text{date}(X), \text{city}(Y), \text{today}(Z))) \rightarrow \text{date}(X) \wedge \text{location}(Y) \wedge \text{today}(Z)$$

So for R we get

$$\text{date}(25\text{July}2001), \text{location}(\text{Malaga}), \text{today}(\text{blizzard}) \in \text{Access}(\Delta, R)$$

Suppose Δ also contains the following access rule.

$$\forall X, Y; \text{report}(\text{forecast}(\text{date}(X), \text{city}(Y), \text{today}(\text{blizzard}))) \rightarrow \text{blizzardWarning}(X, Y)$$

So for R we get

$$\text{blizzardWarning}(25\text{July}2001, \text{Malaga}) \in \text{Access}(\Delta, R)$$

We assume that the textentries in structured news reports are heterogeneous in format. For example, the format of date values is unconstrained (12/12/1974; 31st Dec 96; 12 Nov 2001 etc.) as is the format of numbers and currency values (3 million; 3, 000, 000 GBP; \$4, ¥500K etc.). Elsewhere, we have discussed how this heterogeneity can be handled in logic by various kinds of equivalence axioms [Hun00a, Hun02a, Hun02c]. To simplify our exposition in this paper, we will assume that a preprocessor will convert these textentries into a standard format.

3.3 Domain facts

The domain facts come from a set of domain specific databases. These hold data concerning key entities. In the mergers and acquisitions domain these would include companies, subsidiaries, key personnel, turnover, business activities and so on.

Definition 3.6 *The domain facts are a set of ground literals (i.e. atoms and negated atoms).*

Example 3.3 *For the mergers and acquisitions domain, domain facts may include the following.*

```
in(London, UK)
memberOf(United Kingdom, EU)
sector(Pirelli, tyreAndCable)
¬sector(Pirelli, finance)
¬sector(Pirelli, food)
```

Domain facts may include negative literals. These may be listed explicitly as negative literals or they may have been obtained by the closed world assumption [Rei78]. There may be restrictions on which types of facts may be subjected to the closed world assumption and which are subjected to the open world assumption. We do not consider the implementation details further here.

In many domains such as mergers and acquisitions news, there is much information available in existing relational databases that can be used as domain facts. In order to minimize the problems of knowledge engineering for any given domain, it is intended that the use of such relational data should be harnessed.

4 Event modeling

Reports do not normally exist in isolation. There is an underlying narrative which concerns a number of entities related in some way over a period of time. In many domains, stories will follow a stereotypical sequence. A criminal act for example may begin with the offence, continue through investigation, arrest, trial, through to conviction and possibly appeal.

In some domains, the transition from state to state may not be clear-cut. There may be iterations of sequences of states, states may be omitted or reordered, it may not even be clear where one state ends and another begins. Nonetheless, an understanding of states is imperative if the unfolding of a narrative is to be understood. The progression of states forms the narrative which relates reports and entities to events.

In order to represent and reason with narratives, we need to model states and changes of state. In the domain of mergers and acquisitions for example, phrases include *agree*, *complete*, and *approve*. These words or their synonyms are used to indicate when a state changes. Additional information about narratives is usually found in close proximity to these phrases in news reports, such as dates, entity names and the tense of the phrases (“shareholders will approve” is a very different state to “shareholders have approved”).

Event modeling consists of three parts: state models, characterizing events, and the event calculus. The result of event modeling is an event model for a domain. In use, an event model will be updated with information in news reports. An event model can therefore be regarded as an up-to-date repository of information about a set of stories. For a given domain, we may have a number of different event models capturing different kinds of stories in the domain, though for simplicity, in the rest of the paper we just consider individual event models.

4.1 State models

A state model is an abstract definition of the possible events that can take place in a stereotypical narrative, along with the ordering of these events. We assume that events are initiated by entities. For the mergers and acquisitions domain, these entities include legal entities such as companies, agents for legal entities such as company secretaries and company boards, semi-autonomous parts of companies such as subsidiaries and joint ventures, stakeholders in companies such as unions, shareholders and customers, government agencies such as the monopolies and mergers commission and the inland revenue, and government departments such as the Treasury.

Definition 4.1 A **state model** is a directed graph, whose nodes correspond to states and whose arcs correspond to events. A **state** of an entity is an attribute of an entity with limited duration. An **event** takes place at a point in time at which one or more entities do something to bring about a change in their state.

The state model defines the relationships between events and states for a given domain. We give an example in Figure 2. Each event begins and/or ends a period of time for which a state holds. The state model can be represented by a set of formulae which we assume exhaustively define all possible orderings of events and, by extension, states.

Definition 4.2 An **action atom** is a type of news atom that describes the type of event that a news report is about. We use the predicate symbol `act` for the action atoms. A **state atom** is a type of news atom that contains information concerning entities, dates and values in order to identify and describe a distinct event. An **event** is represented by an action atom and one or more state atoms.

Example 4.1 For the mergers and acquisitions domain, action atoms include `act(bid)`, `act(reject)` and `act(approve)` and state atoms include `buyer(Shell)` and `target(Texaco)`. An example of an event in the mergers and acquisitions domain is:

$$\text{act}(\text{bidMade}), \text{buyer}(\text{Granada}), \text{target}(\text{Carlton}), \\ \text{value}(9\text{BillionGBP}), \text{date}(25\text{Jan}1998)$$

Once the state model has been constructed, it needs to be represented in a form that the system can use. To do this we translate the state model into a set of binary meta-level predicates that allow us to define the behaviour of the action and state atoms.

Definition 4.3 A state model is represented by the `comesAfter`, `initial` and `terminal` meta predicates. The ordering given by a state model can be translated into logic by defining a corresponding binary predicate `comesAfter`. Events which may come after the empty “start” state, in which there is no relationship between two entities, are **initial** events. These events are not preceded by any other event. Events which initiate the empty “end” state and so are not followed by any other event are **terminal** events. Both the `initial` and `terminal` predicates are monadic.

Example 4.2 For the state model in Figure 2, the state where two entities, the buyer and the target, are united by the fact that the target’s board has accepted the buyer’s bid follows the state where the target has made a bid for a buyer:

$$\forall B \in \text{Buyer}, T \in \text{Target}; \text{comesAfter}(\text{boardAcceptance}(B, T), \text{bidMade}(B, T)).$$

The `bidInvited` event may be the first event in a sequence.

$$\forall T \in \text{Target}; \text{initial}(\text{bidInvited}(T))$$

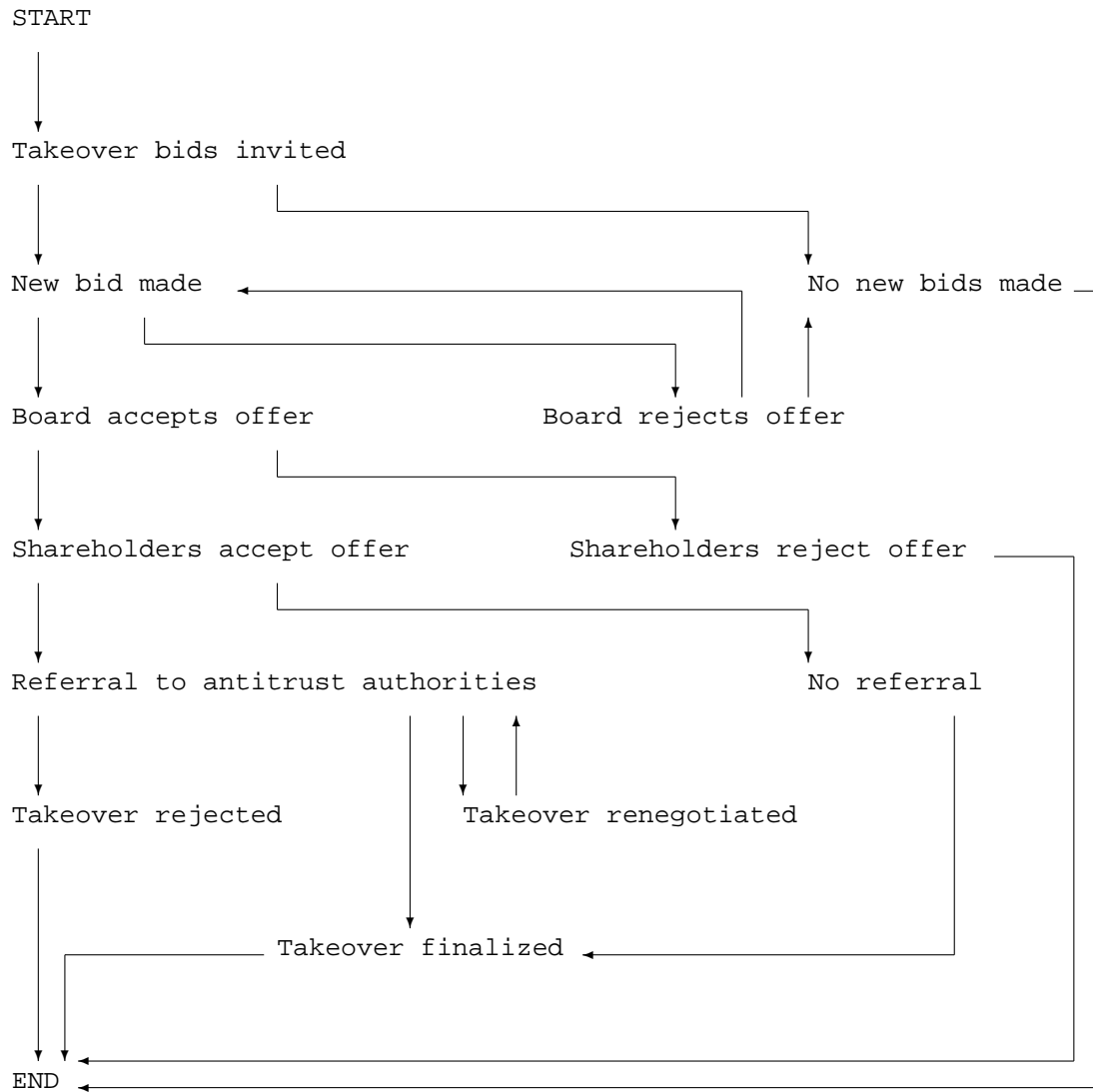


Figure 2: A simplified state model for the acquisitions domain.

The takeoverRejection event is the last event of a sequence.

$$\forall B \in \text{Buyer}, T \in \text{Target}; \text{terminal}(\text{takeoverRejection}(B, T))$$

States are begun and ended by events. The initiates predicate allows us to define which states are begun by which events.

Definition 4.4 An initiates axiom is of the following form where $\{Y_1, \dots, Y_m\} \subseteq \{X_1, \dots, X_n\}$, β is a predicate symbol, and each of the predicates $\alpha_1(X_1), \dots, \alpha_n(X_n)$ is grounded as either an action atom or a state atom, and γ is a optional predicate with its free variables being a subset of $\{X_1, \dots, X_n\}$

$$\forall X_1, \dots, X_n; \text{initiates}(\beta(Y_1, \dots, Y_m)) \leftrightarrow \alpha_1(X_1) \wedge \dots \wedge \alpha_n(X_n) \wedge \gamma$$

Example 4.3 The state during which shareholders have accepted a bid is begun by an event whose action is “acceptance” and the body concerned are the shareholders:

$$\forall B \in \text{Buyer}, T \in \text{Target}; \text{initiates}(\text{shareholderAcceptance}(B, T)) \leftrightarrow \\ \text{act}(\text{acceptance}) \wedge \text{target}(T) \wedge \text{buyer}(B) \wedge \text{body}(\text{shareholders})$$

The act and body predicates relate to the type of event in question, and the buyer and target predicates are state predicates used to tie acts to specific events.

Example 4.4 The state during which a company is profitable is begun by a report of operating profits:

$$\forall C \in \text{Company}, P \in \text{Profit}; \text{initiates}(\text{profitable}(C)) \leftrightarrow \\ \text{act}(\text{profitAnnouncement}) \wedge \text{company}(C) \wedge \text{profit}(P) \wedge P > 0$$

Once the state model is in place we can reason with events. The state model allows us to construct an event sequence. This allows us to decide whether a sequence of events is completed or ongoing and which events may follow the current event.

4.2 Characterizing events

To use a state model, we need to be able to extract information about events from news reports. This raises the need to characterize events in terms of event records. Each event record is a set of event atoms (defined below) that delineates an event. Event records register the entities that play a role in that particular event, the action, the date of the event and any other feature which may combine to uniquely identify an event, such as a bid value in the mergers and acquisitions domain. Event atoms are extracted from reports using the news atoms described in Section 3.2. It is assumed that each structured news report refers to only one event. This is a reasonable assumption if the structured news reports are generated from business newsfeeds by information extraction technology.

Definition 4.5 Let Δ be a set of access rules, and let R be a report. Let $\alpha(t_1, \dots, t_n) \in \text{Access}(\Delta, R)$. An **event atom** for R is of the form $\alpha(e, t_1, \dots, t_i)$ where e is an event identifier (a label used to denote the event record it is contained in).

Example 4.5 The following event atoms capture the event when Walmart has made a bid for Asda on the 3rd of June 1999.

$$\text{act}(e1, \text{bid}) \\ \text{target}(e1, \text{Asda}) \\ \text{bidder}(e1, \text{Walmart}) \\ \text{eventTime}(e1, 03/06/1999)$$

Example 4.6 *The following event atoms capture the event when Asda's board has accepted Walmart's bid on the 7th July 1999:*

```
act(e2, acceptance)
  target(e2, Asda)
bidder(e2, Walmart)
  body(e2, board)
eventTime(e2, 07/07/1999)
```

The way we derive event atoms from news atoms is via event rules as defined below. If the event referred to is not already the subject of an event record, the event rules use a new event identifier to label the information.

Definition 4.6 *An event rule is a formula of the following form where $\alpha(X_1, \dots, X_n)$ is a news atom, and $\alpha(E, X_1, \dots, X_n)$ is an event atom.*

$$\forall X_1, \dots, X_n, E; \alpha(X_1, \dots, X_n) \rightarrow \alpha(E, X_1, \dots, X_n)$$

We use the event rules as follows.

Definition 4.7 *Let Δ be a set of access rules, let Γ be a set of event rules, and let R be a report.*

$$\text{Event}(\Gamma, R) = \{ \text{Ground}(\alpha(e, X_1, \dots, X_k), \Phi) \mid \\ \text{Ground}(\alpha(X_1, \dots, X_k), \Phi) \in \text{Access}(\Delta, R) \\ \text{and } \forall X_1, \dots, X_n, E; \alpha(X_1, \dots, X_n) \rightarrow \alpha(E, X_1, \dots, X_n) \in \Gamma \\ \text{and } e \text{ is a unique event identifier} \}$$

Note, e is a unique identifier if and only if e has not yet been used in any event record.

So $\text{Event}(\Gamma, R)$ is the smallest set of event atoms obtained by exhaustively applying the news atoms obtained from report R to the event rules in Γ using generalized modus ponens and conjunction elimination. Once a set of event records has been constructed they can be applied to the event model. This gives us a way of representing the states in which we find entities at a given time.

Example 4.7 *Given the following report*

```
report(buyer(WHSmiths), target(Stars), act(bidMade),
      date(6/6/02), value(42 million GBP))
```

and the access rule

$$\forall B, T, A, D, V; \text{report}(\text{buyer}(B), \text{target}(T), \text{act}(A), \text{date}(D), \text{value}(V)) \\ \rightarrow \text{buyer}(B) \wedge \text{target}(T) \wedge \text{act}(A) \wedge \text{date}(D) \wedge \text{value}(V)$$

and the event rule

$$\forall B, T, A, D, V, E; \text{buyer}(B) \wedge \text{target}(T) \wedge \text{act}(A) \wedge \text{date}(D) \wedge \text{value}(V) \\ \rightarrow \text{buyer}(E, B) \wedge \text{target}(E, T) \wedge \text{act}(E, A) \wedge \text{date}(E, D) \wedge \text{value}(E, V)$$

we obtain the event atoms where $e3$ is a unique event identifier

```
buyer(e3, WHSmiths)
target(e3, Stars)
act(e3, bidMade)
date(e3, 6/6/02)
value(e3, 42 million GBP)
```

If the event referred to in a report is already the subject of an event record then a new event record is not created. A report concerns an event which features in an existing event record if and only if the information which appear in the report is identical or synonymous with the information in the event record.

4.3 The event calculus

In order to reason with the events and the state model we need some way to define the relations between events and states and the relations between states and times. For this, we adopt the event calculus proposed by Kowalski and Sergot ([KS85]). In this calculus, meta-level predicates are used to identify the states (or relationships) which hold at given timepoints by recording the events which initiate and terminate those states.

The event calculus incorporates the `holds` predicate which relates states to timeperiods and the `holdsAt` predicate which relates times and states. These are axiomatised below, using the `after` and `before` functions, which map events and states to timeperiods, and the `fallsIn` predicate which relates timepoints to states.

Definition 4.8 *A state holds after the event which initiates it and before the event which terminates it:*

$$\begin{aligned} \forall E \in \text{Event}, S \in \text{State}; \text{holds}(S, \text{after}(E, S)) &\leftrightarrow \text{initiates}(E, S) \\ \forall E \in \text{Event}, S \in \text{State}; \text{holds}(S, \text{before}(E, S)) &\leftrightarrow \text{terminates}(E, S) \end{aligned}$$

Definition 4.9 *A state holds at a timepoint if the timepoint falls in the period during which that state holds:*

$$\begin{aligned} \forall E \in \text{Event}, S \in \text{State}, T \in \text{Timepoint}; \text{holdsAt}(S, T) &\leftrightarrow \\ &\text{holds}(S, \text{after}(E, S)) \wedge \text{fallsIn}(T, \text{after}(E, S)) \\ \forall E \in \text{Event}, S \in \text{State}, T \in \text{Timepoint}; \text{holdsAt}(S, T) &\leftrightarrow \\ &\text{holds}(S, \text{before}(E, S)) \wedge \text{fallsIn}(T, \text{before}(E, S)) \end{aligned}$$

We use the `holds` and `holdsAt` definition along with the recorded events to identify which states hold at a given timepoint.

Example 4.8 *Consider the following event atoms,*

$$\begin{array}{ll} \text{act}(e1, \text{bid}) & \text{act}(e2, \text{acceptance}) \\ \text{target}(e1, \text{Asda}) & \text{target}(e2, \text{Asda}) \\ \text{bidder}(e1, \text{Walmart}) & \text{bidder}(e2, \text{Walmart}) \\ \text{eventTime}(e1, 03/06/1999) & \text{eventTime}(e2, 07/07/1999) \\ & \text{body}(e2, \text{board}) \end{array}$$

and the following axioms,

$$\forall E \in \text{Event}, T \in \text{Target}, B \in \text{Bidder}; \text{initiates}(E, \text{bidMade}(B, T)) \leftrightarrow \text{act}(E, \text{bid}) \wedge \text{target}(E, T) \wedge \text{bidder}(E, B)$$

$$\forall E \in \text{Event}, T \in \text{Target}, B \in \text{Bidder}; \text{terminates}(E, \text{bidMade}(B, T)) \leftrightarrow \text{act}(E, \text{acceptance}) \wedge \text{target}(E, T) \wedge \text{bidder}(E, B) \wedge \text{body}(E, \text{board})$$

Using the event calculus, we can infer:

$$\begin{aligned} &\text{holds}(\text{bidMade}(\text{Walmart}, \text{Asda}), \text{after}(e1, \text{bidMade}(\text{Walmart}, \text{Asda}))) \\ &\text{holds}(\text{bidMade}(\text{Walmart}, \text{Asda}), \text{before}(e2, \text{bidMade}(\text{Walmart}, \text{Asda}))) \end{aligned}$$

Further rules confirm that these events are the initiator and terminator for a single, continuous state. Therefore, any timepoint between the 3rd June 1999 and the 7th June 1999 coincides with Asda and Walmart being in the positions of target and buyer respectively.

The above definitions are sufficient when we have complete information about events. Realistically however there will be times when there is no recorded initiating or terminating for a state. To address this, we have augmented the Kowlaski and Sergot rules for `holdsAt` (Definition 4.9) by adding definitions where the `fallsIn` predicate is augmented with `fallsBefore`, `fallsAfter` and `fallsBetween` predicates.

The `fallsBefore` predicate uses the `after` function to ascertain whether the event has any possible predecessors. If there are no recorded predecessors and the event is not an `initial` event then the relationship which holds before that event holds at any timepoint which falls in the time up to that event. The `fallsAfter` predicate uses the `after` function to ascertain whether the event has any possible subsequent events. If there are no recorded subsequent events and the event is not a `terminal` event then the relationship which holds after that event holds at any timepoint which falls in the time since that event. The `fallsBetween` predicate uses the `after` function to identify two event which belong to the same event sequence but which are separated by at least one unrecorded event. In this case there are at least two relations or states which potentially hold at any given time between those two events: the `unterminated` relation and the `uninitiated` relation.

For a more comprehensive coverage of our version of the event calculus, including the complete axiomatization for the state model given in Figure 2, see [BH02].

4.4 Event models

We can now pull together the various object-level and meta-level formulae we have considered in this section to give the following definition of an event model.

Definition 4.10 *An event model is a set of object-level and meta-level formulae that is the union of the following sets: (1) a state model represented by a set of `comesAfter`, `initial`, and `terminal` predicates (see Definition 4.3) and a set of `initiates` axioms (see Definition 4.4); (2) Zero or more event records represented by sets of event axioms (see Definition 4.5); and (3) The event calculus axioms (see Section 4.3).*

An event model can be viewed as a repository of information obtained from news reports. As each news report is processed using access rules and event rules, the set of event records is increased. Then via the state model axioms and the event calculus axioms, the information can be queried. It is intended that each event model is self-contained. In other words, once the event records have been created, queries to an event model use just the formulae in the event model. This means we can regard an event model as a separate module.

Definition 4.11 *Let Π be an event model, Γ be a set of event rules, and R be a structured news report. The Prolog version of inference is denoted by the \vdash_{prolog} relation.*

$$\begin{aligned} \text{EventQueries}(\Pi, \Gamma, R) = & \\ & \{\text{Holds}(s, p) \mid s \text{ and } p \text{ are ground terms and } \text{Event}(\Gamma, R) \cup \Pi \vdash_{prolog} \text{Holds}(s, p)\} \\ & \cup \{\text{HoldsAt}(s, t) \mid s \text{ and } t \text{ are ground terms and } \text{Event}(\Gamma, R) \cup \Pi \vdash_{prolog} \text{HoldsAt}(s, t)\} \end{aligned}$$

For any instantiation of Π , Γ and R , the set $\text{EventQueries}(\Pi, \Gamma, R)$ gives all the `holds` and `holdsAt` atoms that follow from the events in R and the event model Π using Prolog. In the rest of the paper, our approach to seeking violations of expectations uses the set $\text{EventQueries}(\Pi, \Gamma, R)$ rather than querying Π or $\text{Event}(\Gamma, R)$. This maintains the modularity of the event model.

In practice, it is not necessary to generate $\text{EventQueries}(\Pi, \Gamma, R)$. Rather, in the course of seeking violations of expectations, queries are posted to $\text{Event}(\Gamma, R) \cup \Pi$ to see if particular holds atoms and holdsat atoms are members of $\text{EventQueries}(\Pi, \Gamma, R)$. The modularity is still maintained in this case.

Based on this modularity, the system may have more than one event model in order to deal with different types of state. In the mergers and acquisitions domain for example, it is also necessary to keep track of the state of a company's key performance indicators such as profit, earnings per share, and so on. A separate event model exists to record and reason with events pertaining to company figures. Each event model can exist as an independent module. Furthermore, the underlying version of event calculus can differ. We have used the proposal by Kowalski and Sergot and adapted it for our needs. But there are a number variants on event calculus that we could consider (for a review see [MS02]).

5 Expectations about news

Expectations are formulae that capture general relationships between news reports, background knowledge, and event models. Unlike information that we assume is always correct, e.g. definitions, mathematical properties, and integrity constraints, we assume expectations can be violated some of the time. In a sense, an expectation is a form of defeasible or default rule. Violations of expectations allow the system to identify information which is unusual but not necessarily incorrect.

Definition 5.1 *An expectation is a formula of the following form where $\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_m$ are literals and X_1, \dots, X_k are free variables occurring in the literals,*

$$\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \rightarrow \beta_1 \vee \dots \vee \beta_m$$

and for all Φ , if $\text{Ground}(\alpha_1 \wedge \dots \wedge \alpha_n, \Phi)$ is a ground formula, then $\text{Ground}(\beta_1 \vee \dots \vee \beta_m, \Phi)$ is a ground formula.

Examples of expectations are given in Example 5.1 to Example 5.3 below.

Example 5.1 *The heuristic “The value of a tendered bid is expected to be greater than 90% of the company's market capitalisation” can be represented by the following expectation.*

$$\forall C \in \text{Companies}, V \in \text{BidValue}, M \in \text{MarketCapitalization}; \\ \text{bidTendered}(C, V) \wedge \text{marketCapitalization}(C, M) \rightarrow (V/M) \geq 0.9$$

Example 5.2 *The heuristic “A company is expected to have sufficient available capital to be able to cover a bid” can be represented by the following expectation.*

$$\forall C \in \text{Companies}, V \in \text{BidValue}, A \in \text{AvailableCapital}; \\ \text{bidTendered}(C, V) \wedge \text{availableCapital}(C, A) \rightarrow A > V$$

Example 5.3 *The heuristic “A company is expected to bid for a target in a sector which supplies or is supplied by that company's sector or that is compatible with the company's sector” can be represented by the following expectation.*

$$\forall X, Y \in \text{Companies}; \text{target}(X, Y) \rightarrow \\ (\text{sector}(X) = \text{sector}(Y) \vee \text{supplier}(\text{sector}(X), \text{sector}(Y)) \vee \\ \text{supplier}(\text{sector}(Y), \text{sector}(X)) \vee \text{compatible}(\text{sector}(Y), \text{sector}(X)))$$

Expectations may be developed either by manually analysing a body of news reports, by obtaining heuristics from experts in the domain, or by using data mining techniques to extract expectations from a corpus of reports. Data mining software is able to identify rules in databases that capture patterns of recurrent behaviour but we believe that individually these rules are not usually useful. However if these rules are harnessed as expectations in the MBD framework, we believe that collectively they would become much more useful.

Some expectations are specific to certain entities or sets of entities. Furthermore, some expectations hold for all entities at all times, whereas others hold only for entities in given states. The event model is what allows us to identify the state of a given entity and so identify which of these expectations (which we will call state dependent expectations) should hold.

Definition 5.2 *A state dependent expectation is an expectation whose antecedent contains the holdsAt relation.*

State dependent expectations apply to an entity E at timepoint T only if there exists a relation R which holdsAt T for E.

Example 5.4 *The heuristic “A company which is unprofitable will not be expected to have a rising shareprice” can be represented by the following state dependent expectation.*

$$\forall C \in \text{companies}, T \in \text{Timepoints}; \\ \neg \text{holdsAt}(\text{profitable}(C), T) \rightarrow \neg \text{sharemovement}(C, T, \text{rising})$$

Expectations, by their very nature, are not correct 100% of the time. In other words, it is possible for the antecedent of an expectation to hold, and for the negation of the consequent to hold, and hence for there to be an inconsistency with the expectation. It is when an expectation leads to an inconsistency that we expect to see interesting information. Some expectations are unviolated for a greater percentage of reports than others. A rule is stronger than another if for a greater percentage of reports, the consequent holds when the antecedent holds. Violations of strong rules are more interesting than violations of weaker rules. In this section, we focus on what we mean by violating an expectation. In Section 6, we consider evaluation of violations of expectations in detail.

Violations are classified as either cohort or singular violations. We explain these terms next and then formalize the definitions in the rest of this section.

Singular violations These occur when a report concerning an entity includes facts which are inconsistent with the state that that entity is known to be in *at the time* which the reports concerns.

Cohort violations These occur when one or more reports concerning *more than one* entity include facts which are inconsistent with the state that that entity is known to be in *at the time* which the reports concern.

For both singular violations and cohort violations, we need to consider a report atom for each report R together with access rules, events rules, domain facts, an event model, and a set of expectations.

5.1 Singular violations of expectations

As discussed in Section 1, we assume that news reports are consistent with the background knowledge. This is summarized as follows where R is a report, Δ is a set of access rules, Γ is a set of event rules, Λ is

a set of domain facts, and Π is an event model.

$$\text{Access}(\Delta, R) \cup \text{EventQueries}(\Pi, \Gamma, R) \cup \Lambda \not\vdash \perp$$

Using this assumption, we define a singular violation of expectations as follows.

Definition 5.3 *Let R be a report, Δ be a set of access rules, Γ be a set of event rules, Λ be a set of domain facts, Π be an event model, Σ be a set of expectations, and $\phi \in \Sigma$.*

*R is a **singular violation** of ϕ iff $\text{Access}(\Delta, R) \cup \text{EventQueries}(\Pi, \Gamma, R) \cup \Lambda \cup \{\phi\} \vdash \perp$*

Example 5.5 *Let R be the following report*

report(takeover(bidder(Pirelli), target(Olivetti)))

Assume an access rule in Δ gives us

target(Olivetti, Pirelli)

Let Λ contain the following facts

sector(Pirelli, tyresAndCables)
sector(Olivetti, informationTechnology)
¬compatible(tyresAndCables, informationTechnology)
¬compatible(informationTechnology, tyresAndCables)
¬supplier(tyresAndCables, informationTechnology)
¬supplier(informationTechnology, tyresAndCables)
informationTechnology ≠ tyresAndCables

Also assume Σ contains the following expectation.

$\forall X, Y \in \text{Companies}, A, B \in \text{Sectors};$
target(X, Y) ∧ sector(X, A) ∧ sector(Y, B) →
A = B ∨ supplier(A, B) ∨ supplier(B, A) ∨ compatible(A, B) ∨ compatible(B, A)

Hence this expectation is violated by the news report.

Example 5.6 *Consider an event in the event model which allow us to conclude*

holdsAt(inAdministration(Railtrack), 02/02/2002)

Suppose the system then receives a report which we represent by the following report atom

report(company(Railtrack), annualreport(result(profit), amount(GBP292m), date(02/02/2002)))

And from this suppose we can derive

holdsAt(profitable(Railtrack), 02/02/2002)

Now suppose there is a state dependent expectation in Σ such that

$\forall C \in \text{Companies}, T \in \text{Time};$
holdsAt(profitable(C), T) → ¬holdsAt(inAdministration(C), T)

Hence we can derive ¬holdsAt(inAdministration(Railtrack), 02/02/2002) and therefore we obtain an inconsistency.

5.2 Cohort violations of expectations

Some violations of expectations only become interesting when a number of entities all exhibit the same unexpected behaviour. The EntityCount function (which we define below) allows us to determine how many entities appear in a set of reports.

Definition 5.4 An entity classifier is a function of the form Entity_z where z is a set of news atoms. An entity classifier assigns the value True or False to each report R as follows.

$$\text{Entity}_z(\Delta, R) = \text{True iff } z \subseteq \text{Access}(\Delta, R)$$

Example 5.7 Consider the following report atom.

$$\text{report}(\text{act}(\text{bidMade}), \text{target}(\text{Sabena}), \text{buyer}(\text{BritishAirways}))$$

From this, suppose $\text{Access}(\Delta, R) = \{\text{act}(\text{bidMade}), \text{target}(\text{Sabena}), \text{buyer}(\text{BritishAirways})\}$. If we refer to the above report as R , then

$$\begin{aligned} \text{Entity}_{\{\text{target}(\text{Sabena})\}}(\Delta, R) &= \text{True} \\ \text{Entity}_{\{\text{buyer}(\text{BritishAirways})\}}(\Delta, R) &= \text{True} \\ \text{Entity}_{\{\text{target}(\text{Sabena}), \text{buyer}(\text{BritishAirways})\}}(\Delta, R) &= \text{True} \\ \text{Entity}_{\{\text{target}(\text{Sabena}), \text{buyer}(\text{Ryanair})\}}(\Delta, R) &= \text{False} \end{aligned}$$

Definition 5.5 Let Y be a set of semantic labels. We assume each semantic label can also be used as a predicate symbol. The set $\text{EntitySet}(Y)$ is a set of entity classifiers such that for each $\text{Entity}_z \in \text{EntitySet}(Y)$, the following two conditions hold: (1) for each $p(q) \in z$, $p \in Y$; and (2) for each $p \in Y$, there is a $p(q) \in z$.

Definition 5.6 Let $R_1 \dots R_n$ be a set of reports and Y be a set of semantic labels.

$$\text{EntityCount}(\{R_1 \dots R_n\}, \Delta, Y) = |\{\text{Entity}_z \in \text{EntitySet}(Y) \mid \exists R_i \in \{R_1 \dots R_n\} \text{ s.t. } \text{Entity}_z(\Delta, R_i) = \text{True}\}|$$

Example 5.8 For the set of reports $\{R_1, R_2, R_3\}$, where

$$\begin{aligned} \text{Access}(\Delta, R_1) &= \{\text{act}(\text{bidMade}), \text{target}(\text{Sabena}), \text{buyer}(\text{BritishAirways})\} \\ \text{Access}(\Delta, R_2) &= \{\text{act}(\text{bidMade}), \text{target}(\text{Ryanair}), \text{buyer}(\text{BritishAirways})\} \\ \text{Access}(\Delta, R_3) &= \{\text{act}(\text{bidMade}), \text{target}(\text{Northwest}), \text{buyer}(\text{American})\} \end{aligned}$$

we have $\text{EntityCount}(\{R_1, R_2, R_3\}, \Delta, \{\text{target}\}) = 3$, $\text{EntityCount}(\{R_1, R_2, R_3\}, \Delta, \{\text{buyer}\}) = 2$, and $\text{EntityCount}(\{R_1, R_2, R_3\}, \Delta, \{\text{target}, \text{buyer}\}) = 3$.

In general, for $\text{EntityCount}(\{R_1 \dots R_n\}, \Delta, Y)$, we need the semantic labels to delineate appropriate entities. For example, for companies, $Y = \{\text{name}, \text{location}\}$ would allow for companies with the same name but different locations to be differentiated. Similarly, for staff records, $Y = \{\text{firstname}, \text{lastname}, \text{birthdate}\}$ may be appropriate.

We are now ready to define the notion of a cohort violation. As with the definition of a singular violation, we assume that the news reports are consistent with the background knowledge. Using this assumption, we define a cohort violation of expectations as follows.

Definition 5.7 Let $\{R_1, \dots, R_n\}$ be a set of reports, Δ be a set of access rules, Γ be a set of event rules, Π be the event model, Λ be a set of domain facts, Σ be a set of expectations, and $\phi \in \Sigma$.

$\{R_1, \dots, R_n\}$ is a **cohort violation** of ϕ
iff
there is a non-empty Y such that $\text{EntityCount}(\{R_1 \dots R_n\}, \Delta, Y) > 1$
and
 $\text{Access}(\Delta, R_1) \cup \text{EventQueries}(\Pi, \Delta, R_1) \cup \Lambda \cup \{\phi\} \vdash \perp$
and
:
and
 $\text{Access}(\Delta, R_n) \cup \text{EventQueries}(\Pi, \Delta, R_n) \cup \Lambda \cup \{\phi\} \vdash \perp$

Example 5.9 Suppose the set of domain facts Λ includes the following facts

$\neg\text{profitable}(\text{Amazon}), \neg\text{profitable}(\text{Boo}), \neg\text{profitable}(\text{Yahoo})$

and that we receive a set of reports, $\{R_1, \dots, R_n\}$ such that the following is a subset of the access predicates $\text{Access}(\Delta, R_1) \cup \dots \cup \text{Access}(\Delta, R_n)$.

$\{\text{shareMovement}(\text{Amazon}, \text{up}), \text{shareMovement}(\text{Boo}, \text{up}), \text{shareMovement}(\text{Yahoo}, \text{up})\}$

Suppose also that the expectations Σ contains the following

$\forall C \in \text{companies}; \neg\text{profitable}(C) \rightarrow \neg\text{shareMovement}(C, \text{up})$

This leads us to conclude that the news reports taken with the background facts and the above expectation lead to inconsistency. Furthermore, $\text{EntityCount}(\{R_1, \dots, R_n\}, \Delta, \{\text{shareMovement}\}) = 3$.

Specification of the set of semantic labels Y for the EntityCount function needs to appropriately delineate entities. Inappropriate specification may distort the results as illustrated by the following example. To avoid this kind of problem, we can assume that the sets Y are selected in advanced by hand for each expectation, and therefore form part of the meta-knowledge.

Example 5.10 Assume that the following set of reports all violate the same expectation:

$\text{Access}(\Delta, R_1) = \{\text{act}(\text{bidMade}), \text{target}(\text{Sabena}), \text{buyer}(\text{BritishAirways})\}$
 $\text{Access}(\Delta, R_2) = \{\text{act}(\text{bidMade}), \text{target}(\text{Ryanair}), \text{buyer}(\text{BritishAirways})\}$
 $\text{Access}(\Delta, R_3) = \{\text{act}(\text{bidMade}), \text{target}(\text{Northwest}), \text{buyer}(\text{BritishAirways})\}$

Hence, $\text{EntityCount}(\{R_1, R_2, R_3\}, \Delta, \{\text{target}, \text{buyer}\}) = 3$, corresponding to three unique tuples made up of the entity features. Yet $\text{EntityCount}(\{R_1, R_2, R_3\}, \Delta, \{\text{buyer}\}) = 1$, and so this set of reports may suggest that the behaviour which violates this expectation is particular to one company acting as a buyer.

By comparing Definition 5.3 with Definition 5.7, we see the key difference is that for a cohort violation, the condition $\text{EntityCount}(\{R_1, \dots, R_n\}, \Delta, Y) > 1$ holds for some non-empty Y , whereas for the singular violations, there is the implicit condition $\text{EntityCount}(\{R\}, \Delta, Y) = 1$ holding for some non-empty Y that is always satisfiable.

6 Evaluation of inconsistencies

Confirmation theory is a subject that has developed within the fields of statistics, probability theory, and the philosophy of science. The need for such a theory arises because evidence e rarely establishes conclusively

that some hypothesis h is true, but e may nonetheless confirm or corroborate h to some degree [GGH⁺92]. Confirmation theory investigates the relation which holds between h and e when e confirms h to some extent, but does not establish it completely.

We use a simple form of confirmation theory to evaluate expectations, and thereby evaluate the inconsistencies arising when they are violated by news reports. Each expectation can be viewed as a form of hypothesis. Furthermore, information in each news report may be used as evidence to help confirm or disconfirm some of the expectations. We will explain how we can do this below. Our approach is based on the proportion of correct firings of each expectation.

Definition 6.1 *Let \mathbf{R} be a report, Δ be a set of access rules, Γ be a set of event rules, Λ be a set of domain facts, Π be the event model, and Σ be a set of expectations. An expectation in Σ of the form $\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \rightarrow \beta_1 \vee \dots \vee \beta_m$ is **fired** by a report \mathbf{R} when there is a Φ such that*

$$\text{Access}(\Delta, \mathbf{R}) \cup \text{EventQueries}(\Pi, \Gamma, \mathbf{R}) \cup \Lambda \vdash \text{Ground}(\alpha_1 \wedge \dots \wedge \alpha_n, \Phi)$$

There are three possible outcomes, namely Correct, Incorrect, and Unknown, for each firing of an expectation with a report. The outcome Correct means that the consequent of the expectation agrees with the information in the report, background knowledge, and event model. In other words, the report, background knowledge, and event model imply a ground version of the consequent. The outcome Incorrect means that the consequent of the expectation disagrees with the information in the report, background knowledge, and event model. Finally, the outcome Unknown means the consequent of the expectation neither agrees nor disagrees with the information in the report, background knowledge, and event model.

Definition 6.2 *Let \mathbf{R} be a report, Δ be a set of access rules, Γ be a set of event rules, Λ be a set of domain facts, Π be the event model, and Σ be a set of expectations, where*

$$\text{Access}(\Delta, \mathbf{R}) \cup \text{EventQueries}(\Pi, \Gamma, \mathbf{R}) \cup \Lambda \vdash \text{Ground}(\alpha_1 \wedge \dots \wedge \alpha_n, \Phi)$$

The FireValue function is defined as follows.

$$\begin{aligned} \text{FireValue}(\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \rightarrow \beta_1 \vee \dots \vee \beta_m, \mathbf{R}) &= \text{Correct} \\ \text{iff } \text{Access}(\Delta, \mathbf{R}) \cup \text{EventQueries}(\Pi, \Gamma, \mathbf{R}) \cup \Lambda \vdash \text{Ground}(\beta_1 \vee \dots \vee \beta_m, \Phi) \end{aligned}$$

$$\begin{aligned} \text{FireValue}(\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \rightarrow \beta_1 \vee \dots \vee \beta_m, \mathbf{R}) &= \text{Incorrect} \\ \text{iff } \text{Access}(\Delta, \mathbf{R}) \cup \text{EventQueries}(\Pi, \Gamma, \mathbf{R}) \cup \Lambda \vdash \neg \text{Ground}(\beta_1 \vee \dots \vee \beta_m, \Phi) \end{aligned}$$

$$\begin{aligned} \text{FireValue}(\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \rightarrow \beta_1 \vee \dots \vee \beta_m, \mathbf{R}) &= \text{Unknown} \\ \text{iff } \text{Access}(\Delta, \mathbf{R}) \cup \text{EventQueries}(\Gamma, \mathbf{R}) \cup \Lambda \not\vdash \text{Ground}(\beta_1 \vee \dots \vee \beta_m, \Phi) \\ \text{and } \text{Access}(\Delta, \mathbf{R}) \cup \text{EventQueries}(\Pi, \Gamma, \mathbf{R}) \cup \Lambda \not\vdash \neg \text{Ground}(\beta_1 \vee \dots \vee \beta_m, \Phi) \end{aligned}$$

A firing trace for an expectation is the history of when an expectation has been fired and whether the firing was correct or not given some set of reports $\Psi = \{\mathbf{R}_1, \dots, \mathbf{R}_n\}$ obtained over time. If $\phi \in \Sigma$ then $\text{Trace}(\Psi, \phi) = \{(c_1, s_1), \dots, (c_n, s_n)\}$ where each c_i is a value denoting whether the rule fired correctly, i.e. the value assigned by the FireValue function, and so is one of {Correct, Unknown, Incorrect}, and s_i is the corresponding timestamp for the firing.

Definition 6.3 *Let Ψ be a set of report received at different points in time, Δ be a set of access rules, Γ be a set of event rules, Λ be a set of domain facts, Π be the event model, and Σ be a set of expectations, and let $\phi \in \Sigma$.*

$$\begin{aligned} \text{Trace}(\Psi, \phi) &= \{(\text{FireValue}(\phi, \mathbf{R}), \text{TimeStamp}(\mathbf{R})) \mid \mathbf{R} \in \Psi \\ &\text{and } \phi \text{ is of the form } \forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \rightarrow \beta_1 \vee \dots \vee \beta_m \\ &\text{and } \exists \Phi.s.t. \text{Access}(\Delta, \mathbf{R}) \cup \text{EventQueries}(\Pi, \Gamma, \mathbf{R}) \cup \Lambda \vdash \text{Ground}(\alpha_1 \wedge \dots \wedge \alpha_n, \Phi)\} \end{aligned}$$

where $\text{TimeStamp}(R)$ is the timestamp of the report R (i.e. the time that the report was received by the system).

Example 6.1 Let Δ be a set of access rules, and let R be a report with a timestamp of 20/02/2002. Suppose we have

$$\text{target}(\text{BritishAirways}, \text{Sabena}) \in \text{Access}(\Delta, R)$$

Now consider an expectation ϕ in Σ .

$$\forall X, Y \in \text{Companies}, A, B \in \text{Sectors}; \text{target}(X, Y) \wedge \text{Sector}(X, A) \wedge \text{Sector}(Y, B) \rightarrow A = B$$

with domain facts

$$\begin{aligned} \text{sector}(\text{BritishAirways}, \text{civilAviation}) \\ \text{sector}(\text{Sabena}, \text{civilAviation}) \end{aligned}$$

which gives us

$$(\text{Correct}, 20/02/2002) \in \text{Trace}(\{R\}, \phi)$$

We can then calculate a confirmation value based on the number of firings (the ‘‘frequency’’ of an expectation) and the ratio of successful to unsuccessful firings (the ‘‘accuracy’’ of the expectation) since a given timepoint. This is a relative frequency approach [CHS93].

Definition 6.4 Let Ψ be a set of reports received at different points in time, and let Σ be a set of expectations. For $\phi \in \Sigma$, and timepoints t_1 and t_2 , the following are subsets of $\text{Trace}(\Psi, \phi)$.

$$\text{Correct}(\Psi, \phi, t_1, t_2) = \{(\text{Correct}, t) \in \text{Trace}(\Psi, \phi) \mid t_1 \leq t \text{ and } t \leq t_2\}$$

$$\text{Incorrect}(\Psi, \phi, t_1, t_2) = \{(\text{Incorrect}, t) \in \text{Trace}(\Psi, \phi) \mid t_1 \leq t \text{ and } t \leq t_2\}$$

$$\text{Unknown}(\Psi, \phi, t_1, t_2) = \{(\text{Unknown}, t) \in \text{Trace}(\Psi, \phi) \mid t_1 \leq t \text{ and } t \leq t_2\}$$

Definition 6.5 Let Ψ be a set of reports received at different points in time, and let Σ be a set of expectations. If $\phi \in \Sigma$ then $\text{Confirmation}(\Psi, \phi, t_1, t_2) = (x, y)$ where x is the ratio of successful firings to total firings and y is the number of firings from time t_1 to t_2 .

$$x = \frac{|\text{Correct}(\Psi, \phi, t, t')|}{|\text{Correct}(\Psi, \phi, t, t')| + |\text{Incorrect}(\Psi, \phi, t, t')| + |\text{Unknown}(\Psi, \phi, t, t')|}$$

$$y = |\text{Correct}(\Psi, \phi, t, t')| + |\text{Incorrect}(\Psi, \phi, t, t')| + |\text{Unknown}(\Psi, \phi, t, t')|$$

Example 6.2 Suppose $\text{Trace}(\Psi, \phi) = \{(\text{Correct}, 1), (\text{Incorrect}, 2), (\text{Correct}, 3), (\text{Incorrect}, 4)\}$.

$$\text{Confirmation}(\Psi, \phi, 4, 5) = (0, 1)$$

$$\text{Confirmation}(\Psi, \phi, 3, 5) = (0.5, 2)$$

$$\text{Confirmation}(\Psi, \phi, 2, 5) = (0.33, 3)$$

$$\text{Confirmation}(\Psi, \phi, 1, 5) = (0.5, 4)$$

For any Ψ, ϕ, t_1, t_2 , when $\text{Confirmation}(\Psi, \phi, t_1, t_2) = (x, y)$, the measure x reflects the degree of accuracy with which an expectation predicts the real world as reflected in the news reports in Ψ , and the measure y reflects the coverage of ϕ . The coverage of a Confirmation measure is the total number of times the expectation has been fired by an incoming report in Ψ . The higher the coverage value, the more significant the confirmation measure is.

Confirmation values can be used to rank violations of expectations. The higher the confirmation value, the more unlikely it is that an expectation will be violated, hence the more interesting a violation of that expectation will be. One approach is to aggregate the two dimensions of the confirmation value into a partial ordering over the expectations is to use the \succeq relation defined as follows.

Definition 6.6 *Let i, j, p, q be real numbers and let \geq be the usual ordering over real numbers. The \succeq relation is a partial ordering relation over pairs of real numbers.*

$$(i, p) \succeq (j, q) \text{ iff } i \geq j \text{ and } p \geq q$$

The \succeq relation induces a ranking over expectations according to the confirmation values for each expectation. So if ϕ_1 and ϕ_2 are expectations such that $\text{Confirmation}(\Psi, \phi_1, t_1, t_2) \succeq \text{Confirmation}(\Psi, \phi_2, t_1, t_2)$, then violation of ϕ_1 is regarded as more significant than ϕ_2 .

Rankings will change over time as rules continue to be fired by the reports received. Continued violations of an expectation suggest a trend emerging which should lead to a change in the strength of the expectation which is violated. For example, we may have an expectation in the domain of mergers and acquisitions, where it is expected that a company will sell rather than acquire subsidiaries, indicating a market going through a period of consolidation, and also an expectation that companies will sell, rather than buy subsidiaries, indicating a period of decentralisation. If every time an expectation is violated its confirmation decreases by some degree and the confirmation of the opposing expectation increases by the same degree, we have a self-setting market state indicator.

In order to determine which rules are currently accurate, it may be necessary to give greater weight to the results of more recent firings than to firings further in the past. In order to achieve this, each firing and its associated outcome will be assigned a weight which decreases with time.

Applying confirmation theory to a cohort is more difficult. For cohort violations, we need to consider the following goals.

- maximizing the number of entities involved in the violation
- minimizing the time frame considered for the violation
- maximizing the specificity of the entity specification

and trade these against the number of entities not violating the expectation within the time frame. Rather than propose an aggregation of these goals, we believe that the user should be able to vary the time frame considered for the violation, and thereby see whether the relative number of entities violating the expectation differs from the number of entities not violating the expectation.

7 Compilation of consistency checking

Checking whether a set of formulae is consistent is in general an undirected activity. For example, given a set of formulae we could construct a semantic tableau. But this potentially involves decomposing every formula in the set into literals. For finding violations of expectations, this involves an inordinate amount of unnecessary search since only a relatively restricted set of formulae needs to be considered for each expectation.

One solution to this problem is to compile the consistency checking for each expectation. So rather than checking the consistency directly, we attempt to prove whether each expectation has been violated. The way we do this is to rewrite each expectation into another formula that we call a viaduct as follows.

Definition 7.1 Let q be the name of an expectation of the form $\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \rightarrow \beta_1 \vee \dots \vee \beta_m$. The **viaduct** of q is the following formula.

$$\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg\beta_1 \wedge \dots \wedge \neg\beta_m \rightarrow \text{violation}(q)$$

If Σ is a set of expectations, let $\text{Viaduct}(\Sigma)$ be the set of viaducts obtained from Σ .

Reasoning with $\text{Viaduct}(\Sigma)$ and reasoning with Σ is identical in the sense that the set of expectations that are identified as being violated is identical. This is demonstrated for singular violations in Proposition 7.1 and for cohort violations in Proposition 7.2 below.

Proposition 7.1 Let R be a structured news report, Δ be a set of access rules, Γ be a set of event rules, Λ be a set of domain facts, Π be an event model, and Σ be a set of expectations. Also let ϕ be the expectation $\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \rightarrow \beta_1 \vee \dots \vee \beta_m$, and let q be the name of ϕ . R is a singular violation of q where there is an expectation named q in Σ iff the following two conditions hold:

(1) $\text{Viaduct}(\Sigma)$ includes the following viaduct

$$\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg\beta_1 \wedge \dots \wedge \neg\beta_m \rightarrow \text{violation}(q)$$

(2) there is a grounding set Φ such that

$$\text{Access}(\Delta, R) \cup \text{EventQueries}(\Pi, \Gamma, R) \cup \Lambda \vdash \text{Ground}(\alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg\beta_1 \wedge \dots \wedge \neg\beta_m, \Phi)$$

Proof: R is a singular violation of q where there is an expectation named q in Σ

iff R is a singular violation of ϕ and $\phi \in \Sigma$

iff $\text{Access}(\Delta, R) \cup \text{EventQueries}(\Pi, \Gamma, R) \cup \Lambda \cup \{\phi\} \vdash \perp$ and $\phi \in \Sigma$

iff $\exists \Phi$ s.t. $\text{Access}(\Delta, R) \cup \text{EventQueries}(\Pi, \Gamma, R) \cup \Lambda \vdash \text{Ground}(\neg\beta_1 \wedge \dots \wedge \neg\beta_m, \Phi)$

and $\text{Access}(\Delta, R) \cup \text{EventQueries}(\Pi, \Gamma, R) \cup \Lambda \vdash \text{Ground}(\alpha_1 \wedge \dots \wedge \alpha_n, \Phi)$

and $\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \rightarrow \beta_1 \vee \dots \vee \beta_m \in \Sigma$

iff $\exists \Phi$ s.t. $\text{Access}(\Delta, R) \cup \text{EventQueries}(\Pi, \Gamma, R) \cup \Lambda \vdash \text{Ground}(\alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg\beta_1 \wedge \dots \wedge \neg\beta_m, \Phi)$

and $\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg\beta_1 \wedge \dots \wedge \neg\beta_m \rightarrow \text{violation}(q) \in \text{Viaduct}(\Sigma)$. \square

Proposition 7.2 Let $\{R_1, \dots, R_n\}$ be a set of structured news reports, Δ be a set of access rules, Γ be a set of event rules, Λ be a set of domain facts, Π be an event model, and Σ be a set of expectations. Also let ϕ be the expectation $\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \rightarrow \beta_1 \vee \dots \vee \beta_m$, and let q be the name of ϕ . $\{R_1, \dots, R_n\}$ is a cohort violation of q where there is an expectation named q in Σ iff the following three conditions hold:

(1) $\text{Viaduct}(\Sigma)$ includes the following viaduct

$$\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg\beta_1 \wedge \dots \wedge \neg\beta_m \rightarrow \text{violation}(q)$$

(2) $\exists \Phi$ s.t. $\text{Access}(\Delta, R_1) \cup \text{EventQueries}(\Pi, \Delta, R_1) \cup \Lambda \vdash \text{Ground}(\alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg\beta_1 \wedge \dots \wedge \neg\beta_m, \Phi)$

and

:

and

$\exists \Phi$ s.t. $\text{Access}(\Delta, R_n) \cup \text{EventQueries}(\Pi, \Delta, R_n) \cup \Lambda \vdash \text{Ground}(\alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg\beta_1 \wedge \dots \wedge \neg\beta_m, \Phi)$

(3) there is a Y s.t. $\text{EntityCount}(\{R_1, \dots, R_n\}, \Delta, Y) > 1$.

Proof: $\{R_1, \dots, R_n\}$ is a cohort violation of q where there is an expectation named q in Σ

iff $\{R_1, \dots, R_n\}$ is a cohort violation of ϕ and $\phi \in \Sigma$
iff there is a non-empty Y such that $\text{EntityCount}(\{R_1 \dots R_n\}, \Delta, Y) > 1$
and $\text{Access}(\Delta, R_1) \cup \text{EventQueries}(\Pi, \Delta, R_1) \cup \Lambda \cup \{\phi\} \vdash \perp$
and...and $\text{Access}(\Delta, R_n) \cup \text{EventQueries}(\Pi, \Delta, R_n) \cup \Lambda \cup \{\phi\} \vdash \perp$
iff there is a non-empty Y such that $\text{EntityCount}(\{R_1 \dots R_n\}, \Delta, Y) > 1$
and R_1 is a singular violation of ϕ and $\phi \in \Sigma$
and...and R_n is a singular violation of ϕ and $\phi \in \Sigma$
iff there is a non-empty Y such that $\text{EntityCount}(\{R_1 \dots R_n\}, \Delta, Y) > 1$
and $\exists \Phi$ s.t. $\text{Access}(\Delta, R_1) \cup \text{EventQueries}(\Pi, \Gamma, R_1) \cup \Lambda \vdash$
 $\text{Ground}(\alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg \beta_1 \wedge \dots \wedge \neg \beta_m, \Phi)$
and
:
and $\exists \Phi$ s.t. $\text{Access}(\Delta, R_n) \cup \text{EventQueries}(\Pi, \Gamma, R_n) \cup \Lambda \vdash$
 $\text{Ground}(\alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg \beta_1 \wedge \dots \wedge \neg \beta_m, \Phi)$
and $\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg \beta_1 \wedge \dots \wedge \neg \beta_m \rightarrow \text{violation}(q) \in \text{Viaduct}(\Sigma)$. \square

We have described the use of viaducts as a compilation of consistency checking. Proposition 7.1 (respectively 7.2) show that we maintain consistency checking as in the original definition for singular (respectively cohort) violation. The following example illustrates that compilation loses much of the possible consistency checking questions that could be asked by the original formalization of expectations. In other words, there may be a number of possible subsets of $\text{Access}(\Delta, R) \cup \text{EventQueries}(\Pi, \Gamma, R) \cup \Lambda \cup \Sigma$ that are inconsistent but that we cannot detect these if we use $\text{Viaduct}(\Sigma)$ instead of Σ . But since we were not looking for other possible inconsistencies in our original formalization, this is not a loss for the MBD framework.

Example 7.1 Suppose we have $a(d)$ and $b(d)$ in $\text{Access}(\Delta, R)$ for some report R . Also suppose two of the expectations in Σ are

$$\begin{aligned} \forall X; a(X) &\rightarrow c(X) \\ \forall X; b(X) &\rightarrow \neg c(X) \end{aligned}$$

Clearly, the set $\{a(d), b(d), \forall X; a(X) \rightarrow c(X), \forall X; b(X) \rightarrow \neg c(X)\}$ is inconsistent. However, this inconsistency is not a violation of an expectation.

A key advantage of compilation of consistency checking is the increase in computational viability. Consistency checking for classical logic is expensive in general. Deciding whether a set of propositional classical formulae is consistent is an NP-complete decision problem [GJ79]. Furthermore, deciding whether a set of propositional classical formulae Υ is a minimal inconsistent set involves (1) checking that Υ is inconsistent and (2) checking whether each maximal subset of Υ is consistent. If the cardinality of Υ is k , then doing (2) involves k consistency checks, where k is no more than linear in the size of the input. Hence, this decision problem is equivalent (modulo polynomial time) to the original PSAT problem. However, if we consider the problem as an abduction problem, where we seek the existence of a minimal subset of a set of formulae that implies a contradiction, then the problem is in the second level of the polynomial hierarchy [EG95]. Even worse deciding whether a set of first-order classical formulae is consistent is an undecidable decision problem [BBJ02]. This means compilation of consistency checking is highly advantageous as highlighted by the following proposition.

Proposition 7.3 Let R be a structured news report, Δ be a set of access rules, Γ be a set of event rules, Λ be a set of domain facts, Π be an event model, and Σ be a set of expectations. Also assume the following two sets of literals:

$$\begin{aligned} \Theta_1 &= \{\alpha \mid \text{Access}(\Delta, R) \vdash \alpha \text{ and } \alpha \text{ is a ground literal}\} \\ \Theta_2 &= \{\alpha \mid \text{EventQueries}(\Pi, \Delta, R) \vdash \alpha \text{ and } \alpha \text{ is a ground literal}\} \end{aligned}$$

Also let $\text{GroundViaduct}(\Sigma)$ be a set of ground formulae of the form $\alpha'_1 \wedge \dots \wedge \alpha'_n \wedge \neg\beta'_1 \wedge \dots \wedge \neg\beta'_m \rightarrow \text{violation}(\mathbf{q})$ where each such formula is obtained by instantiation of a formula

$$\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg\beta_1 \wedge \dots \wedge \neg\beta_m \rightarrow \text{violation}(\mathbf{q}) \in \text{Viaduct}(\Sigma)$$

using only ground terms in $\text{Terms}(\Theta_1 \cup \Theta_2 \cup \Lambda)$. Deciding whether R violates an expectation in Σ is a coNP-complete decision problem when $\Theta_1 \cup \Theta_2$ is a finite set.

Proof: R violates an expectation in Σ iff there is an expectation $\phi \in \Sigma$ such that R is a singular violation of ϕ . Suppose, ϕ is called \mathbf{q} . So, R is a singular violation of ϕ iff

(1) $\text{Viaduct}(\Sigma)$ includes the following viaduct

$$\forall X_1, \dots, X_k; \alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg\beta_1 \wedge \dots \wedge \neg\beta_m \rightarrow \text{violation}(\mathbf{q})$$

and (2) there is a grounding set Φ such that

$$\text{Access}(\Delta, R) \cup \text{EventQueries}(\Pi, \Gamma, R) \cup \Lambda \vdash \text{Ground}(\alpha_1 \wedge \dots \wedge \alpha_n \wedge \neg\beta_1 \wedge \dots \wedge \neg\beta_m, \Phi)$$

iff

$$\text{Access}(\Delta, R) \cup \text{EventQueries}(\Pi, \Gamma, R) \cup \Lambda \cup \text{Viaduct}(\Sigma) \vdash \text{violation}(\mathbf{q})$$

iff

$$\Theta_1 \cup \Theta_2 \cup \Lambda \cup \text{GroundViaduct}(\Sigma) \vdash \text{violation}(\mathbf{q})$$

Since, $\Theta_1 \cup \Theta_2 \cup \Lambda \cup \text{GroundViaduct}(\Sigma)$ is a finite set of classical propositional formulae, and $\text{violation}(\mathbf{q})$ is a classical propositional formula, determining whether this inference holds, using the classical consequence relation, is a coNP-complete decision problem. \square

This result means that checking whether a given news report R violates an expectation is reduced to a decidable problem of classical propositional logic. We have developed a prototype implementation in Prolog. This incorporates a meta-interpreter (adapted from [SS94]) that takes each viaduct in turn and checks whether the antecedent holds. Checking each literal in the antecedent of a viaduct involves querying a Prolog knowledgebase. If this knowledgebase is propositional, then Proposition 7.3 applies. Otherwise, the complexity of deciding whether R violates an expectation is given by the complexity of querying the Prolog knowledgebase.

Knowledge compilation has become an increasingly attractive proposition for dealing with the general intractability problem of propositional reasoning. In this, a propositional theory is compiled once into a target language, and this is then used to answer potentially numerous repeated queries in polynomial time, and hence the cost of the original compilation is amortized over the repeated queries (e.g. [Mar95, Dar99, DM01]). Clearly, knowledge compilation is a different idea to the notion of compiled consistency checking that we have proposed here.

8 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. Whilst structured text is useful as a resource, there are techniques to handle, analyse, and reason with it, including merging potentially inconsistent sets of news reports [Hun00a, Hun02a, Hun02c], and deriving inferences from potentially inconsistent sets of news reports [Hun00b, Hun00c, BH01, Hun01]. 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.

Central to the MBD framework is the need to evaluate the violations of expectations. This evaluation is a form of measuring of inconsistency. The better confirmed an expectation is, the more significant the inconsistency arising when the expectation is violated. This approach has wider applicability in measuring inconsistency in knowledge. Current techniques for measuring the degree of inconsistency in a set of formulae are underdeveloped. Some approaches touch on the topic. In diagnostic systems, there are proposals that offer preferences for certain kinds of consistent subsets of inconsistent information [KW87, Rei87]; in proposals for belief revision, epistemic entrenchment is an ordering over formulae which reflects the preference for which formulae to give up in case of inconsistency [Gar88]; in proposals for drawing inferences from inconsistent information there is a preference for inferences from some consistent subsets (e.g. [Bre89, BDP93]); in proposals for approximating entailment, two sequences of entailment relation are defined (the first is sound but not complete, and the second is complete but not sound) which converge to classical entailment [SC95]; and in proposals for partial consistency checking, checking is terminated after the search space exceeds a threshold which gives a measure of partial consistency of the data. However, none of these proposals provide a direct definition for degree of inconsistency.

In belief revision theory, and the related field of knowledgebase merging, there are some proposals that do provide some description of the degree of inconsistency of a set of formulae. For example, the Dalal distance [Dal88], essentially the Hamming distance between two propositional interpretations, can be used to give a profile of an inconsistent knowledgebase. Unfortunately, this does not provide a very succinct way of describing the degree of inconsistency in a given set of formulae, and it is not clear how we could compare sets of formulae using this approach. Furthermore, operators for aggregating these distances, such as the majority operator [LM98], egalitarian operator [Rev97], or the leximax operator [KP98], do not seem to be appropriate summaries of the degree of inconsistency in the original knowledgebase since they seek to find the most appropriate model for particular kinds of compromise of the original knowledge. Related techniques for knowledgebase revision are similarly inappropriate.

Another approach to handling inconsistent information is that of possibility theory [DLP94]. Let (ϕ, α) be a weighted formula where ϕ is a classical formula and $\alpha \in [0, 1]$. A possibilistic knowledgebase B is a set of weighted formulae. An α -cut of a possibilistic knowledgebase, denoted $B_{\geq \alpha}$, is $\{(\psi, \beta) \in B \mid \beta \geq \alpha\}$. The inconsistency degree of B , denoted $Inc(B)$, is the maximum value of α such that the α -cut is inconsistent. As presented, the problem with this measure is that it assumes weighted formulae. In other words, we need some form of preference ordering in addition to the set of classical formulae in the knowledgebase. The knowledgebase can be used to induce such an ordering as suggested in [BDKP00], where an ordering over inferentially weaker forms of the original formulae are generated. Again this does not offer a direct lucid view on the inconsistency in the original set of formulae.

Measuring the “amount of information” is related to the idea of measuring inconsistency. Information theory can be used to measure the information content of sets of inconsistent formulae. Applying Shannon’s measure of information, Lozinskii proposes that the information in a set of propositional formulae Γ , that has been composed from n different atom symbols, is the logarithm of the number of models (2^n) divided by the number of models for the maximum consistent subsets of Γ [Loz94]. This information theoretic measure increases with additions of consistent information and decreases with additions of inconsistent information. However, as highlighted by Wong and Besnard [WB01], the measure by Lozinskii is syntax sensitive and it is sensitive to the presence of tautologies in Γ . To address this, they suggest the use of a normal form for the formulae in Γ that is obtained by rewriting Γ into conjunctive normal form, and then applying disjunction elimination and resolution exhaustively. However, this approach does not provide a direct measure of inconsistency since for example, the value for $\{\alpha\}$ is the same as for $\{\alpha, \neg\alpha, \beta\}$.

A general characterization of inconsistency has been based on quasi-classical logic (a form of paraconsistent logic with a more expressive semantics than Belnap’s four-valued logic, and unlike other paraconsistent logics, allows the connectives to appear to behave as classical connectives). Inconsistent knowledge is anal-

ysed by considering the conflicts arising in the minimal quasi-classical models for that knowledge. This is used for a measure of coherence for each knowledgebase, and for a measure of significance of inconsistencies in each knowledgebase [Hun02b]. Whilst this is potentially useful in various applications such as comparing heterogeneous sources of information, it does not seem to help in evaluating inconsistencies arising through violations of expectations. In the MBD approach, we evaluate the inconsistency on the basis of the expectation rather than all the formulae involved in the inconsistency. We may regard the two approaches as complementary, and in some applications it may be of interest to integrate them. The confirmation theoretic approach to measuring significance that we have presented in this paper can be described as a likelihood-based significance. However, there are other dimensions for measuring significance of inconsistency. In measuring inconsistency in structured news reports, we are interested in two further kinds of significance which are described below.

Subject-based significance This can be viewed as a weighting so that some subjects (denoted by a tag-name for a structured news report or a predicate symbol for a news atom) are more significant than others when part of a violation of an expectation. For example, suppose in a weather report R1 for London the humidity is given as 10%, and in a weather report R2 for London the temperature is given as 50C. Both would violate expectations about weather in London but for most people violation with regard to temperature is regarded as more significant than an inconsistency with regard to humidity.

Entry-based significance This can be viewed as a weighting so that some text entries are more significant than others when part of a violation of an expectation. For example, suppose we have an expectation that a new Ferrari is at least 200K Euros. A violation of this with a news report about the price of a new Ferrari would be more significant if the text entry was 1K Euros as opposed to 50K Euros.

We aim to capture these two further forms of significance in future developments of the MBD framework. In addition to the framework issues, these forms of significance needs to be tuned for individual user groups since different groups would need to weight subjects and entries according to their interests and goals.

In conclusion, in this paper, we have provided a basic framework for finding interesting inconsistencies in structured news reports. We have a prototype implementation of the framework for finding and evaluating violations of expectations in mergers and acquisitions reports. This has been implemented in Prolog and incorporates an event model for this domain and a meta-interpreter for the compiled consistency checking.

References

- [BBJ02] G Boolos, J Burgess, and R Jeffrey. *Computability and Logic*. Cambridge University Press, 2002.
- [BDKP00] S Benferhat, D Dubois, S Kaci, and H Prade. Encoding information fusion in possibilistic logic: A general framework for rational syntactic merging. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI'2000)*, pages 3–7. IOS Press, 2000.
- [BDP93] S Benferhat, D Dubois, and H Prade. Argumentative inference in uncertain and inconsistent knowledge bases. In *Proceedings of Uncertainty in Artificial Intelligence (UAI'93)*, pages 1449–1445. Morgan Kaufmann, 1993.
- [BH01] Ph Besnard and A Hunter. A logic-based theory of deductive arguments. *Artificial Intelligence*, 128:203–235, 2001.
- [BH02] E Byrne and A Hunter. Event modeling for reasoning with structured news reports. Technical report, Department of Computer Science, University College London, 2002.

- [Bre89] G Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *Proceedings of the Eleventh International Conference on Artificial Intelligence (IJCAI'89)*, pages 1043–1048, 1989.
- [CHS93] J Cussens, A Hunter, and A Srinivasan. Generating explicit orderings for non-monotonic logics. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'93)*, pages 420–425. MIT Press, 1993.
- [CL96] J Cowie and W Lehnert. Information extraction. *Communications of the ACM*, 39:81–91, 1996.
- [Dal88] M Dalal. Investigations into a theory of knowledge base revision: Preliminary report. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI'88)*, pages 3–7. MIT Press, 1988.
- [Dar99] A Darwiche. Compiling knowledge into decomposable negation normal form. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 284–289, 1999.
- [DLP94] D Dubois, J Lang, and H Prade. Possibilistic logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 439–513. Oxford University Press, 1994.
- [DM01] A Darwiche and P Marquis. A perspective on knowledge compilation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 175–182, 2001.
- [EG95] T Eiter and G Gottlob. The complexity of logic-based abduction. *Journal of the ACM*, 42:3–42, 1995.
- [Gar88] P Gardenfors. *Knowledge in Flux*. MIT Press, 1988.
- [GGH⁺92] D Gabbay, D Gillies, A Hunter, S Muggleton, Y Ng, and B Richards. The Rule-based Systems Project: Using confirmation theory and non-monotonic logics for incremental learning. In S Muggleton, editor, *Inductive Logic Programming*. Academic Press, 1992.
- [GH91] D Gabbay and A Hunter. Making inconsistency respectable. In *Foundations of Artificial Intelligence Research*, volume 535 of *Lecture Notes in Computer Science*, pages 19–32. Springer, 1991.
- [GJ79] M Garey and D Johnson. *Computers and Intractability*. W H Freeman, 1979.
- [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. Hybrid argumentation systems for structured news reports. *Knowledge Engineering Review*, 16:295–329, 2001.
- [Hun02a] A Hunter. Logical fusion rules for merging structured news reports. *Data and Knowledge Engineering*, 42:23–56, 2002.
- [Hun02b] A Hunter. Measuring inconsistency in knowledge via quasi-classical models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'02)*, pages 68–73. MIT Press, 2002.

- [Hun02c] A Hunter. Merging structured text using temporal knowledge. *Data and Knowledge Engineering*, 41:29–66, 2002.
- [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.
- [KS85] R Kowalski and M Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1985.
- [KW87] J De Kleer and B Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.
- [LM98] J. Lin and A.O. Mendelzon. Merging databases under constraints. *International Journal of Cooperative Information Systems*, 7(1):55–76, 1998.
- [Loz94] E Lozinskii. Information and evidence in logic systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 6:163–193, 1994.
- [Mar95] P Marquis. Knowledge compilation using prime implicates. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 837–843, 1995.
- [MS02] R Miller and M Shanahan. Some alternative formulations of the event calculus. In *Computational Logic: Logic Programming and Beyond*, volume 2408 of *Lecture Notes in Computer Science*, pages 452–490. Springer, 2002.
- [Rei78] R Reiter. On closed world databases. In H Gallaire and J Minker, editors, *Logic and Databases*, pages 55–76. Plenum Press, 1978.
- [Rei87] R Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.
- [Rev97] P Revesz. On the semantics of arbitration. *International Journal of Algebra and Computation*, 7:133–160, 1997.
- [SC95] M Schaerf and M Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74:249–310, 1995.
- [SS94] L Sterling and E Shapiro. *The Art of Prolog*. MIT Press, 1994.
- [WB01] P Wong and Ph Besnard. Paraconsistent reasoning as an analytic tool. *Journal of the Interest Group in Propositional Logic*, 9:233–246, 2001.