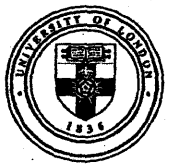


REFERENCE ONLY

UNIVERSITY OF LONDON THESIS

Degree PhD Year 2005 Name of Author BYRNE E-C.

The Expectation Violation Analysis Framework: The analysis and evaluation of interesting news by means of inconsistency with expectations

Emma Louise Byrne

Department of Computer Science
University College London

PhD
October 2005

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

Large volumes of news are available around the clock and around the world. It is increasingly difficult to sort interesting from uninteresting news. Information filtering and retrieval have addressed this problem by comparing lexical information in documents with a query or profile of user's interests. However, this approach does little to address the fact that the most interesting news is usually that which is unexpected. This thesis presents a framework that addresses this problem by reasoning about the information in reports, comparing it with the general knowledge of background information and expectations. This framework makes it possible to identify and rank unexpected, and therefore interesting, news.

The Expectation Violation Analysis (EVA) framework presented here draws *reasoned conclusions* as to why the information in a report is or is not interesting. The EVA framework contains a set of background knowledge and a set of expectations which represent the usual state of the world. When a report contains information that contradicts this knowledge and these expectations then that report contains information that is of interest.

This thesis demonstrates that the EVA framework supports the development of a system that rates news reports for interest together and explains why the report was considered unexpected.

This thesis also shows that the set of all expectations is very large. It is demonstrated that the strength of expectations is related to the implication order of the antecedents and consequents of those expectations. A method for generating expectations based on news reports and background knowledge is presented, as is empirical work that indicates the approach is plausible. Also presented is a novel extension to the event calculus that permits reasoning about missing states. These extensions support the development of expectations and background knowledge that make it possible to reason about time.

Acknowledgements

I would like to acknowledge the help and guidance of my supervisor, Dr Tony Hunter, without whom this research would not have been possible. I would also like to thank Professor Janet McDonnell and Professor John Washbrook for their help and their invaluable and inspiring comments.

I would like to thank my colleagues at UCL who have been so instrumental in my progress, whether as sounding boards, sources of interesting references or, not least, moral support. Among these I would like to single out Dr Simone Stumpf for her unfailing encouragement, and Robert Bowerman for his insightful critiques of my developing work. Last, but never least, my heartfelt thanks to my husband, Dr David Corney for guidance on machine learning, text mining, thesis writing, and for having my dinner on the table after many a long day's writing up.

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

Emma Byrne
June 2005

Contents

Chapter 1

Introduction

The number of news reports published online is now so great that it is impossible for any person to read all of them. Not all of these reports are equally interesting. Automating the identification and evaluation of interest in news is therefore a potentially valuable goal. For the purposes of this thesis, interesting news is that which is contrary to expectations. In the Expectation Violation Analysis (EVA) framework proposed in this thesis, expectations, news reports and background knowledge are represented by logical formulae. These three sets of knowledge are at the heart of the EVA framework.

The overall goal of the research presented here is to formalise the EVA framework and in particular to define notions of interestingness, analyse the set of expectations, propose a method for generating expectations and explore a method for reasoning about time. The current study is restricted to the domain of business news, but the solution proposed can be adapted to any news domain for which a set of expectations can be generated and that is sufficiently restricted such that a body of background knowledge can be collected.

This introductory chapter examines why the identification of interesting news is a useful goal and why it presents a particular problem. It then examines several different notions of interest. Finally it outlines the contribution this research will make and the structure of the thesis.

1.1 The challenge of identifying interesting news

Much has been done to address the problems of information overload. Information retrieval and information filtering are two well known approaches, for example. However, news reports present a unique challenge: news reports are, by their nature, frequently updated and their interest value is short lived; news is often most interesting because it is unexpected and news is also heavily

context dependent, that is, the assumption is made that the reader has knowledge of the entities involved in and the historical context for the events described.

Current methods for dealing with information overload such as information retrieval and information filtering (IR/IF) [BC92] work by looking for documents that are relevant to keywords in a query or a profile. These approaches work well for most bodies of information, but keyword filtering has two main drawbacks with respect to filtering news:

- Keywords do not focus on the examination of the *context* of the report, that is, the terms that are indexed come only from the document itself or meta-information that is explicitly coded in the document or index. Related information from databases and other sources is not usually examined. When analysing the interestingness of news, it is necessary to be able to consider such background knowledge, as it forms the context for the report. For example, a news report that details the activities of Marks and Spencer plc. may not state that Marks and Spencer plc. is a British high-street retailer, listed on the sock exchange as such information is held to be common knowledge. However, that same information may be essential in deciding whether or not the news in the report is interesting.
- Keyword filtering does not address the *unexpectedness* of information. Keyword searches match terms in a report without requiring any deeper understanding of the expected behaviour of entities in the real world. As a result, these systems do not decide whether the news in a report is unexpected. For example, news of a large utility company, such as Enron, going bankrupt is (or was) unexpected. It would be necessary to formulate keyword searches that matched a variety of unexpected situations, in order to identify such behaviour. To do so is problematic in the light of the variety of unexpected, and often unthinkable, possible events.

The unique features of news reports demand a special approach for identifying and explaining their interestingness. Such an approach should take into account the context of the report and expectations about the world. The EVA framework is such an approach.

In order to identify interesting news it is necessary to understand what interestingness is. The next section presents the findings from several fields within computer science and cognitive science and defines interest for the purposes of the EVA framework.

1.2 The nature of interest

To address the interestingness of news reports it is necessary to define what it means for a news report to be interesting. Researchers in the fields of knowledge discovery in databases, machine learning, machine creativity and cognitive science have explored the characteristics that make

information interesting. The first thing to note is that, whilst there is no agreement on a single definition of interest, three features are common to most definitions:

1. Interest has what is sometimes referred to as an *intrinsic* component [Sch79, ST96]. The intrinsic component is those items that are of interest to what Perelman and Olbrechts-Tytecha [POT69] call the *universal audience*, an audience made up of hypothetical rational, informed members. Interest also has what is sometimes called an *extrinsic* component, that is, items that are of interest to a particular audience. For example: if my brother were to declare bankruptcy, that would be interesting to me, as a particular audience, but of little interest to the universal audience. In contrast, the news that Bill Gates had declared bankruptcy could be considered interesting to the universal audience.

2. The second common feature is that there is an *ordering* over interest: it is possible to define some things as more interesting than others.

3. Finally, there is agreement that interest is at least partially due to the *unexpected* nature of the interesting object or, at the very least, things that are expected are not as interesting as things that are unexpected. This is related to the information theory notion that news of something improbable has more "information value" than news of something probable.

Interest is a notion of use in knowledge discovery in databases (KDD). KDD is the process of identifying potentially useful new patterns in data. However the number of patterns generated is often too great to be assessed effectively by the user. Interestingness measures have been introduced as a way of reducing the number of patterns presented to the user. In [ST95] and [ST96] Silberschatz and Tuzhilin recognise that interest has both an intrinsic component and an extrinsic component. Silberschatz and Tuzhilin subdivide extrinsic interest into actionability (whether the information is of use in relation to the desires and intentions of the user) and unexpectedness (whether the information is *inconsistent* with the beliefs of the user). To illustrate: news that interest rates are going up, thus making one's mortgage more expensive is actionable, by means of paying off one's mortgage, or selling one's home, even if it is not unexpected. In contrast, the abdication of Queen Elizabeth II in favour of Prince Charles would be unexpected but, for most of us, not actionable.

A different perspective, that of Piatetsky-Shapiro and Matheus ([PSM94]), is that interest is due to the *'novelty, utility, relevance and statistical significance'* of the discovered knowledge. Novelty is closely related to the notion of unexpectedness, as, for information to be unexpected, it must be novel. Statistically significant information is defined as numerical data that is deviant from the usual range, and therefore unexpected. Relevance and utility are closely related to the notion of actionability: for information to lead to action related to the user's desires and intentions, the information must be relevant to those desires and intentions and must also be useful in informing that action in some way.

The idea that interest is at least in part due to unexpectedness or surprisingness is also reflected in results from the field of cognitive science. The findings support the hypothesis that a report is significant to users both because of that report's general appeal (intrinsic interest) and the user's own beliefs, desires and intentions (extrinsic interest). The cognitive science literature also supports the notion of *degrees* of interest. Schank addressed the notion of unexpectedness and, in [Sch79], proposes that unexpected events are interesting and that "things are interesting in direct proportion to their abnormality". However, Schank's notion of a degree of interest is informal and no measures are suggested.

One of the differences between intrinsic and extrinsic unexpectedness is the notion of the audience. Intrinsic unexpectedness relates to what would be unexpected for a hypothetical universal audience. Assuming that such an audience could exist, the expectations that they hold can be approximated by the objective probability of an event's occurrence. In contrast, extrinsic unexpectedness is based on the particular audience's subjective reckoning of the normality of an event's occurrence. A person's expectations about what is normal are influenced by cognitive noise, that is, any cognitive effect that accounts for some of the differences between the measured probability of an event and that person's subjective impressions of that event's probability. Several mechanisms have been identified that are responsible for this noise in human cognition. However, the findings presented in [Plo93] demonstrate that measured probabilities do give an approximation of the expectations held by most people.

In [OP87], Ortony and Partridge differentiate between unexpectedness and surprisingness. They claim that, in order for something to be *unexpected* there must be an *expectation*, which is explicit, whereas *surprise* arises from events that challenge *assumptions*, which are implicit. Therefore, assumptions differ from expectations in that they are never consciously expressed but they are what Ortony and Partridge call "practically deducible". The term "deducible" is not used in a logical sense: a practically deducible assumption is one which can be arrived at from the agent's current knowledge without "many and complex inferences". This difference is not universal: one person's expectation may be another's assumption. However, with regard to the EVA framework, the term assumption has no real meaning, as all expectations in an EVA system will be made explicit, *a priori*. However, an EVA system may contain expectations that are held by the user as assumptions. For simplicity's sake, the term surprise will not be used, and we will restrict discussion to unexpectedness. It should be understood, however, that such unexpectedness may be experienced by the user as unexpectedness *or* as surprise.

1.2.1 A combined view of interest

The results given above provide an intuition of what a notion of interestingness might be. In this section, the terms that will be used throughout this thesis are presented. I propose a view of interest that combines the various findings discussed above within a single framework.

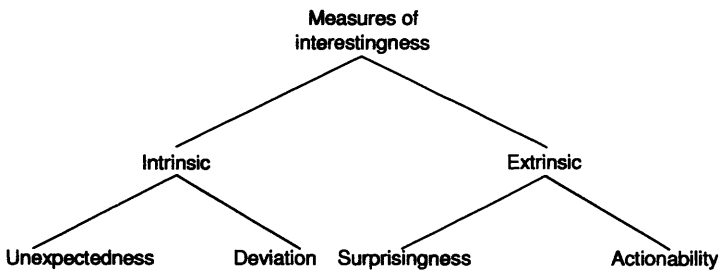


Figure 1.1: A schema of the components of interest.

Figure 1.1 presents a synthesis of the findings outlined above. Interest is divided into intrinsic and extrinsic components, as suggested by Silberschatz and Tuzhilin and Schank.

The *extrinsic* component of interest in this combined taxonomy is subdivided into *actionability* and *surprisingness*. Actionability is a reflection of the relevance of that information to a particular task at hand and the utility of the information. Surprisingness in this definition encompasses both of what Ortony and Partridge call unexpectedness and surprisingness. In this thesis, the term surprisingness will be used to refer to an extrinsic element of interest that arises when information is inconsistent with a given user's beliefs. Surprise is then a reflection of the user's subjective probability of events.

The *intrinsic* component of interest includes an element that will be referred to in this thesis as *unexpectedness*. Unexpectedness is the interest that arises when news violates expectations that have a high measured probability of not being violated. The intrinsic component also includes a *deviation* measure. Deviation is a numeric measure that indicates to what degree a number is outside some expected range. It can be argued that deviation is a special case of unexpectedness, where ranges of values are definitions of expected values.

The terms expectations and unexpectedness will be formalised in Chapters 2 and 3. For now it is enough to know that expectations are rules about usual behaviour, and that they may be violated. This thesis will concentrate on unexpectedness as a measure of interest for two reasons: unexpectedness is not only a measure of interest in its own right, but is also a good indication of interest that arises from surprise and is a way of identifying deviations. Secondly, unexpectedness supports a measure, or at the very least, a ranking, of interest. The stronger the expectation that is violated, the more interesting the violation. One drawback is that unexpectedness is not necessarily indicative of actionability as actionability is largely dependent on the *relevance* of a document to the user's task. It is not the intention of this thesis to address relevance as it is already well understood due to the work done in the fields of information retrieval and filtering. It is intended that the EVA framework could complement, rather than replace, traditional information retrieval and filtering methods.

1.3 Interest and the Expectation Violation Analysis framework

As in the adage, "When a dog bites a man, that is not news. When a man bites a dog, that is news"[1]. Information that is unexpected is of interest and news in particular owes much of its interest to unexpectedness. Many published news articles confirm or reiterate knowledge that the reader would already be expected to know, or else concerns events which are, to some degree, predictable. The more predictable the information in a news report, the less interesting that information is. In the case of business, technical, and scientific news, where new discoveries or the reversal of trends are noteworthy, unexpectedness is largely responsible for the interest-value of news. It is therefore a worthwhile goal to automate the identification of unexpected news.

The EVA framework proposed in this thesis supports the development of a system that will present a user with a set of reports that are ranked or rated for interest. By representing expectations, domain knowledge and the information in a news report as classical formulae, the violation of an expectation can be characterised as a type of logical inconsistency. In an EVA system, violations of expectations are used to identify unexpected news.

A system based on the EVA framework is able to take news reports, a set of background knowledge and a set of expectations in the form of logical formulae. An EVA system can identify inconsistencies between these reports and background knowledge and expectations. Furthermore an EVA system can give an explanation for why a report is interesting in terms of the expectation that is violated and background knowledge that is deemed relevant and give a measure of unexpectedness. Finally, the EVA framework supports the automatic generation of expectations based on news reports and background knowledge. The EVA framework is described in more detail in Chapter 2.

1.4 Introduction to the thesis

This section presents the scope of the thesis, the proposed contributions and an outline of the thesis structure.

Scope of the thesis: The EVA framework is a theoretical framework that supports the development of a system to identify and evaluate interesting news by means of identifying violations of expectations. The framework also supports the presentation of an explanation for the decision to classify this news as interesting and the ranking interest over reports. The thesis will concentrate on the identification of unexpected news and the generation of a set of expectations that are at the heart of an EVA system.

[1] Attributed to Charles Anderson Dana, Editor of the New York Sun, 1882

Contributions: The overall contribution of this thesis is a logic-based framework that supports the identification and evaluation of interesting news reports. A system based on the framework analyses news reports in the form of logical formulae and returns those reports that are unexpected with respect to a set of expectations and a set of background knowledge. Unexpected news reports are presented to the user, along with relevant background knowledge and a measure of the expected degree of interest in the news report. This work is unique in that logic has so far not been applied to the problem of identifying interesting news. The advantage of this approach is that the content of news is analysed at a semantic level, taking into account background knowledge, rather than at a lexicographic level (that is, searching by keywords). This allows interesting events to be identified and a justification for the decision to rate an article as interesting to be given. The justification presented to the user may include information from the news report and details of the expectation violated along with relevant facts derived from the background knowledge.

The thesis presents four major contributions and several minor contributions. These contributions, and their location within the thesis, are outlined in figure 1.2 on page 15.

A summary of the chapters in this thesis is given below:

Chapter 2: news of events that are unexpected is, in general, more interesting than news of events that are expected. It is therefore a useful goal to automate the identification of news that is unexpected. Chapter 2 presents the EVA framework that enables the identification and evaluation of unexpected news. This framework consists of a set of news reports in the form of logical facts, a set of background knowledge in the form of logical facts and rules and a set of expectations in the form of logical rules. The set of background knowledge provides contextual information that is relevant to reports. This background knowledge also may include a set of facts and rules pertaining to the relations between states and events. This set, collectively called the event model, enables the construction of narratives from sequences of news reports, thus providing *historical* context for news. Unexpected news is identified by searching for inconsistencies between an expectation and a consistent set of facts from a news report and the background knowledge.

Chapter 3: expectations are used by an EVA system to identify news that is unexpected. It is possible for news and facts derived from the set of background knowledge to either contradict or to confirm an expectation. Chapter 3 defines expectations and the relationships between expectations, news and background knowledge. This chapter also defines measures of the strength of expectations. Chapter 3 defines how inconsistency can be identified in a way that is computationally viable.

Chapter 4: for any domain, as defined by the set of predicate symbols, constant symbols and connectives in the language, there is a set of all possible expectations. It is useful to understand the characteristics of this set. Chapter 4 defines the set of expectations and demon-

strates that the implication order of the expectations in the set is related to the order of the strength of those expectations. These orders can be harnessed to identify those expectations that are strong enough to be of use in recommending reports.

Chapter 5: for even quite restricted languages the set of all possible expectations is very large. An undirected search for the best expectation to characterise the interestingness of a news report is impractical. Chapter 5 presents a heuristic that reduces the set of expectations to be searched. This set, the set of working expectations, contains only those expectations that are sufficiently strong that a violation of any one of those expectations would be interesting. This set is also small enough to be searched on receipt of each report.

Chapter 6: the heuristic presented in Chapter 5 is key to the viability of an EVA system. It is therefore useful to validate the heuristic empirically. Chapter 6 presents a set of simulations that empirically assess the performance of a prototype of a working expectation generator. The results of these simulations demonstrate that the set of working expectations can be generated more effectively in some domains than in others. Those domains in which the variance between probabilities of certain events is high are most suitable for the generation of expectations. Chapter 6 also demonstrates that business news is a suitable domain for the application of the heuristic that generates the set of working expectations.

Chapter 7: the set of background knowledge provides contextual information that complements news reports. One important subset of the background knowledge is an event model. An event model supports the construction of a narrative of related events from a series of news reports. This narrative provides context for later news reports. Chapter 7 presents a method for constructing an event model based on Kowalski and Sergot's event calculus and explains its use within the EVA framework. This chapter also explores the problem of unreported events and defines functions that can infer what these events may have been. The event calculus allows narratives to be derived from news reports. However, the Kowalski and Sergot formulation of the event calculus assumes complete information. As this is not a safe assumption to make concerning news reports, where much information is unstated, the event interpolation operator was devised. The event interpolation operator allows complete narratives to be created, even where information is lacking by identifying rules in the event model that can be used to reason abductively about missing states.

Chapter 8: the EVA framework was not developed in isolation. There are several bodies of literature that are relevant to the work presented in this thesis. Chapter 8 discusses the literature that is relevant to the EVA framework. The findings from various domains are presented with respect to methods of measuring inconsistency, identifying interesting information and generating rules. The EVA framework is considered in relation to these approaches and its relative strengths and weaknesses discussed.

Chapter 9 concludes with a summary and a discussion of the findings in this document. Chapter 2 sets out a programme for a framework that will fulfil the goal of identifying and

reasoning about interest in news. In subsequent chapters, the useful formal properties of the framework are developed. Outstanding issues are explored with suggestions for future research directions. Chapter 9 also discusses ways in which the results from this thesis can be exploited in other domains including identifying inconsistencies in information merging systems and defining the space of all possible templates for an information extraction system.

1.4.1 Publications

Results from Chapter 2 were published in [BH04] and results from Chapter 4 were published in [BH05]. Material in section 2.6 has been developed to explore the explanatory abilities of the EVA framework, and has been accepted for publication ([Byr05]). Material from Chapters 5, 6 and 7 will be submitted for publication at a later date. Work from Chapters 5 and 6 will be published as a presentation and evaluation of the heuristic for generating a set of working expectations. Work from Chapter 7 will be published as a discussion of the use of the event calculus and the interpolation operators presented in this thesis as a way of creating and maintaining an event model.

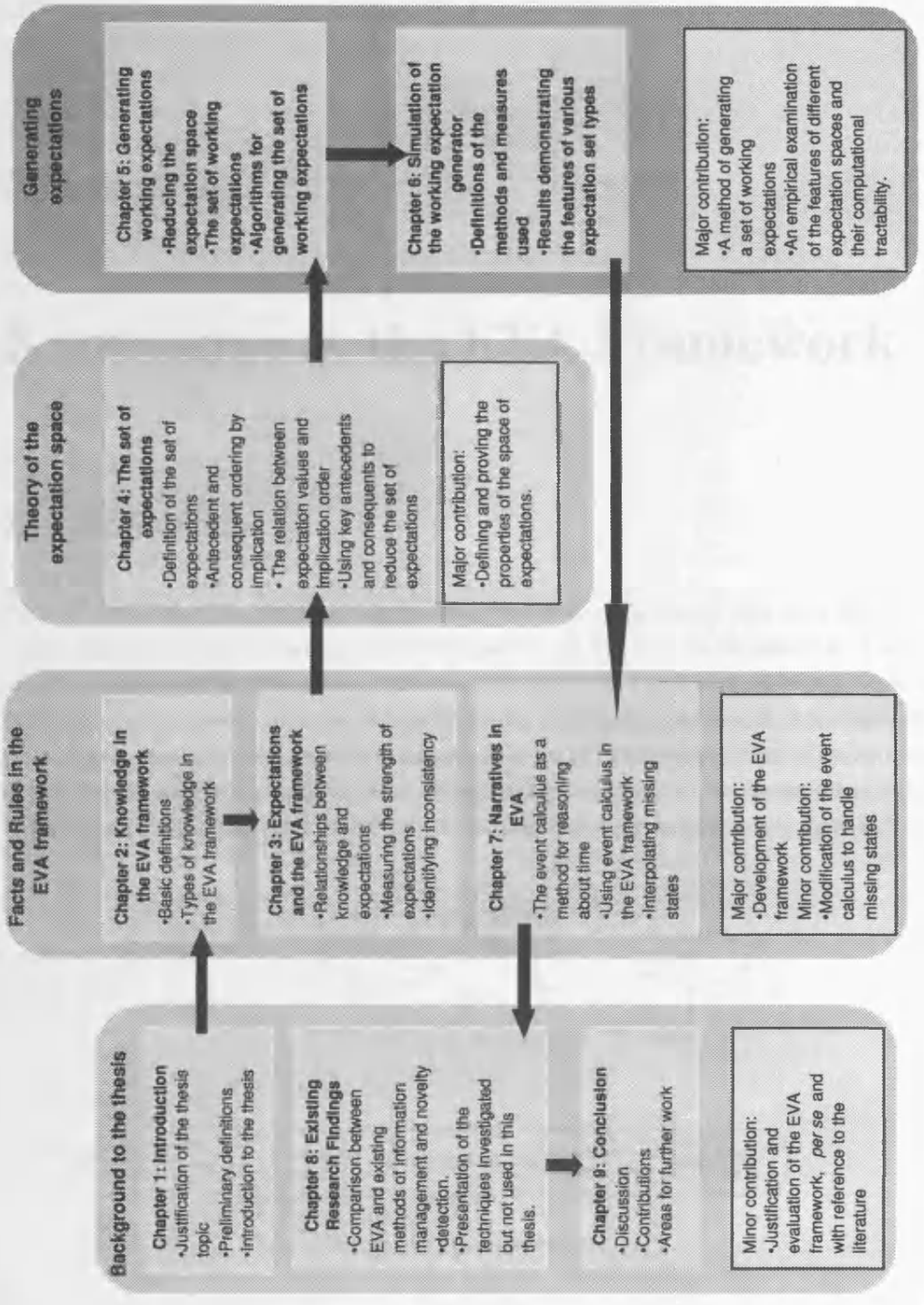


Figure 1.2: The structure of this thesis and the major and minor contributions

Chapter 2

Knowledge in the EVA Framework

2.1 Introduction

Chapter 1 introduced the notion that unexpected news is of more interest than news that is expected. Current information management techniques do not focus on the identification of unexpected information. The Expectation Violation Analysis (EVA) Framework addresses the need for a technique to identify unexpected information by identifying violations of expectations by news reports. Inputs to an EVA system consists of a set of news reports, a set of background knowledge and a set of expectations. An implemented system based on the framework is called an EVA system. The sequence of activities within the EVA framework is shown in figure 2.1.

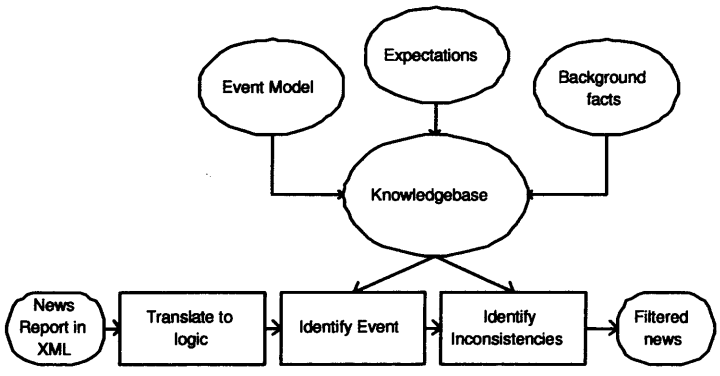


Figure 2.1: The EVA process.

The input to an EVA system is a stream of news reports in structured text format such as XML. The process of an EVA system is then to: (1) translate each incoming news report into logic; (2) identify the events featured in each report; (3) identify inconsistencies between each report,

together with a set of background knowledge, and a repository of expectations; (4) evaluate each inconsistency for significance and (5) output interesting reports, highlighting violations of expectations. The activities will be the same for a wide variety of domains but the background knowledge and set of expectations need to be created for each domain.

The rest of this chapter is laid out as follows. Section 2.2 reviews some basic definitions for classical logic. Section 2.3 formalises the notion of structured news reports, defines how they can be represented as logical literals, and presents access rules that are used for accessing information encoded in news reports. Section 2.4, defines the set of background knowledge which includes a set of facts and a set of rules. A special subset of the background knowledge is an event model that supports reasoning about sequences of events. Event models are fully defined in Chapter 7. Section 2.5 defines expectations, which are key to identifying unexpected information.

2.2 Basic definitions

The symbols used throughout this thesis are those of classical first-order logic. This includes the usual symbols $\forall$ and $\exists$ for quantification, the symbols $\wedge, \vee, \rightarrow$ and $\neg$ for logical connectives and $\vdash$ for the classical consequence relation. Other symbols denote predicate symbols, function symbols, constant symbols, variable symbols and sets of such symbols.

Definition 2.2.1. *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 $t_1, ..., t_n$ are terms then $f(t_1, ..., t_n)$ is a term. For all terms t, if t incorporates a variable, then t is an* **unground term**, *otherwise t is* **ground term**. *If $f(t_1, ..., t_n)$ is a term, then t_1 is a* **subterm** *and...and t_n is a subterm. If t is a term, then $t \in \mathsf{subterms}(t)$, and for any term $t' \in \mathsf{subterms}(t)$, the subterms of t' are in $\mathsf{subterms}(t)$.*

Definition 2.2.2. *Let $\mathcal{P}$ be a set of predicate symbols, and $\mathcal{T}$ be a set of terms. If $t_1, ..., t_n \in \mathcal{T}$ and $p \in \mathcal{P}$ then $p(t_1, ..., t_n)$ is a* **predicate**. *If $p(t_1, ..., t_n)$ is a predicate and t_1 and... and t_n are ground terms, then $p(t_1, ..., t_n)$ 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 2.2.3. *Let α be a literal. $\mathsf{terms}(\alpha)$ is the set of terms in α as follows:*

$$\mathsf{terms}(p(t_1, ..., t_n)) = \mathsf{subterms}(t_1) \cup ... \cup \mathsf{subterms}(t_n)$$
$$\mathsf{terms}(\neg p(t_1, ..., t_n)) = \mathsf{subterms}(t_1) \cup ... \cup \mathsf{subterms}(t_n)$$
$$\mathsf{terms}(\{\alpha_1, ..., \alpha_n\}) = \mathsf{terms}(\alpha_1) \cup ... \cup \mathsf{terms}(\alpha_n)$$

Definition 2.2.4. *Let* $p_1(t_1, ..., t_j), ..., p_n(t_k, ..., t_n)$ *be predicates.* α *is a* **formula** *iff:*

$\alpha = p_1(t_1, ..., t_j) \wedge ... \wedge p_n(t_k, ..., t_n)$ *or*
$\alpha = p_1(t_1, ..., t_j) \vee ... \vee p_n(t_k, ..., t_n)$ *or*
if α_i, α_j *are formulae then*
$\alpha = \alpha_i \wedge ... \wedge \alpha_k$ *or*
$\alpha = \alpha_i \vee ... \vee \alpha_k$ *or*
$\alpha = \neg\alpha_i$ *or*
$\alpha = \alpha_i \rightarrow \alpha_k$

Definition 2.2.5. *Let* x, y, z *be variable symbols and* a, b, c *be constant symbols. A* **grounding set** *is a set of equalities between variable and constant symbols.*

Example 2.2.1. *Let* $\Phi = \{x = a, y = b, c = d\}$. Φ *is a grounding set.*

Definition 2.2.6. *Let* Φ *be a grounding set and* α *be a formula. A* **grounding** *is a function* $\mathsf{ground}(\alpha, \Phi)$ *that gives the results of substituting each variable* x *in each predicate in* α *with constant* c *where the grounding* $x = c$ *is in* Φ.

Example 2.2.2. *Let* $\mathtt{a(b(x), c(y), d(z)) \wedge e(x)}$ *be a formula where* `x`, `y` *and* `z` *are variables. Let the grounding set* Φ *be* $\{\mathtt{y = john}, \mathtt{z = betty}\}$.

$$\mathsf{ground}(\mathtt{a(b(x), c(y), d(z)) \wedge e(x)}, \Phi) = \mathtt{a(b(x), c(john), d(betty)) \wedge e(x)}$$

Definition 2.2.7. *Let* α *be a formula such that the variables in* α *are* $x_1, ..., x_n$. Φ *is a* **total grounding set** *if there exists a set of constants* $\{c_1, ..., c_n\}$ *such that for each* $x_j \in \{x_1, ..., x_n\}$ *there exists* $c_j \in \{c_1, ..., c_n\}$ *and such that* $x_j = c_j \in \Phi$.

Example 2.2.3. *Let* $\mathtt{a(b(x), c(y), d(z)) \wedge e(x)}$ *be a formula where* `x`, `y` *and* `z` *are variables. Let the grounding set* Φ *be* $\{\mathtt{x = fred}, \mathtt{y = john}, \mathtt{z = betty}\}$. Φ *is a total grounding set and*

$$\mathsf{ground}(\mathtt{a(b(x), c(y), d(z)) \wedge e(x)}, \Phi) = \mathtt{a(b(fred), c(john), d(betty)) \wedge e(fred)}$$

For the sake of simplicity, throughout this thesis, unless explicitly specified, all grounding sets are total grounding sets and all ground formulae are completely ground, that is, they do not contain variable symbols.

2.3 Representing news reports in first order logic

This section examines news atoms, which are logical facts. It is assumed that each report is provided in the form of structured text. 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, CWL99, Cir01, SHWA03] are expected to significantly increase the amount of structured text available.

2.3.1 Structured news reports

Techniques exist to handle, analyse, and reason with data in structured text form, including merging potentially inconsistent sets of news reports [Hun00a, Hun02a, Hun02c], and deriving inferences from potentially inconsistent sets of news reports [Hun00c, Hun00d, BH01, Hun01]. Structured text can be naturally viewed in logic, as each item of structured text can be represented by a formula of classical logic, which means that consistency checking and inferencing can be undertaken with structured text using domain knowledge.

This section also introduces the notion of i) news atoms as facts extracted from news and ii) of a set of background knowledge used in conjunction with news atoms and iii) how these can be used to create a representative set of facts arising from news and selected background knowledge.

Structured text can be used to publish semi-structured data, where text entries are unstructured text, or structured data such as relational data [FMST01]. It is possible to apply integrity constraints to structured text data [FS00]. This supports the validation of the data input to an EVA system. Because of the structured nature of such documents, data can easily be transformed into logic [Hun00b].

Throughout this thesis it is assumed that the structured text news is made available. In practice this may need to be extracted from news reports. The degree of accuracy with which such data extraction could be carried out has not been put to the test, although some information extraction was performed on news from the Economist and Reuters (see Section 6.1.7). In the business domain, news tends to be reported in a largely homogeneous fashion, that is, the same form of words appears frequently when reporting certain types of events. This simplifies the task of generating templates. It remains to be shown that the output from information extraction systems is presently at the degree of reliability that an EVA system would require, but current applications of information extraction tools are encouraging (for example, [CBLJ04]).

Each structured news report is an XML document, but not vice versa, as defined below.

Definition 2.3.1. *Let τ be a tag name (i.e. an element name), ψ be a text entry (i.e. a phrase represented by a string) and $\nu_1, ..., \nu_n$ be structured news reports. R is a* **structured news report**

iff:

$$R = \langle\tau\rangle\psi\langle/\tau\rangle \text{ or}$$
$$R = \langle\tau\rangle\nu_1, ..., \nu_n\langle/\tau\rangle$$

Example 2.3.1.

```
⟨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⟩
```

An example of a structured news report

Each structured news report is isomorphic to a tree, with each tag name being represented by a non-leaf node and each text entry being represented by a leaf node. Furthermore, each subtree in a structured news report is isomorphic to a ground term where each tag name is represented by a function symbol and each text entry is represented by a constant symbol. Hence, each structured news report can be represented by a ground logical atom in classical logic as follows.

Definition 2.3.2. *A* **report atom** *is a ground atom that is the representation of a structured news report in first order logic. Let* $\langle\tau\rangle\psi_1, ..., \psi_n\langle/\tau\rangle$ *be a structured news report,* t_1 *be a ground term that represents* ψ_1 *and...and* t_n *be a ground term that represents* ψ_n. $\tau(t_1, ..., t_n)$ *is the report atom for* $\langle\tau\rangle\psi_1, ..., \psi_n\langle/\tau\rangle$.

Example 2.3.2. *Consider the following structured news report.*

```
⟨auctionreport⟩
    ⟨buyer⟩
        ⟨name⟩
            ⟨firstname⟩John⟨/firstname⟩
            ⟨surname⟩Smith⟨/surname⟩
        ⟨/name⟩
    ⟨/buyer⟩
    ⟨property⟩Lot37⟨/property⟩
⟨/auctionreport⟩
```

This report can be represented by the following report atom:

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

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 chapter, the root of the structured news report in each example, and the predicate symbol of the corresponding report atom, is the symbol `report`.

2.3.2 News atoms and access rules

News atoms are ground predicates extracted from report atoms. Access rules extract ground predicates nested within report atoms. These predicates can then be directly used with other information in the background knowledge.

Definition 2.3.3. *Let $\rho(t_1, ..., t_n)$ be a report atom. A* **news atom** *for this report is a literal that represents information in the report atom. A news atom is of the form $p(s_1, ..., s_m)$ where p is a predicate symbol and $s_1, ..., s_m$ are in* $\mathsf{subterms}(t_1) \cup ... \cup \mathsf{subterms}(t_n)$.

The news atoms that are obtained for a given report atom are defined by a set of access rules. Access rules may be manually generated and may be based on information extraction templates used. These templates would define the structure of news atoms, which in turn determines the structure of access rules. Alternatively they may need to be learnt. It is assumed in this thesis that a set of access rules, Γ, is readily available.

Definition 2.3.4. *Let α, $\beta_1 ... \beta_n$ be formulae without implication symbols, $\bar{x}$ be the variables in α, and let the variables in $\beta_1 ... \beta_n$ be a subtuple or equal to $\bar{x}$. A rule that extracts news atoms from a report atom, denoted an* **access rule**, *is a first order formula of the form*

$$\forall \bar{x}\ \alpha \rightarrow \beta_1 \wedge ... \wedge \beta_n$$

Generalised Modus Ponens is used to derive ground news atoms from report atoms and access rules. Modus Ponens is a rule of inference. From a conditional statement and the assertion of that statement's antecedent, it is necessary to conclude that the consequent of that statement holds. Generalised Modus Ponens is a rule of inference that permits groundings in first order formulae. If a grounding Φ exists for the antecedent of a first order formula, such as an access rule, then that grounding is applied to the consequent of that same formula in order to derive the ground consequent.

Definition 2.3.5. *Let Γ be a set of access rules and let ρ be a report atom.* $\mathsf{access}(\rho, \Gamma)$ *is the smallest set of news atoms obtained by exhaustively applying the report atom ρ for a structured*

news report to the access rules in Γ *using generalised modus ponens and conjunction elimination as defined below.*

$$\mathsf{access}(\rho, \Gamma) = \{\mathsf{ground}(\beta_1, \Phi), ..., \mathsf{ground}(\beta_n, \Phi) \mid \forall x_1, ..., x_k\ \alpha \rightarrow \beta_1 \wedge ... \wedge \beta_n \in \Gamma \\ \textit{and}\ \mathsf{ground}(\alpha, \Phi)\ \textit{is}\ \rho\}$$

Example 2.3.3. *Let* ρ *be the following report:*

$$\mathtt{report(forecast(date(25July01), city(Malaga), today(sunny)))}$$

Suppose the set of access rules, Γ*, contains the following:*

$$\forall \mathtt{x,y,z}\ \mathtt{report(forecast(date(x), city(y), today(z)))} \rightarrow \\ \mathtt{date(x)} \wedge \mathtt{location(y)} \wedge \mathtt{today(z)}$$

Γ *applied to* ρ *results in:*

$$\mathtt{date(25July2001), location(Malaga), today(sunny)} \in \mathsf{Access}(\rho, \Gamma)$$

Suppose Γ *also contains the following access rule:*

$$\forall \mathtt{x,y}\ \mathtt{report(forecast(date(x), city(y), today(sunny)))} \rightarrow \mathtt{sunnyWarning(x,y)}$$

As a result,

$$\mathtt{sunnyWarning(25July2001, Malaga)} \in \mathsf{Access}(\rho, \Gamma)$$

Text entries in structured news reports are usually 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.). This heterogeneity can be handled in logic by various kinds of equivalence axioms [Hun00a, Hun02a, Hun02c]. To simplify exposition in this thesis, it is assumed that a preprocessor will convert these text entries into a standard format, that is, the input to an EVA system is homogenous. If entries are not homogenous, inconsistencies may arise that are "noise", rather than the result of unexpected news. It is assumed that the application of access rules to news reports always results in a consistent set of news atoms.

2.4 Background knowledge

Background knowledge consists of rules and facts that can be considered alongside information from news reports in order to provide context. Background knowledge may come from a variety of sources including databases, domain experts, machine learning and earlier news reports. Only the background knowledge that is *relevant* to a particular report is considered when identifying inconsistencies with expectations. Section 2.4.4 formalises relevance and defines a *representative set* as a consistent set of facts derived from background knowledge and a news report. Background knowledge includes domain facts, domain rules and an event model. These are defined in this section. It is assumed that the set of facts derived from domain knowledge is internally consistent, and that it also consistent with the set access(ρ, Γ).

For many domains, background knowledge is readily available. Facts may be extracted from databases, such as the Kompass database of businesses, the ABI/Inform database of companies and the Economist Intelligence Unit countries database. Domain rules are readily available in the form of regulations, laws, and ontologies. Event models can be automatically constructed from reports previously received by an EVA system.

Background knowledge consists of relatively static information. That is, we expect that background rules and facts do not need to be changed regularly. Information that is more dynamic is generated from news reports by means of one or more event models (see Section 2.4.3). Domain rules also differ from access rules, in that access rules are used to transform news atoms into predicates that can be used for further reasoning, whereas domain rules are used to generate new knowledge.

2.4.1 Domain facts

Domain facts may 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 2.4.1. *The* **domain facts** *are a set of ground literals (i.e. atoms and negated atoms).*

Example 2.4.1. *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)
```

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. 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. The implementation details are not considered further in this thesis.

2.4.2 Domain rules

Domain rules may come from several sources including machine learning and domain experts. Domain rules differ from expectations in that they are "hard rules" that are considered to be inviolable.

Definition 2.4.2. *Let α and β be unconditional formulae, $\bar{x}$ be the variables in α, and let the variables in β be a subtuple or equal to $\bar{x}$. A domain rule is a formula of the form*

$$\forall \bar{x}\ \alpha \rightarrow \beta$$

Example 2.4.2. *The domain rule that all companies in the FTSE 100 share index are public limited companies (PLCs) could be expressed as follows:*

$$\forall \mathtt{x}\ \mathtt{ftse100(x)} \rightarrow \mathtt{plc(x)}$$

The domain rule that companies do not launch takeover bids for companies they already own could be expressed as:

$$\forall \mathtt{x,y}\ \mathtt{takeover(x,y)} \rightarrow \neg \mathtt{owns(x,y)}$$

Domain rules are also ideal for representing ontologies. Special predicate symbols define relations between entities, such as `partOf`, `typeOf`. In several domains there are already standard ontologies. For example, in the business domain, Standard Industry Classification (SIC) codes are used to classify sectors of industrial activity.

Example 2.4.3. *The following are domain rules that can be derived from the SIC codes:*

$$\forall \mathtt{x}\ \mathtt{softwareCompany(x)} \rightarrow \mathtt{computerProgrammingAndDataProcessingCompany(x)}$$
$$\forall \mathtt{x}\ \mathtt{computerProgrammingAndDataProcessingCompany(x)} \rightarrow \mathtt{businessServicesCompany(x)}$$
$$\forall \mathtt{x}\ \mathtt{businessServicesCompany(x)} \rightarrow \mathtt{serviceCompany(x)}$$

Example 2.4.4. *The following are domain rules that could apply to the mergers and acquisitions domain:*

$$1)\forall \mathtt{x}\ \mathtt{ftse100(x)} \rightarrow \mathtt{listed(x)}$$
$$2)\forall \mathtt{x,y}\ \mathtt{buyer(x)} \wedge \mathtt{target(y)} \rightarrow \neg\mathtt{owns(x,y)}$$

Rule 1 states that if company x *is one of the FTSE100 (Financial Times Share Index top 100) companies then company* x *must be a "listed" company, that is, listed on the stock exchange. Rule 2 states that if company* x *wishes to buy company* y *then company* x *does not currently own company* y.

2.4.3 The event model

Reports do not normally exist in isolation. There is an underlying narrative which concerns a number of entities that are related in some way over a period of time. In most domains, reports form narratives such that each report tells part of an ongoing story. In the mergers and acquisitions domain for example, a narrative may begin with rumours of an impending bid, continue with news of a bid being made, then go on through the negotiations until the bid is finally agreed on or rejected. All reports are part of at least one narrative and all narratives contain at least one report. Some approach that captures and reasons with these narratives is a necessary part of the background knowledge.

Narratives link entities, events and states. Whilst some expectations are applicable to all entities in all states, others are only applicable to entities in certain states. For example, companies in the state of bankruptcy are expected not to launch takeover bids. In order to apply this expectation it is necessary to know whether an entity is in the state of being bankrupt. This requires an event model, or some similar way of representing the order of events and states in first order logic.

There are certain words and phrases in news reports that indicate events that change the state of entities. 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 (for example, "shareholders will approve" indicates a different state to "shareholders have approved").

The event calculus introduced by Kowalski and Sergot [KS85] is one approach to capturing and reasoning with events in news. There are other approaches to dealing with state and event information but the event calculus has the advantages of being appropriate for a Prolog implementation, of not being subject to the frame problem and of having clear semantics. The event model is modular and as such can be replaced by any other approach that results in a model that can be interrogated by event queries. The event model is presented in detail in Chapter 7.

From an event model it must be possible to derive facts that record which states hold at which timepoints. The example below demonstrates how this could be achieved using the event calculus. The details of this example will be explored in Chapter 7. Informally, states are descriptions of the position in which an entity may find itself: bankrupt, subject to a takeover bid, profitable and so on. Entities are the elements of the domain that can perform some action or be in some state.

Definition 2.4.3. *Let a* **state** *be a ground predicate that denotes the state of one or more entities and a timepoint be a constant that represents a time point of any granularity, such as a date or a time.* `holdsAt` *is a meta level predicate that relates states to timeperiods. For some state, s, and some timepoint of any granularity, t,* $\texttt{holdsAt}(s, t)$ *means that state s holds at timepoint t.*

Example 2.4.5. *Let* $\texttt{t} = 15/12/03$ *and let* $\texttt{s} = \texttt{takingOver}(\texttt{morrisons}, \texttt{safeway})$. *The ground predicate* $\texttt{holdsAt}(\texttt{takingOver}(\texttt{morrisons}, \texttt{safeway}), 15/12/03)$ *indicates that on the 15th December 2003, Morrisons was in the process of mounting a takeover bid for Safeway.*

Let $\texttt{t} = 2004$ *and let* $\texttt{s} = \texttt{primeMinister}(\texttt{tonyBlair}, \texttt{uk})$.
$\texttt{holdsAt}(\texttt{primeMinister}(\texttt{tonyBlair}, \texttt{uk}), 2004)$ *indicates that Tony Blair was the UK Prime Minister during 2004.*

In this example we see that a timepoint may have non-zero duration. A timepoint of granularity g has duration g. However, duration can also be bounded by timepoints of lesser granularity. For example, the timepoint 2004 has the same duration as the period bounded by the timepoints `01/01/2004` and `31/12/2004`. Duration and granularity will not be discussed further in this chapter.

The `holdsAt` predicates can be incorporated in expectations to restrict that expectation's applicability to only those entities in a given state as in the following example:

Example 2.4.6. *The following expectation states that a company that has launched a takeover bid is expected to be profitable:*

$$\forall \texttt{c}_1, \texttt{c}_2 \in \texttt{companies}, \texttt{t} \in \texttt{times}\ \texttt{holdsAt}(\texttt{biddingFor}(\texttt{c}_1, \texttt{c}_2), \texttt{t}) \rightarrow \texttt{holdsAt}(\texttt{profitable}(\texttt{c}_1), \texttt{t})$$

The expectation given in the above example demonstrates clearly the way in which the event calculus meta-predicates can be used in expectations. The event calculus is only one way of creating an event model. There may also be other specialised subsets of the background knowledge that would benefit from a similar formalism expressed using meta-predicates. Again, this is beyond the scope of this particular thesis.

2.4.4 Representative sets

When assessing the potential violation of an expectation by a report it is necessary to ensure that only the subset of the background knowledge that is somehow *relevant* to that report is considered. Let Δ be the set of background knowledge. The function $\mathsf{match}(\rho, \Gamma, \Delta)$ returns the subset of Δ whose members are all the facts in Δ that concern the entities in ρ. Informally, entities are things with a distinct, not necessarily concrete, existence. Marks and Spencer plc, the FTSE 100 share index and Britain are examples of entities. Distinct from entities are things such as attributes, such as monetary values for example, and actions or events, such as the making of a bid. Entities are also used to identify *cohorts*, as defined in Section 3.3.1.

Definition 2.4.4. *Let ρ be a report, let Γ be a set of access rules and let Δ be a set of background knowledge. The set* $\mathsf{match}(\rho, \Gamma, \Delta)$ *is a set of literals that can be derived from Δ and ρ such that there are one or more constants in each literal that appear as constants in one or more literals in* $\mathsf{access}(\rho, \Gamma)$*:*

$$\begin{aligned}\mathsf{match}(\rho, \Gamma, \Delta) = \{&\alpha(p_1, ..., p_j) | \mathsf{access}(\rho, \Gamma) \cup \Delta \vdash \alpha(p_1, ..., p_j) \text{ and} \\ &\text{there exists } \beta(q_1, ..., q_k) \in \mathsf{access}(\rho, \Gamma) \\ &\text{such that there exists } p_h \in p_1, ..., p_j \text{ and } q_i \in q_1, ..., q_k \text{ where } p_h = q_i\}\end{aligned}$$

Example 2.4.7. *Let ρ be a report and let Γ be a set of access rules. Let* $\mathsf{access}(\rho, \Gamma) = \{\mathtt{profitable(ba)}\}$. *Let Δ be a set of background knowledge. Let*

$$\begin{aligned}\Delta = \{&\mathtt{airline(ba)}, \mathtt{airline(ryanair)}, \\ &\mathtt{british(ba)}, \mathtt{british(marksandspencer)}, \\ &\mathtt{irish(ryanair)}, \mathtt{retailer(marksandspencer)}, \\ &\forall \mathtt{x profitable(x)} \rightarrow \neg \mathtt{bankrupt(x)}\}\end{aligned}$$

$\mathsf{match}(\rho, \Gamma, \Delta)$ *contains all literals in Δ which share a constant symbol with the literals in* $\mathsf{access}(\rho, \Gamma)$. *Therefore* $\mathsf{match}(\rho, \Gamma, \Delta) = \{\mathtt{airline(ba)}, \mathtt{british(ba)}, \neg\mathtt{bankrupt(ba)}\}$.

The weakness of this definition arises from the fact that constant symbols that are not entities may incorrectly add to the set $\mathsf{match}(\rho, \Gamma, \Delta)$ as the following example demonstrates:

Example 2.4.8. *Let*

$$\mathsf{access}(\rho, \Gamma) = \{\mathtt{buyer(ryanair)}, \mathtt{target(buzz)}, \mathtt{bidvalue(GBP15.6m)}\}$$

Let

$$\{\mathtt{shareIssue(wellingtonUnderwriting, GBP15.6m)}\} \subseteq \Delta$$

$\mathsf{match}(\rho, \Gamma)$ *will include the fact* $\mathtt{shareIssue(wellingtonUnderwriting, GBP15.6m)}$. *However, the only relevance of this fact to the report ρ is that the share issue value of Wellington*

Underwriting PLC is identical to the value of the bid made by Ryanair for Buzz.

No study has been undertaken to determine the extent of this problem. However, this weakness could, at least partially, be overcome by typing constant symbols and only identifying matches between constant symbols of certain types (e.g. company names, industry sectors) and excluding matches of other types (e.g. dates, values). It is also possible that techniques of assessing relevance, from fields such as information retrieval, may lead to a better $\mathsf{match}(\rho, \Gamma)$ function. This issue will not be addressed further in this thesis.

It is now possible to define a set that contains all the facts in a report ρ and the set of background knowledge that is relevant to ρ. The set $\mathsf{representatives}(\rho, \Gamma, \Delta)$ is the union of the literals extracted from a report and the subset of relevant literals from Δ. $\mathsf{representatives}(\rho, \Gamma, \Delta)$ is a first order representation of the situation which is in the report.

Definition 2.4.5. *Let ρ be a report, Γ be a set of access rules and Δ be a set of background knowledge. The set of all facts relevant to ρ, the* **representative set** *is a set such that:*

$$\mathsf{representatives}(\rho, \Gamma, \Delta) = \mathsf{access}(\rho, \Gamma) \cup \mathsf{match}(\rho, \Gamma, \Delta)$$

We assume that $\mathsf{representatives}(\rho, \Gamma, \Delta)$ *is a consistent set.*

Example 2.4.9. *Let ρ be a report and let Γ be a set of access rules. Let*

$$\mathsf{access}(\rho, \Gamma) = \{\mathtt{airline(ryanair)}, \mathtt{takeover(ryanair, buzz)}\}$$

and let

$$\mathsf{match}(\rho, \Gamma, \Delta) = \{\mathtt{irish(ryanair)}, \mathtt{profitable(ryanair)}, \neg\mathtt{profitable(buzz)},\}$$

The set

$$\begin{aligned}\mathsf{representatives}(\rho, \Gamma, \Delta) = \{&\mathtt{airline(ryanair)}, \mathtt{takeover(ryanair, buzz)},\\ &\mathtt{irish(ryanair)}, \mathtt{profitable(ryanair)}, \neg\mathtt{profitable(buzz)}\}\end{aligned}$$

2.5 Expectations about news

Expectations are formulae that capture general patterns concerning news reports and background knowledge. The set of expectations is at the heart of an EVA framework as it is central to the identification of unexpected information. Unlike background rules, which it is assumed are always correct, it is assumed expectations will be violated by representative sets 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 2.5.1. *Let $\alpha_1, ..., \alpha_n, \beta$ be unground literals and $\bar{x}$ be free variables occurring in the literals. An* **expectation** *is a formula of the following form*

$$\forall \bar{x}\ \alpha_1 \wedge ... \wedge \alpha_n \rightarrow \beta$$

where for all Φ, if ground$(\alpha_1 \wedge ... \wedge \alpha_n, \Phi)$ *is a ground formula, then* ground(β, Φ) *is a ground formula.*

Examples of expectations are given in Example 2.5.1 and Example 2.5.2 below.

Example 2.5.1. *The expectation "A company is expected to have sufficient available capital to be able to cover a bid" can be represented as follows.*

$$\forall \mathtt{c} \in \mathtt{companies}, \mathtt{v} \in \mathtt{bidValue}, \mathtt{m} \in \mathtt{monetaryValue}$$
$$\mathtt{bidTendered(c,v)} \wedge \mathtt{availableCapital(c,m)} \rightarrow \mathtt{m} > \mathtt{v}$$

Example 2.5.2. *The expectation "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 as follows.*

$$\forall \mathtt{x}, \mathtt{y} \in \mathtt{companies}\ \mathtt{target(x,y)} \rightarrow$$
$$(\mathtt{sector(x)} = \mathtt{sector(y)} \vee \mathtt{supplier(sector(x), sector(y))} \vee$$
$$\mathtt{supplier(sector(y), sector(x))} \vee \mathtt{compatible(sector(y), sector(x))})$$

Good expectations are those that are rarely violated by news reports. The more accurately an expectation reflects the behaviour of entities in the real world, the more interesting the news is that violates it. A method for generating a set of good expectations with respect to the information in news reports is presented in subsequent chapters.

2.6 An example user interface for an EVA system

The example user interface in Figure 2.2 demonstrates the various types of knowledge in the EVA framework. This is a non functional prototype and is only one example of an interface for an EVA system. It is included here to demonstrate the potential uses of the different sets of knowledge in the EVA framework. An EVA system would continually filter news from a live feed. News, in structured text form, would be converted to logic. A representative set would then be generated from the news report and relevant background knowledge. The representative set would then be analysed to identify which, if any, expectations it violates. The resulting set of news, background knowledge and expectations can be presented to the user. One example interface is shown below.

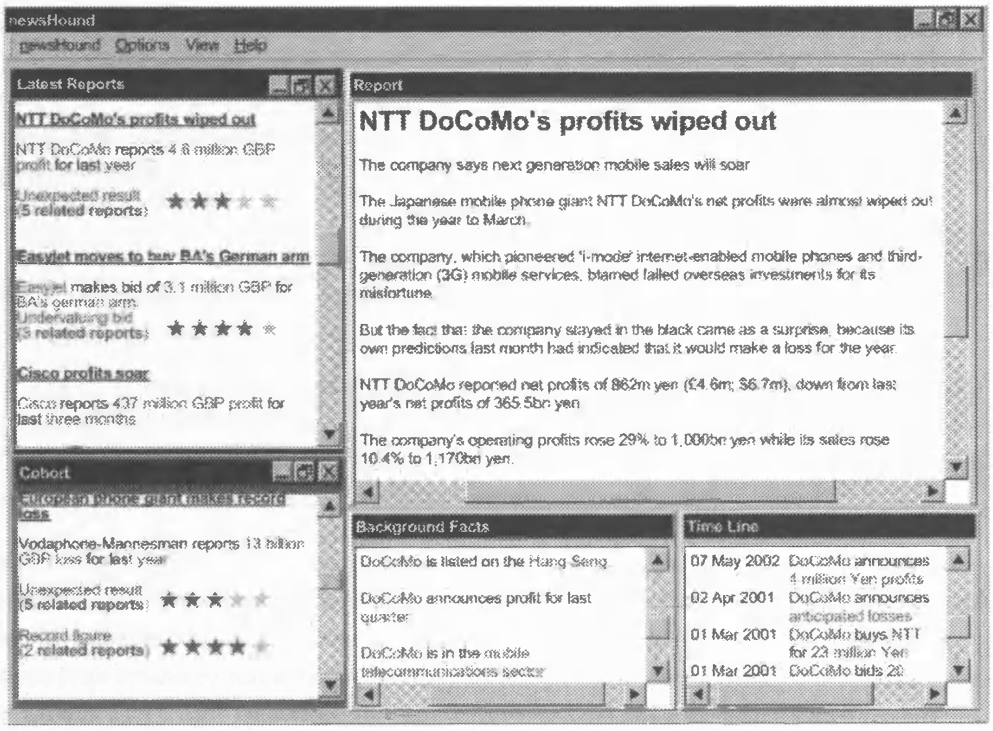


Figure 2.2: An example graphical user interface for an EVA system created by the author

In the above example it is assumed that the user wishes to prioritise their reading of news such that the most unexpected news is displayed first. They may wish to read the entire report, and they may require contextual information and explanation as to why the report has been recommended. The user may be able to set parameters, such as the cut off point for unexpectedness, beyond which reports will not be displayed, or the length of timeline given.

In this example, the information presented is displayed in a number of panes. The Report pane shows either the original, free text news report, or else a version created from the structured text by a natural language generator. The Latest Reports pane shows the list of interesting reports given as a headline, a one sentence summary and a short phrase that identifies the expectation that has been violated. For example the first story is interesting because it is an "Unexpected result". This pane gives a star value to each report, indicating the degree of interest arising from that violation. The provenance of this value will be defined in the next chapter. All the user needs to be aware of is that the higher the star rating, the more interesting the report.

Background information is also presented to the user. The Cohort pane contains information about other entities that have recently violated the same expectation. Cohort violations are defined in detail in Section 3.3. Information about cohort violations allows the user to examine emerging trends. Finally, relevant background knowledge is presented in two panes at the bottom of the

window. The Background Facts pane contains general contextual information that is relevant to the report. This information is a subset of $\mathsf{match}(\rho, \Gamma, \Delta)$ and excludes information from the event model. The information from the event model is presented in the Time Line pane. The example interface shown here presents the user with information from all of the sets of knowledge in the EVA framework.

2.7 Discussion of Chapter 2

In Chapter 1 it was argued that there is a need for a framework that supports the development of a system that can identify and rate unexpectedness in news reports. This chapter has introduced the EVA framework, which addresses that need. The framework contains three sets of information: news reports, background knowledge and expectations.

Expectations are rules that reflect the expected events for a domain. Background knowledge consists of rules and facts that provide context for information in news reports. The set of background knowledge includes an event model that supports reasoning about states and events. News reports come from structured text news and are converted into logical facts. These facts are then combined with background knowledge and compared with expectations in order to identify interesting information.

Chapter 3

Expectations in the EVA Framework

Inconsistency is usually regarded as something to be avoided in logic. As a result of *ex falso quodlibet*, reasoning from an inconsistent set of formulae leads to unsafe conclusions. The least sophisticated method for dealing with inconsistency is to treat it as an *error*, and to avoid reasoning from inconsistent sets of data. More sophisticated approaches such as truth maintenance and belief revision aim to *eradicate* inconsistency, thus returning the data to a state from which reasoning can take place. However, the EVA framework proposed in this thesis *exploits* inconsistency. Inconsistency between facts and an expectation is harnessed as an indicator of an unexpected event.

The EVA framework uses unexpectedness as a measure of interest. Unexpected information is information that violates an expectation that more or less accurately represents the real world events. An expectation is violated by information that confirms the expectations's antecedent whilst negating its consequent. As a measure, unexpectedness has several advantages:

1. Unexpectedness has clear semantics: the violation of an accurate expectation clearly indicates an unexpected event. Such an indication is Boolean in nature (a report violates an expectation or it does not) but as we shall see later, the *strength* of the expectation that is violated can be used to rate the interest-value of a given report.
2. When identifying unexpectedness, it is possible to also take account of information in a set of background knowledge. This is useful, as background knowledge provides the context that supports a wider understanding of the information in news reports.
3. Expectations can be based on real data: the same reports that are analysed for interest can later be used as a basis from which an EVA system can learn what is and is not expected.

This is a significant advantage to the EVA framework: an EVA system would incorporate a self updating, autonomous expectation generator similar to that presented in Chapters 5 and 6.

In order to identify unexpected information it is necessary to know what is and is not expected. To this end, the EVA framework, as defined in Chapter 2, includes a set of expectations. Unexpected information is discovered when a report, along with a relevant subset of the background knowledge, violates an accurate expectation. In this chapter, Section 3.1 defines the relations between representative sets and expectations. Section 3.2 defines the way in which the strength of expectations is measured. Section 3.3 defines special types of violations of expectations and discusses their significance. Section 3.4 introduces a method of checking for inconsistency between representative sets and expectations that is computationally viable.

3.1 Relationships between reports, background knowledge and expectations

The notion that interest is identified when expectations are violated was informally introduced in Section 2.1. This section now presents definitions of violations of expectations as well as firings and confirmations of expectations. Each report results in a representative set, $\mathsf{representatives}(\rho, \Gamma, \Delta)$, as in Definition 2.4.5. Each expectation, ϵ, consists of an antecedent and a consequent. It is necessary to identify when a representative set implies a ground version of an expectation's antecedent, a ground version of an expectation's consequent or a ground version of the negation of an expectation's consequent. The representative set may also imply a ground version of the negation of the antecedent. However this does not convey any information other than that the expectation does not apply to the report in question. Therefore, the implication of the negation of the antecedent is not of interest.

It is sometimes necessary to consider the antecedent or the consequent of an expectation in isolation. The functions $\mathsf{antecedent}(\epsilon)$ and $\mathsf{consequent}(\epsilon)$ are defined for convenience of notation.

Definition 3.1.1. *Let E be a set of expectations and $\epsilon = \forall \bar{x}\ \alpha \rightarrow \beta$ be an expectation in E.* $\mathsf{antecedent}(\epsilon)$ *is a function from any expectation ϵ, or any ground expectation* $\mathsf{ground}(\Phi, \epsilon)$*, to the set of all formulae in the language such that*

$$\mathsf{antecedent}(\forall \bar{x}\ \alpha \rightarrow \beta) = \forall \bar{x}\ \alpha$$

where the universal quantifier is omitted if ϵ is ground.

Example 3.1.1. *Let* $\epsilon_1 = \forall \mathtt{c}, \mathtt{s}\ \mathtt{sector}(\mathtt{c}, \mathtt{s}) \rightarrow \mathtt{profitable}(\mathtt{c})$ *be an expectation in E. It follows that* $\mathsf{antecedent}(\epsilon_1) = \forall \mathtt{c}, \mathtt{s}\ \mathtt{sector}(\mathtt{c}, \mathtt{s})$.

Let Φ be a grounding set $\Phi = \{\mathtt{c} = \mathtt{ryanair}, \mathtt{s} = \mathtt{airline}\}$. *The formula* $\mathsf{ground}(\Phi, \epsilon) = \mathtt{sector(ryanair, airline)} \rightarrow \mathtt{profitable(ryanair)}$. *As a result,*

$$\mathsf{antecedent}(\mathsf{ground}(\Phi, \epsilon)) = \mathtt{sector(ryanair, airline)}$$

Definition 3.1.2. *Let E be a set of expectations and $\epsilon = \forall \bar{x}\ \alpha \rightarrow \beta$ be an expectation in E.* $\mathsf{consequent}(\epsilon)$ *is a function from any expectation ϵ, or any ground expectation* $\mathsf{ground}(\Phi, \epsilon)$, *to the set of all formulae in the language such that*

$$\mathsf{consequent}(\forall \bar{x}\ \alpha \rightarrow \beta) = \forall \bar{x}\ \beta$$

where the universal quantifier is omitted if ϵ is ground.

Example 3.1.2. *Let* $\epsilon = \forall \mathtt{c}, \mathtt{s}\ \mathtt{sector(c, s)} \rightarrow \mathtt{profitable(c)}$. *It follows that* $\mathsf{consequent}(\epsilon) = \forall \mathtt{c}, \mathtt{s}\ \mathtt{profitable(c)}$.

Let Φ be a grounding set $\Phi = \{\mathtt{c} = \mathtt{ryanair}, \mathtt{s} = \mathtt{airline}\}$. *The formula* $\mathsf{ground}(\Phi, \epsilon) = \mathtt{sector(ryanair, airline)} \rightarrow \mathtt{profitable(ryanair)}$. *As a result,*

$$\mathsf{consequent}(\mathsf{ground}(\Phi, \epsilon)) = \mathtt{profitable(ryanair)}$$

Note that the set of all universally quantified variables is used for both the antecedent and consequent of the expectation. This is to avoid the necessity of identifying which variables appear in the consequent and which appear only in the antecedent.

If $\mathsf{representatives}(\rho, \Gamma, \Delta)$ implies a ground version of $\mathsf{antecedent}(\epsilon)$ then we say that ρ *fires* that expectation.

Definition 3.1.3. *Let ϵ be an expectation, Γ be a set of access rules, Δ be a set of background knowledge and ρ be a report. ϵ is* **fired** *by ρ iff there exists a grounding set Φ such that* $\mathsf{ground}(\epsilon, \Phi) = \epsilon_\Phi$ *and*

$$\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\epsilon_\Phi)$$

Example 3.1.3. *Let ϵ be the expectation*

$$\forall \mathtt{x}\ \mathtt{british(x)} \wedge \mathtt{departmentStore(x)} \rightarrow \mathtt{takeoverTarget(x)}$$

Let Γ be a set of access rules, Δ be a set of background knowledge and ρ be a report. Let Φ be a

grounding set, $\{\mathtt{x} = \mathtt{marks\&spencer}\}$.

Let : $\mathsf{ground}(\epsilon, \Phi) =$
$\mathtt{british(marks\&spencer)} \wedge \mathtt{departmentStore(marks\&spencer)} \rightarrow$
$\mathtt{takeoverTarget(marks\&spencer)}$

$\mathsf{access}(\rho, \Gamma) = \{\neg\mathtt{takeoverTarget(marks\&spencer)}\}$

$\mathsf{match}(\rho, \Gamma, \Delta) =$
$\{\mathtt{departmentStore(marks\&spencer)}, \mathtt{british(marks\&spencer)}\}$

$\mathsf{representatives}(\rho, \Gamma, \Delta) =$
$\{\neg\mathtt{takeoverTarget(marks\&spencer)}\} \cup$
$\{\mathtt{departmentStore(marks\&spencer)}, \mathtt{british(marks\&spencer)}\}$

$\mathsf{ground}(\mathsf{antecedent}(\epsilon), \Phi) =$
$\mathtt{british(marks\&spencer)} \wedge \mathtt{departmentStore(marks\&spencer)}$

As a result, $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\epsilon_\Phi)$ *therefore* ρ *fires* ϵ.

Note that neither the consequent of ϵ nor its negation need be implied by $\mathsf{representatives}(\rho, \Gamma, \Delta)$ in order for ϵ to be fired.

If $\mathsf{representatives}(\rho, \Gamma, \Delta)$ implies a ground version of $\neg\mathsf{consequent}(\epsilon)$ then we say that ρ *attacks* that expectation.

Definition 3.1.4. *Let* ϵ *be an expectation,* Γ *be a set of access rules,* Δ *be a set of background knowledge,* ρ *be a report and* Φ *be a grounding set.* $\mathsf{ground}(\epsilon, \Phi)$ *is* **attacked** *by a report* ρ *iff there exists a grounding set* Φ *such that* $\mathsf{ground}(\epsilon, \Phi) = \epsilon_\Phi$ *and*

$$\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \neg\mathsf{consequent}(\epsilon_\Phi)$$

Example 3.1.4. *Let* ϵ *be an expectation* $\forall\mathtt{x}\ \mathtt{airline(x)} \vee \mathtt{bank(x)} \rightarrow \mathtt{profitable(x)}$, Γ *be a set of access rules,* Δ *be a set of background knowledge,* ρ *be a report and* Φ *be a grounding set,* $\{\mathtt{x} = \mathtt{marks\&spencer}\}$.

$\mathsf{ground}(\epsilon, \Phi) = \mathtt{airline(marks\&spencer)} \vee \mathtt{bank(marks\&spencer)} \rightarrow \mathtt{profitable(marks\&spencer)}$

Let

$$\mathsf{access}(\rho, \Gamma) = \{\neg\texttt{profitable(marks\&spencer)}\}$$

$$\mathsf{match}(\rho, \Gamma, \Delta) = \{\texttt{retail(marks\&spencer)}\}$$

$$\mathsf{representatives}(\rho, \Gamma, \Delta) = \{\neg\texttt{profitable(marks\&spencer)}\} \cup \{\texttt{retail(marks\&spencer)}\}$$

$$\mathsf{ground}(\mathsf{consequent}(\epsilon), \Phi) = \texttt{profitable(marks\&spencer)}$$

As a result, $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \neg\mathsf{consequent}(\epsilon_\Phi)$ *therefore* ρ *attacks* ϵ.

Note that the antecedent of ϵ need not be fired in order for $\mathsf{consequent}(\epsilon)$ to be attacked. Also, $\mathsf{representatives}(\rho, \Gamma, \Delta)$ is consistent but if ρ attacks ϵ then the union of $\mathsf{representatives}(\rho, \Gamma, \Delta)$ and $\{\mathsf{consequent}(\epsilon_\Phi)\}$ will be inconsistent. The inconsistency arises from the inclusion of the ground consequent, $\mathsf{consequent}(\epsilon_\Phi)$.

If $\mathsf{representatives}(\rho, \Gamma, \Delta)$ implies a ground version of $\mathsf{consequent}(\epsilon)$ then we say that ρ *supports* that expectation.

Definition 3.1.5. *Let* ϵ *be an expectation,* Γ *be a set of access rules,* Δ *be the set of background knowledge,* ρ *be a report and* Φ *be a grounding set. The expectation* $\mathsf{ground}(\epsilon, \Phi)$ *is* **supported** *by a report* ρ *iff there exists a grounding set* Φ *such that* $\mathsf{ground}(\epsilon, \Phi) = \epsilon_\Phi$ *and*

$$\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{consequent}(\epsilon_\Phi)$$

Example 3.1.5. *Let* ϵ *be an expectation* $\forall\texttt{x}\ \texttt{airline(x)} \vee \texttt{bank(x)} \rightarrow \texttt{profitable(x)}$, Γ *be a set of access rules,* Δ *be a set of background knowledge and* ρ *be a report. Let* Φ *be a grounding set,*$\{\texttt{x} = \texttt{ryanair}\}$.

$$\mathsf{ground}(\epsilon, \Phi) = \texttt{airline(ryanair)} \vee \texttt{bank(ryanair)} \rightarrow \texttt{profitable(ryanair)}$$

Let $\mathsf{access}(\rho, \Gamma) = \{\texttt{profitable(ryanair)}\}$

$$\mathsf{match}(\rho, \Gamma, \Delta) = \{\texttt{airline(ryanair)}\}$$

$$\mathsf{representatives}(\rho, \Gamma, \Delta) = \{\texttt{profitable(ryanair)}\} \cup \{\texttt{airline(ryanair)}\}$$

$$\mathsf{ground}(\mathsf{consequent}(\epsilon), \Phi) = \texttt{profitable(ryanair)}$$

$\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{consequent}(\epsilon_\Phi)$, *therefore* ρ *supports* ϵ.

Note that the antecedent of ϵ need not be fired in order that the consequent ϵ be supported.

An EVA system identifies as interesting information that which violates an expectation. In order for an expectation to be violated, it must be both fired and attacked by the same representative set.

Definition 3.1.6. *Let ϵ be an expectation, let Γ be a set of access rules, Δ be a set of background knowledge and ρ be a report. Let Φ be a grounding set. ϵ is* **violated** *by a report ρ iff there is a grounding set Φ such that* $\mathsf{ground}(\epsilon, \Phi) = \epsilon_\Phi$ *and*

$$\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\epsilon_\Phi) \wedge \neg \mathsf{consequent}(\epsilon_\Phi)$$

Example 3.1.6. *Let ϵ be an expectation* $\forall \mathtt{x}\ \mathtt{airline(x)} \vee \mathtt{bank(x)} \rightarrow \mathtt{profitable(x)}$, *$\Gamma$ be a set of access rules, Δ be a set of background knowledge and ρ be a report. Let Φ be a grounding set,* $\{\mathtt{x} = \mathtt{buzz}\}$. $\mathsf{ground}(\epsilon, \Phi) = \mathtt{airline(buzz)} \vee \mathtt{bank(buzz)} \rightarrow \mathtt{profitable(buzz)}$ *and*

$$\mathsf{access}(\rho, \Gamma) = \{\neg\mathtt{profitable(buzz)}\}$$
$$\mathsf{match}(\rho, \Gamma, \Delta) = \{\mathtt{airline(buzz)}\}$$

$$\mathsf{representatives}(\rho, \Gamma, \Delta) = \{\neg\mathtt{profitable(buzz)}\} \cup \{\mathtt{airline(buzz)}\}$$

$$\mathsf{ground}(\mathsf{antecedent}(\epsilon) = \mathtt{airline(buzz)} \vee \mathtt{bank(buzz)})$$

$$\mathsf{ground}(\mathsf{consequent}(\epsilon), \Phi) = \mathtt{profitable(buzz)}$$

It then follows that

$$\{\neg\mathtt{profitable(buzz)}, \mathtt{airline(buzz)}\} \vdash$$
$$\mathtt{airline(buzz)} \vee \mathtt{bank(buzz)} \wedge \neg\mathtt{profitable(buzz)}$$

therefore ρ violates ϵ.

If a report both fires and supports an expectation then that report **confirms** that expectation is, in this case, representative of the real world.

Definition 3.1.7. *Let ϵ be an expectation, let Γ be a set of access rules, Δ be a set of background knowledge and ρ be a report. ϵ is* **confirmed** *by a report ρ iff there is a grounding set Φ such that* $\mathsf{ground}(\epsilon, \Phi) = \epsilon_\Phi$ *and*

$$\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\epsilon_\Phi) \wedge \mathsf{consequent}(\epsilon_\Phi)$$

Example 3.1.7. *Let ϵ be an expectation* $\forall \mathtt{x}\ \mathtt{airline(x)} \vee \mathtt{bank(x)} \rightarrow \mathtt{profitable(x)}$, *$\Gamma$ be a set of access rules, Δ be a set of background knowledge and ρ be a report. Let Φ be a grounding*

set, $\{\mathtt{x} = \mathtt{ryanair}\}$. *It then follows that*

$$\mathsf{ground}(\epsilon, \Phi) = \mathtt{airline(ryanair)} \vee \mathtt{bank(ryanair)} \rightarrow \mathtt{profitable(ryanair)}$$

Let:

$$\mathsf{access}(\rho, \Gamma) = \{\mathtt{profitable(ryanair)}\}$$

$$\mathsf{match}(\rho, \Gamma, \Delta) = \{\mathtt{airline(ryanair)}\}$$

$$\mathsf{representatives}(\rho, \Gamma, \Delta) = \{\mathtt{profitable(ryanair)}\} \cup \{\mathtt{airline(ryanair)}\}$$

$$\mathsf{ground}(\mathsf{antecedent}(\epsilon) = \mathtt{airline(ryanair)} \vee \mathtt{bank(ryanair)})$$

$$\mathsf{ground}(\mathsf{consequent}(\epsilon), \Phi) = \mathtt{profitable(ryanair)}$$

$\{\mathtt{profitable(ryanair)}, \mathtt{airline(ryanair)}\} \vdash (\mathtt{airline(ryanair)} \vee \mathtt{bank(ryanair)}) \wedge \mathtt{profitable(ryanair)}$ *therefore* ρ *confirms* ϵ.

3.2 Measuring expectation strength

The EVA framework uses a confirmation measure to rate the strength, accuracy and coverage of expectations. Expectations, by their very nature, may be violated by news reports. In other words, it is possible for a representative set to cause the antecedent and the negation of the consequent of a ground expectation to hold, resulting in an inconsistency with the expectation. The stronger the expectation that is violated, the more interesting that violation is.

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.

Recall from section 2.3.2 that the evidence comes in the form of representative sets. Representative sets are consistent sets of facts derived from the news reports and background knowledge available to an EVA system.

For any two reports ρ_1 and ρ_2 and their associated representative sets $\mathsf{representatives}(\rho_1, \Gamma, \Delta)$ and $\mathsf{representatives}(\rho_2, \Gamma, \Delta)$, if $\mathsf{representatives}(\rho_1, \Gamma, \Delta)$ violates ϵ_1 and $\mathsf{representatives}(\rho_2, \Gamma, \Delta)$ violates ϵ_2, which of ρ_1 and ρ_2 is more unexpected? For each expectation ϵ, there are several measures of interest that depend on the set of reports, Π, the set of access rules, Γ and the set of background knowledge, Δ and that give an indication of the relative strengths of any two expectations. These are defined below.

Three of these measures are dependent on either the antecedent or the consequent of the expectation. The *fired* value for an expectation records how many members in the set of all reports, Π, have led to representative sets that imply a ground version of $\mathsf{antecedent}(\epsilon)$. The *attacked* value for an expectation records how many reports in Π lead to a representative set that implies a ground version of $\neg\mathsf{consequent}(\epsilon)$ and the *supported* value for an expectation records how many reports in Π have lead to a representative set that implies a ground version of $\mathsf{consequent}(\epsilon)$.

The set of reports that leads to the firing of an expectation, denoted $\mathsf{fset}(\epsilon, \Pi, \Gamma, \Delta)$, is defined as follows:

Definition 3.2.1. *Let Γ be a set of access rules, Δ be a set of background knowledge, ϵ be an expectation and Π be a set of reports. The subset of Π whose members fire ϵ, denoted* $\mathsf{fset}(\epsilon, \Pi, \Gamma, \Delta)$ *is as follows:*

$$\mathsf{fset}(\epsilon, \Pi, \Gamma, \Delta) = \{\rho \in \Pi \mid \textit{there exists a grounding set } \Phi \textit{ such that } \mathsf{ground}(\epsilon, \Phi) = \epsilon_\Phi \textit{ and } \mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\epsilon_\Phi)\}$$

For ease of notation in later definitions. $\mathsf{fired}(\epsilon, \Pi, \Gamma, \Delta) = |\mathsf{fset}(\epsilon, \Pi, \Gamma, \Delta)|$

The set of reports that attack an expectation is that expectation's aset. The attacked value for an expectation is dependent on the size of the aset:

Definition 3.2.2. *Let Γ be a set of access rules, Δ be a set of background knowledge, ϵ be an expectation and Π be a set of reports. The subset of Π whose members attack ϵ, denoted* $\mathsf{aset}(\epsilon, \Pi, \Gamma, \Delta)$ *is as follows:*

$$\mathsf{aset}(\epsilon, \Pi, \Gamma, \Delta) = \{\rho \in \Pi \mid \textit{there exists a grounding set } \Phi \textit{ such that } \mathsf{ground}(\epsilon, \Phi) = \epsilon_\Phi \textit{ and } \mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \neg\mathsf{consequent}(\epsilon_\Phi)\}$$

For ease of notation. $\mathsf{attacked}(\epsilon, \Pi, \Gamma, \Delta) = |\mathsf{aset}(\epsilon, \Pi, \Gamma, \Delta)|$.

The set of reports that support an expectation is that expectation's sset. The support value for an expectation is dependent on the size of the sset.

Definition 3.2.3. *Let Γ be a set of access rules, Δ be a set of background knowledge, ϵ be an expectation and Π be a set of reports. The subset of Π whose members support ϵ, denoted* $\mathsf{sset}(\epsilon, \Pi, \Gamma, \Delta)$ *is as follows:*

$$\mathsf{sset}(\epsilon, \Pi, \Gamma, \Delta) = \{\rho \in \Pi \mid \textit{there exists a grounding set } \Phi \textit{ such that } \mathsf{ground}(\epsilon, \Phi) = \epsilon_\Phi \textit{ and } \mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{consequent}(\epsilon_\Phi)\}$$

For ease of notation. $\mathsf{support}(\epsilon, \Pi, \Gamma, \Delta) = |\mathsf{sset}(\epsilon, \Pi, \Gamma, \Delta)|$.

The next two measures to be defined are dependent on the whole of the expectation. The *violated* value for an expectation records how many reports in Π have lead to a representative set that

implies ground(antecedent(ϵ)$\wedge\neg$consequent(ϵ), Φ). The set of reports that violate an expectation is that expectation's vset. The violation value for an expectation is dependent on the size of the vset.

Definition 3.2.4. *Let* Γ *be a set of access rules,* Δ *be a set of background knowledge,* ϵ *be an expectation and* Π *be a set of reports. The subset of* Π *whose members violate* ϵ*, denoted* $\mathsf{vset}(\epsilon, \Pi, \Gamma, \Delta)$ *is as follows:*

$$\mathsf{vset}(\epsilon, \Pi, \Gamma, \Delta) = \{\rho \in \Pi \mid \textit{there exists a grounding set } \Phi \textit{ such that } \mathsf{ground}(\epsilon, \Phi) = \epsilon_\Phi \textit{ and } \mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\epsilon_\Phi) \wedge \neg\mathsf{consequent}(\epsilon_\Phi)\}$$

For ease of notation. $\mathsf{violated}(\epsilon, \Pi, \Gamma, \Delta) = |\mathsf{vset}(\epsilon, \Pi, \Gamma, \Delta)|$

The *confirmed* value for an expectation records how many reports in Π have lead to a representative set that implies ground(antecedent(ϵ) $\wedge$ consequent(ϵ), Φ). The set of reports that confirm an expectation is that expectation's cset. The confirmed value for an expectation is dependent on the size of the cset.

Definition 3.2.5. *Let* Γ *be a set of access rules,* Δ *be a set of background knowledge,* ϵ *be an expectation and* Π *be a set of reports. The subset of* Π *whose members confirm* ϵ*, denoted* $\mathsf{cset}(\epsilon, \Pi, \Gamma, \Delta)$ *is as follows:*

$$\mathsf{cset}(\epsilon, \Pi, \Gamma, \Delta) = \{\rho \in \Pi \mid \textit{there exists a grounding set } \Phi \textit{ such that } \mathsf{ground}(\epsilon, \Phi) = \epsilon_\Phi \textit{ and } \mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\epsilon_\Phi) \wedge \mathsf{consequent}(\epsilon_\Phi)\}$$

For ease of notation. $\mathsf{confirmed}(\epsilon, \Pi, \Gamma, \Delta) = |\mathsf{cset}(\epsilon, \Pi, \Gamma, \Delta)|$

Two further values are expressed as ratios between those values that depend on the whole expectation and those values that depend only on the antecedent. The *accuracy* value measures the strength of an expectation with respect to the number of times it has been fired and not attacked, and the *validity* value measures the strength of an expectation with respect to the number of times it has been fired and confirmed.

Definition 3.2.6. *Let* Γ *be a set of access rules,* Δ *be a set of background knowledge,* ϵ *be an expectation and* Π *be a set of reports. Let* $\mathsf{fired}(\epsilon, \Pi, \Gamma, \Delta)$ *be the number of reports in* Π *that fire the antecedent of* ϵ*. Let* $\mathsf{violated}(\epsilon, \Pi, \Gamma, \Delta)$ *be the number of reports in* Π *that both fire the antecedent of* ϵ *and attack the consequent of* ϵ*, given the same grounding set.*

If $\mathsf{fired}(\epsilon, \Pi, \Gamma, \Delta) > 0$ *then*

$$\mathsf{accuracy}(\epsilon, \Pi, \Gamma, \Delta) = 1 - \frac{\mathsf{violated}(\epsilon, \Pi, \Gamma, \Delta)}{\mathsf{fired}(\epsilon, \Pi, \Gamma, \Delta)}$$

If $\mathsf{fired}(\epsilon, \Pi, \Gamma, \Delta) = 0$ *then* $\mathsf{accuracy}(\epsilon, \Pi, \Gamma, \Delta)$ *is undetermined.*

Definition 3.2.7. *Let* Γ *be a set of access rules,* Δ *be a set of background knowledge,* ϵ *be an expectation and* Π *be a set of reports. Let* $\mathsf{fired}(\epsilon, \Pi, \Gamma, \Delta)$ *be the number of reports in* Π *that fire the antecedent of* ϵ*. Let* $\mathsf{confirmed}(\epsilon, \Pi, \Gamma, \Delta)$ *be the number of reports in* Π *that both fire the antecedent of* ϵ *and support the consequent of* ϵ*, given the same grounding set.*

If $\mathsf{fired}(\epsilon, \Pi, \Gamma, \Delta) > 0$ *then*

$$\mathsf{validity}(\epsilon, \Pi, \Gamma, \Delta) = \frac{\mathsf{confirmed}(\epsilon, \Pi, \Gamma, \Delta)}{\mathsf{fired}(\epsilon, \Pi, \Gamma, \Delta)}$$

If $\mathsf{fired}(\epsilon, \Pi, \Gamma, \Delta) = 0$ *then* $\mathsf{validity}(\epsilon, \Pi, \Gamma, \Delta)$ *is undetermined.*

If the closed world assumption were to be applied to the facts in $\mathsf{representatives}(\rho, \Gamma, \Delta)$ then each report that does not support an expectation will attack it and vice versa. Therefore

$$\mathsf{validity}(\epsilon, \Pi, \Gamma, \Delta) = \mathsf{accuracy}(\epsilon, \Pi, \Gamma, \Delta)$$

However, the closed world assumption will not be used as to do so would lead to unsafe inferences based on what is likely to be incomplete information. Therefore, the validity and accuracy values for any given expectation are not necessarily equal.

Firing, confirmation and accuracy 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. Knowing the order over values of accuracy, validity and coverage can enable us to determine the relative strength of expectations. The ordering is simply the order relation over positive real numbers, for the accuracy and validity values, and natural numbers in the case of the coverage value.

Definition 3.2.8. *Let* $\geq$ *be the usual ordering over real numbers. For any set of expectations* E*, any set of background knowledge,* Δ*, any set of access rules,* Γ *and any set of reports* Π*,* $\epsilon_1 \in E$ *is* **more accurate** *than* $\epsilon_2 \in E$ *iff* $\mathsf{accuracy}(\epsilon_1, \Pi, \Gamma, \Delta) \geq \mathsf{accuracy}(\epsilon_2, \Pi, \Gamma, \Delta)$.

$\epsilon_1 \in E$ *has* **greater validity** *than* $\epsilon_2 \in E$ *iff* $\mathsf{validity}(\epsilon_1, \Pi, \Gamma, \Delta) \geq \mathsf{validity}(\epsilon_2, \Pi, \Gamma, \Delta)$.

$\epsilon_1 \in E$ *has* **better coverage** *than* $\epsilon_2 \in E$ *iff* $\mathsf{fired}(\epsilon_1, \Pi, \Gamma, \Delta) \geq \mathsf{fired}(\epsilon_2, \Pi, \Gamma, \Delta)$.

These three values allow us to determine the strength of an expectation with respect to its ability to identify unexpected information. The coverage value records the number of reports that have lead to that expectation being fired. This is a measure of the amount of "evidence" there is, on which the accuracy and validity values are based. The accuracy value indicates how well the expectation represents the real world (as reported in the news reports), as it is a measure of the proportion of times that the expectation has not been violated. The validity value is a yet more stringent test of whether the expectation represents the real world, as it records the proportion of news reports fire the expectation that also confirm the consequent of that expectation.

Other measures may be needed for specific purposes. For example, if one of the aims was to present explanations then perhaps expectations with the greatest "explanatory power" might be favoured. This may be measured by the number of predicates in the expectation for example. However, for the purposes of determining how well an expectation represents the real world, a combination of validity and coverage or accuracy and coverage is appropriate.

The question at the beginning of this section was: for any two representative sets representatives(ρ_1, Γ, Δ) and representatives(ρ_2, Γ, Δ), if representatives(ρ_1, Γ, Δ) violates ϵ_1 and representatives(ρ_2, Γ, Δ) violates ϵ_2, which of ρ_1 and ρ_2 is more unexpected?

The measures defined above give a partial answer: If accuracy$(\epsilon_2, \Pi, \Gamma, \Delta) \leq$ accuracy$(\epsilon_1, \Pi, \Gamma, \Delta)$ then, all other things being equal, ρ_1 is more interesting than ρ_2. If validity$(\epsilon_1, \Pi, \Gamma, \Delta) \leq$ validity$(\epsilon_2, \Pi, \Gamma, \Delta)$ then, all other things being equal ρ_1 is less interesting than ρ_2. If fired$(\epsilon_1, \Pi, \Gamma, \Delta) \leq$ fired$(\epsilon_2, \Pi, \Gamma, \Delta)$ then all other things being equal, ρ_1 is less interesting than ρ_2, as there is less evidence for the validity and support of ϵ_1 than ϵ_2.

The question arises: what is to be done when violated$(\epsilon_1, \Pi, \Gamma, \Delta) \geq$ violated$(\epsilon_2, \Pi, \Gamma, \Delta)$ but fired$(\epsilon_1, \Pi, \Gamma, \Delta) \geq$ fired$(\epsilon_2, \Pi, \Gamma, \Delta)$? Which expectation is the stronger? There are several choices that can be made concerning the combination of strength values. In certain contexts it may be most appropriate to choose a single one of the three measures as the indicator of expectation strength. The measures may be combined using some function of difference, such as the Euclidean distance, or in some order such as the lexicographic order. The choice of measure will depend on what is most appropriate to the context under consideration and will not be addressed further in this thesis.

3.3 Types of violations of expectations

When a report violates an expectation, this may be an indicator of interesting information. Several violations of the same expectation by a number of different entities, within a short period, may indicate a cohort violation of expectations. Cohort violations of expectations are defined in Section 3.3.1.

There is a subset of expectations that are only applicable to entities known to be in a certain state. If the event model introduced in section 2.4.3 is used then any expectation that contains a `holdsAt` predicate is state dependent. All other expectations are state independent.

It is possible that some violations of expectations are not the result of interesting information. In section 3.3.3, the various reasons for uninteresting violations of expectations are set out and suggestions made regarding identifying interesting from uninteresting violations of expectations.

The above categories are not mutually exclusive, that is, there may be an interesting cohort vi-

olation of a state dependent expectation, an interesting singular violation of a state dependent expectation or an uninteresting cohort violation of a state dependent expectation and so on.

3.3.1 Cohort violations of expectations

Definition 3.1.6 defines a single entity violation of an expectation. For the rest of this thesis, only expectations that are violated by single entities will be considered. However, some violations of expectations only become interesting when a number of entities all exhibit the same unexpected behaviour. This section presents definitions that are of use if these types of violations, which we will call cohort violations of expectations, are to be considered. The enitityCount function (which is defined below) allows us to determine how many entities appear in a set of reports.

Definition 3.3.1. *Let ρ be a report, Z be a set of news atoms, Γ be a set of access rules and Δ be a set of background knowledge. An* **entity classifier**, *denoted* $\mathsf{entity}_z(\rho, \Gamma, \Delta)$, *assigns the value* True *if the news atoms in Z appear in* $\mathsf{representatves}(\rho, \Gamma, \Delta)$ *and* False *otherwise.*

$$\mathsf{entity}_Z(\rho, \Gamma, \Delta) = \mathsf{True} \textit{ iff } Z \subseteq \mathsf{representatives}(\rho, \Gamma, \Delta)$$

Example 3.3.1. *Consider the following representative set:*

$$\mathsf{representatives}(\rho, \Gamma, \Delta) = \{\mathtt{act(bidMade)}, \mathtt{target(Sabena)}, \mathtt{buyer(BritishAirways)}\}$$

It follows that:

$$\begin{aligned}
&\mathsf{entity}_{\{\mathtt{target(Sabena)}\}}(\rho, \Gamma, \Delta) = \mathsf{True}\\
&\mathsf{entity}_{\{\mathtt{buyer(BritishAirways)}\}}(\rho, \Gamma, \Delta) = \mathsf{True}\\
&\mathsf{entity}_{\{\mathtt{target(Sabena)},\mathtt{buyer(BritishAirways)}\}}(\rho, \Gamma, \Delta) = \mathsf{True}\\
&\mathsf{entity}_{\{\mathtt{target(Sabena)},\mathtt{buyer(Ryanair)}\}}(\rho, \Gamma, \Delta) = \mathsf{False}
\end{aligned}$$

Definition 3.3.2. *Let Y be a set of predicate symbols and Z be a set of news atoms. The set* $\mathsf{entitySet}(Y)$ *is a set of entity classifiers such that for each* $\mathsf{entity}_Z \in \mathsf{entitySet}(Y)$, *the following two conditions hold: (1) for each $p(q_1, ..., q_n) \in Z$, $p \in Y$; and (2) for each $p \in Y$, there exists a $p(q_1, ..., q_n) \in Z$.*

Definition 3.3.3. *Let $\{\rho_1, ..., \rho_n\}$ be a set of reports, Γ be a set of access rules and Δ a set of background knowledge and Y be a set of predicate symbols.*

$$\begin{aligned}
\mathsf{entities}(\{\rho_1, ..., \rho_n\}, \Gamma, \Delta, Y) = \{&\mathsf{entity}_Z \in \mathsf{entitySet}(Y) |\\
&\exists \rho_i \in \{\rho_1, ..., \rho_n\} \textit{ s.t. } \mathsf{entity}_Z(\rho_i, \Gamma, \Delta) = \mathsf{True}\}
\end{aligned}$$

and

$$\mathsf{entityCount}(\{\rho_1, ..., \rho_n\}, \Gamma, \Delta, Y) = |\mathsf{entities}(\{\rho_1, ..., \rho_n\}, \Gamma, \Delta, Y)|$$

Example 3.3.2. *For the set of reports* $\{\rho_1, \rho_2, \rho_3\}$*, where*

$$\text{representatives}(\rho_1, \Gamma, \Delta) = \{\texttt{act(bidMade)}, \texttt{target(Sabena)}, \texttt{buyer(BritishAirways)}\}$$
$$\text{representatives}(\rho_2, \Gamma, \Delta) = \{\texttt{act(bidMade)}, \texttt{target(Ryanair)}, \texttt{buyer(BritishAirways)}\}$$
$$\text{representatives}(\rho_3, \Gamma, \Delta) = \{\texttt{act(bidMade)}, \texttt{target(Northwest)}, \texttt{buyer(American)}\}$$

then $\text{entityCount}(\{\rho_1, \rho_2, \rho_3\}, \Gamma, \Delta, \{\texttt{target}\}) = 3$, $\text{entityCount}(\{\rho_1, \rho_2, \rho_3\}, \Gamma, \Delta, \{\texttt{buyer}\}) = 2$, *and* $\text{entityCount}(\{\rho_1, \rho_2, \rho_3\}, \Gamma, \Delta, \{\texttt{target}, \texttt{buyer}\}) = 3$.

In general, for $\text{entityCount}(\{\rho_1, ..., \rho_n\}, \Gamma, \Delta, Y)$, semantic labels are needed to delineate appropriate entities. For example, for companies, $Y = \{\texttt{name}, \texttt{location}\}$ would allow for companies with the same name but different locations to be differentiated. Similarly, for staff records, $Y = \{\texttt{firstname}, \texttt{lastname}, \texttt{birthdate}\}$ may be appropriate.

Let us now examine cohort violations of expectations. As with the definition of a singular violation, it is assumed that the news reports are consistent with the background knowledge. Using this assumption, a cohort violation of expectations is defined as follows:

Definition 3.3.4. *Let* $\{\rho_1, ..., \rho_n\}$ *be a set of reports,* Γ *be a set of access rules,* Δ *be a set of domain knowledge,* E *be a set of expectations, and* $\epsilon \in E$*. A cohort violation occurs when more than one entity is reported as violating that expectation.*

$\{\rho_1, ..., \rho_n\}$ *is a* **cohort violation** *of* ϵ
iff
there exists a non-empty Y *such that* $\text{entityCount}(\{\rho_1, ..., \rho_n\}, \Gamma, \Delta, Y) > 1$
and
$\text{violates}(\text{representatives}(\rho_1, \Gamma, \Delta), \epsilon)$
and...and
$\text{violates}(\text{representatives}(\rho_n, \Gamma, \Delta), \epsilon)$

Example 3.3.3. *Suppose the set of domain facts* Δ *includes the following facts*

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

and that a set of reports, $\{\rho_1, ..., \rho_n\}$ *is received such that the following is a subset of the access predicates* $\text{representatives}(\rho_1, \Gamma, \Delta) \cup ... \cup \text{representatives}(\rho_n, \Gamma, \Delta)$.

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

Suppose also that the expectations E *contains the following*

$$\forall \texttt{c} \in \texttt{companies}\ \neg\texttt{profitable(c)} \rightarrow \neg\texttt{shareMovement(c, up)}$$

This leads us to conclude that the news reports taken with the background facts and the above expectation lead to an inconsistency. Furthermore,

$$\mathsf{entityCount}(\{\rho_1, ..., \rho_n\}, \Gamma, \Delta, \{\texttt{shareMovement}\}) = 3.$$

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, it is possible to assume that the sets Y are selected in advance by hand for each expectation, and therefore form part of the meta-knowledge.

Example 3.3.4. *Assume that the following set of reports all violate the same expectation:*

$$\begin{aligned}
&\mathsf{representatives}(\rho_1, \Gamma, \Delta) = \\
&\qquad \{\texttt{act(bidMade)}, \texttt{target(Sabena)}, \texttt{buyer(BritishAirways)}\} \\
&\mathsf{representatives}(\rho_2, \Gamma, \Delta) = \\
&\qquad \{\texttt{act(bidMade)}, \texttt{target(Ryanair)}, \texttt{buyer(BritishAirways)}\} \\
&\mathsf{representatives}(\rho_3, \Gamma, \Delta) = \\
&\qquad \{\texttt{act(bidMade)}, \texttt{target(Northwest)}, \texttt{buyer(BritishAirways)}\}
\end{aligned}$$

Hence, $\mathsf{entityCount}(\{\rho_1, \rho_2, \rho_3\}, \Gamma, \Delta, \{\texttt{target}, \texttt{buyer}\}) = 3$, *corresponding to three unique tuples made up of the entity features. Yet* $\mathsf{entityCount}(\{\rho_1, \rho_2, \rho_3\}, \Gamma, \Delta, \{\texttt{buyer}\}) = 1$, *and so this set of reports may suggest that the behaviour which violates this expectation is particular to one company acting as a buyer.*

To clarify the difference between singular and cohort violations. For a cohort violation, the condition $\mathsf{entityCount}(\{\rho_1, ..., \rho_n\}, \Gamma, \Delta, Y) > 1$ holds for some non-empty Y. For the singular violations, there is the implicit condition $\mathsf{entityCount}(\{\rho\}, \Gamma, \Delta, Y) = 1$ holding for some non-empty Y that is always satisfiable.

A cohort violation of an expectation must consist of a number of violations falling within a short period. The question of the period of time that is appropriate will not be addressed in this thesis and is a subject for further work.

3.3.2 State dependent and state independent expectations

Some expectations hold for all entities at all times, whereas others hold only for entities in given states. The event model, as defined in Section 2.4.3, is what allows us to identify the state of a given entity and so identify which of these expectations (referred to as state dependent expectations) should hold.

State dependent expectations apply to an entity e at timepoint t if and only if there exists a state s

that holds at time t for entity e.

Definition 3.3.5. *Let e, be an entity, let σ be a state predicate symbol. $s = \sigma(e)$ is a state. A* **state dependent expectation** *is an expectation whose antecedent contains the* $\texttt{holdsAt}(s, t)$ *relation.*

Example 3.3.5. *The expectation "A company which is unprofitable will not be expected to have a rising share price" can be represented by the following state dependent expectation.*

$$\forall \texttt{c} \in \texttt{companies}, \texttt{t} \in \texttt{timePoints}$$
$$\texttt{holdsAt}(\texttt{unprofitable}(\texttt{c}), \texttt{t}) \rightarrow \neg\texttt{shareMovement}(\texttt{c}, \texttt{t}, \texttt{rising})$$

Example 3.3.6. *Consider an event in the event model that allows us to conclude:*

$$\texttt{holdsAt}(\texttt{inAdministration}(\texttt{Railtrack}), 02/02/2002)$$

Suppose the system then receives a report that is represented by the following report atom:

$$\texttt{report}(\texttt{company}(\texttt{Railtrack}), \texttt{annualreport}(\texttt{result}(\texttt{profit}),$$
$$\texttt{amount}(\texttt{GBP292m}), \texttt{date}(02/02/2002))$$

And from this suppose it is possible to derive

$$\texttt{holdsAt}(\texttt{profitable}(\texttt{Railtrack}), 02/02/2002)$$

Now suppose there exists a state dependent expectation in E such that

$$\forall \texttt{c} \in \texttt{Companies}, \texttt{t} \in \texttt{Time}$$
$$\texttt{holdsAt}(\texttt{profitable}(\texttt{c}), \texttt{t}) \rightarrow \neg\texttt{holdsAt}(\texttt{inAdministration}(\texttt{c}), \texttt{t})$$

Hence it is possible to derive $\neg\texttt{holdsAt}(\texttt{inAdministration}(\texttt{Railtrack}), 02/02/2002)$ *and therefore the expectation is violated.*

3.3.3 Uninteresting violations of expectations

It is assumed throughout this thesis that all violations of expectations are the result of the receipt of unexpected information. In this section that assumption is suspended as, in reality, it may be possible for a report to violate an expectation and yet for that report to refer to facts that are uninteresting. For example: if some error in the report or the background facts causes these facts to be misinterpreted or misrepresented within an EVA system. These errors may arise out of misinterpretation errors, that is, the expectation checker failing to recognise thesaurus variants of a term, or from a a misrepresentation error, that is, errors in the presentation of the news report. Misrepresentation errors may include such errors as typographical errors or factual errors in the report.

Throughout this section it will be assumed that the set of background knowledge is exhaustive, that is, the closure of the set of background facts contains all the information that it is necessary to know. It will also be assumed that the set of background knowledge is correct. This assumption simplifies certain definitions and is a reasonable assumption to make. The background facts can be kept largely current and correct by a variety of methods from the fully automatic to the fully manual. Providing the background knowledge is mainly correct we can be confident that the majority of representative sets generated from it are correct.

This section will demonstrate how each of these types of error may arise and suggest strategies for dealing with them. Let us first address misinterpretations. Synonyms, metonyms and formatting variants are used by authors of natural language documents to maintain the reader's attention. Collectively, synonyms, metonyms and reformatting are referred to as thesaurus variants. Metonyms and format variations are special cases of synonymy, but present particular challenges as thesauruses tend not to list metonyms and format abbreviations.

Synonyms are strings of one or more words which have sufficiently similar meaning to another string that substitution of one for the other in a sentence leaves the meaning of the sentence unchanged.

Metonyms are strings of one or more words which *idiomatically* refer to an entity designated by another string (for example, "Number 10" for the office of the British Prime Minister). Substitution of metonyms in a sentence leaves the meaning of the sentence unchanged. Metonyms are highly context dependent: the meaning of the phrase"Number 10" in a news report concerning British politics would be different to the meaning of the same phrase in a report of the music charts for instance.

Format variation are strings that are subject to case or white space differences or that have been abbreviated. If t and t' are strings of one or more words, t' is a reformatted version of t when one of the following applies: (1) t has more white space or non-printing characters than t'; (2) Letters in t are in a different case than those in t'; or (3) t contains abbreviations of terms in t'. Format abbreviations are usually not to be found in thesauri.

Example 3.3.7. *Let the following be a free text report:*
"British Airways lobbied the Prime Minister for a moratorium on plans to increase fuel duty. Number 10 said that BA and other airlines can no longer be exempt from fuel taxes"

In the above text the words "duty" and "tax" are synonyms; "Number 10" is a metonym for "Prime Minister" and "BA" is a format variant of "British Airways".

Synonyms, metonyms and format variants, known collectively as thesaurus variants, create difficulties at two stages in the EVA process: during the generation of the set $\mathtt{match}(\rho, \Gamma, \Delta)$ and in

the grounding of expectations. The following examples will help to illustrate these points. Firstly, the difficulty of generating the set $\mathsf{match}(\rho, \Gamma, \Delta)$:

Example 3.3.8. *Let ρ be a report, Γ be a set of access rules. Let*

$$\mathsf{access}(\rho, \Gamma) = \{\mathtt{lobbies(britishAirways, number10, fuelDuty)}\}$$

and

$$\Delta = \{\mathtt{airline(ba)}, \mathtt{politician(primeMinister)}, \mathtt{levy(fuelTax)}\}$$

By Definition 2.4.4, $\mathsf{match}(\rho, \Gamma, \Delta) = \emptyset$ *as there is no match between the constant symbols in* $\mathsf{access}(\rho, \Gamma)$ *and those in* Δ. *If* $\mathsf{match}(\rho, \Gamma, \Delta)$ *could process thesaurus variants, the following set would be produced:*

$$\mathsf{match}(\rho, \Gamma, \Delta) = \{\mathtt{airline(ba)}, \mathtt{politician(primeMinister)}, \mathtt{levy(fuelTax)}\}$$

Likewise, when grounding expectations it is necessary that thesaurus variants are recognised as otherwise they may lead to false positive or false negative groundings of expectations.

Example 3.3.9. *Let*

$$\begin{aligned}\mathsf{representatives}(\rho, \Gamma, \Delta) = \\ \{&\mathtt{mobilePhoneCo(1\text{-}2\text{-}1)}, \mathtt{mobilePhoneCo(tmobile)}, \\ &\mathtt{takeover(t\text{-}mobile, one\text{-}2\text{-}one)}, \mathtt{profitable(one2one)}\}\end{aligned}$$

$\mathsf{representatives}(\rho, \Gamma, \Delta)$ *should violate the following expectation:*

$$\begin{aligned}\forall \mathtt{c}_1, \mathtt{c}_2\ &\mathtt{mobilePhoneCompany}(\mathtt{c}_1) \wedge \mathtt{mobilePhoneCompany}(\mathtt{c}_2) \wedge \\ &\mathtt{takeover}(\mathtt{c}_1, \mathtt{c}_2) \rightarrow \neg \mathtt{profitable}(\mathtt{c}_2)\end{aligned}$$

However, unless the thesaurus variants are recognised, $\mathsf{representatives}(\rho, \Gamma, \Delta)$ *would no violate that expectation.*

It is therefore evident that some means of identifying thesaurus variants is required. One such method is to define a thesaurus as a set of labels, each associated with a set of words that are synonyms, metonyms or format variations of one-another. Some thesaurus variants may be obtained from existing thesaurus records, others may need to be coded manually. Knowledge bases used in information extraction provide much of this type of information. These labels can be included in the knowledge base Δ and a specialised set of facts.

Definition 3.3.6. *Let $c_1, ..., c_n$ be canonical entry labels that uniquely identify each canonical entry and $t_1, ..., t_n$ each be a string of one or more words. The set of labelled canonical terms,*

to be used in place of one of the thesaurus variants, denoted the **canonical set** *is a follows:* $\mathsf{canonicalSet} = \{c_1, (t_1), ..., c_n(t_n)\}$

Definition 3.3.7. *Let* $\mathsf{canonicalSet} = \{c_1, (t_1), ..., c_n(t_n)\}$ *be a canonical set,* $e_1, ..., e_n$ *be entry labels that uniquely identify each set of thesaurus variants and* $t_1, ..., t_m$ *each be a string of one or more words. Let* $\{t_j, ..., t_k\}$ *and* $\{t_l, ..., t_m\}$ *be sets of terms such that each member of a set is a thesaurus variant of the other members of that set. The set of labelled sets of thesaurus variants is as follows:*

$$\mathsf{thesaurusVariants} = \{e_1(t_j, ..., t_k), ..., e_n(t_l, ..., t_m)\}$$

For each e_n *in the set of thesaurus labels* $\{e_1, ..., e_n\}$ *there is a* c_n *in the set of canonical labels* $c_1, ..., c_n$.

Example 3.3.10. *Let* $\mathsf{canonicalSet} = \{\mathtt{car(car)}\}$ *and let*

$$\mathsf{thesaurusVariants} = \{\mathtt{car(automobile, motor, wagon}\}$$

Definition 3.3.8. *Let* $e_n(t_1, ..., t_m) \in \mathsf{thesaurusVariants}$, t_j, t_k *be strings of one or more words and* t_j, t_k *be subterms of* $e_n(t_l, ..., t_m) \in$ *ThesaurusVariants iff one of the following applies:*

1. t_j *is a synonym for* t_k
2. t_j *is a metonym for* t_k
3. t_j *is a reformatted version of* t_k

For all $t_j \in \mathsf{subterms}(e_1(t_i, ..., t_m)), ..., \mathsf{subterms}(e_n(t_n, ..., t_o))$, t_j *can be substituted for the canonical version* $t_j^* \in \mathsf{subterms}(c_1(t_1)) \cup ... \cup \mathsf{subterms}(c_n(t_n))$ *if the label* e_m *for the set that includes term* t_m *is equal to the label of the canonical set* c_m *that contains* t_j^*.

$$\begin{aligned}\mathsf{substitution}(t_j) = \{t_j^* \mid\ & t_j^* \in \mathsf{subterms}(c_n(t_j^*)) \text{ and} \\ & t_j \in \mathsf{subterms}(e_n(t_1, ..., t_n)) \text{ and } e_n = c_n\}\end{aligned}$$

Abbreviations may be derivable within articles. For example within an article which mentions British Petroleum and then goes on to use the abbreviation BP, it is probably safe to assume that BP is an abbreviation for British Petroleum. However, the inverse process, deriving full names from abbreviations, is not so straightforward. Treating abbreviations as thesaurus variants may work in limited domains. For example, in the domain of company reports BP is probably an abbreviation for British Petroleum, however, in medical journals, it probably refers to blood pressure. If the domain is known, it may be possible to treat common abbreviations as a thesaurus variant of the full term.

The function substitution(t_m) can be applied to all reports prior to the application of any other function, thus ensuring that the terms in the resulting representative set representatives(ρ, Γ, Δ) are in canonical form. It is assumed that the facts derived from Δ also contain only canonical terms. By this method, all misinterpretation errors can be avoided.

I now address misrepresentation errors, which fall into one of three categories: typographical errors; text structuring errors and factual errors in reports.

As typographical errors are not predictable, they cannot be spotted using the substitution(t_m) function defined above. Therefore there is no set that can be created that can be used to capture and correct these errors. However, this is an error that may be corrected at the information extraction stage. If the information extraction engine is equipped with an effective spell checker it can use its knowledge of parts of speech and likely patterns of words to select the most likely correct rendering of the misspelt word. Whilst this is not a guaranteed solution, the number of expected typographical errors is low and therefore should not unduly affect the performance of an EVA system.

Incorrect classification of extracted information arises where a text entry in the structured text is contained between the wrong pair of tags. This creates a difficulty when matching representative sets with expectations as in the following example:

Example 3.3.11. *Let ρ be a report and Γ be a set of access rules. Let*

$$\mathsf{access}(\rho, \Gamma) = \{\mathtt{takeover(ryanair, ireland)}\}$$

and

$$\mathsf{match}(\rho, \Gamma, \Delta) = \{\mathtt{basedIn(ryanair, ireland)}, \mathtt{country(ireland)}, \mathtt{airline(ryanair)}\}$$

The set representatives(ρ, Γ, Δ) *appears to tell us that the airline Ryanair has launched a takeover bid for the country, Ireland. This is because the report ρ contains a fact where a country has been misclassified as a company in the set* access(ρ, Γ)*, as the predicate* `takeover` *should only ever refer to two companies. This would lead to a false positive violation of the expectation:*

$$\forall \mathtt{c_1}, \mathtt{c_2}\ \mathtt{airline(c_1)} \wedge \mathtt{country(c_2)} \rightarrow \neg\mathtt{takeover(c_1, c_2)}$$

Such errors could be considered "noise". The degree of noise will largely be determined by the efficacy of the information extraction process that generates news atoms. No work has yet been done to determine the frequency or effect of such noise in the representative set. This remains a question for further study.

It is difficult to identify a characteristic pattern that arises in the case of text misclassification

that would allow us to identify misclassified text and to deal with it. A gazetteer is a list of words of a single type, used by information extraction engines. An example of the "Animal" gazetteer for instance may be "*Aardvark, African elephant, albino monkey, ant, antelope...zebra*". Misclassification errors usually result from an ambiguity over the part of speech or the gazetteer to which a term belongs. As the type of a term classifies either its meaning, part of speech or both, any attempt to use typing to solve this problem will undoubtedly fall foul of the same ambiguities.

However, misclassification is fundamentally an information extraction issue. Therefore the problem of misclassification is beyond the scope of this thesis. It will be assumed from hereon that all information received in structured news reports is correctly classified, or sufficiently well classified to provide a representation of the real world that is approximately accurate. To do so is not unreasonable. Many statistical and machine learning techniques are satisfied with providing an indicative measure of accuracy rather than "book accuracy".

Factual errors in reports also cause problems with the matching of representative sets with expectations, as in Example 3.3.11. They are also difficult to identify unless the error is such that the facts in the representative set are somehow rendered inconsistent, as in Definition 3.3.9.

Definition 3.3.9. *Let ρ be a report, Γ be a set of access rules and Δ be a set of background knowledge, which is assumed to always be correct. ρ contains an erroneous fact if*

$$\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \bot$$

However, facts may be in error and yet not lead to an inconsistency. If factual errors are not inconsistent with background knowledge, they are impossible to detect within the EVA framework. However, we assume that reports are factually accurate and the problem of erroneous information does not receive further discussion in this thesis.

In the situation that a report is known to lead to a misrepresentation error it can be disregarded as a potential source of useful information, or it can be dealt with by some other means. However, if a report is not identified as being misrepresentative when it indeed is, that report may generate a false negative, not violating an expectation that should be violated, or a false positive, violating a report that should not be violated. If greater accuracy is required, one way to improve performance is to require more than one source for any claim of extraordinary interest. There is much work that addresses the difficulty of merging information in reports. A review can be found in [HS04]. This issue will not be addressed further in this thesis.

For the sake of simplicity it will be assumed, unless explicitly stated, that all violations are single entity violations of state independent expectations. The principles discussed from here onwards are applicable to all types of expectation violation unless stated otherwise. It will also be assumed that all misinterpretation and misrepresentation errors have been eliminated. As such all inconsistencies between representative sets and expectations are indicative of unexpected, and therefore interesting, information.

3.4 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 it is possible to 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. Rather than checking the consistency, directly proving whether or not each expectation has been violated is a directed activity and therefore does not rely on the use of search methods. This is achieved by rewriting each expectation into another formula that is called a viaduct as follows:

Definition 3.4.1. *Let* $\texttt{label}(\epsilon)$ *be the label of an expectation of the form* $\forall x_1, ..., x_k\ \alpha_1 \wedge .. \wedge \alpha_n \rightarrow \beta$. *A rule that can be used to check for the violation of an expectation, denoted a* **viaduct**, *is a formula of the following form:*

$$\forall x_1, ..., x_k\ \alpha_1 \wedge .. \wedge \alpha_n \wedge \neg\beta \rightarrow \texttt{violation}(\texttt{label}(\epsilon))$$

If E *is a set of expectations,* $\mathsf{viaduct}(E)$ *is the set of viaducts obtained from* E.

Example 3.4.1. *Let expectation* $\epsilon_1 =$

$$\forall \texttt{x}\ \texttt{profitable(x)} \wedge \texttt{large(x)} \rightarrow \texttt{listed(x)}$$

The viaduct for $\epsilon_1 =$

$$\forall \texttt{x}\ \texttt{profitable(x)} \wedge \texttt{large(x)} \wedge \neg\texttt{listed(x)} \rightarrow \texttt{violation}(\texttt{label}(\epsilon_1))$$

Reasoning with $\mathsf{viaduct}(E)$ and reasoning with E is identical in the sense that the set of expectations that are identified as being violated is identical. This is demonstrated below.

Proposition 3.4.1. *Let* ρ *be a structured news report,* Γ *be a set of access rules,* Δ *be a set of domain knowledge,* E *be a set of expectations and* $\epsilon \in E$. *Let* $\epsilon = \forall x_1, ..., x_k\ \alpha_1 \wedge ... \wedge \alpha_n \rightarrow \beta$.

ρ *is a* **violation** *of* $\epsilon \in E$ *iff the following two conditions hold:*

(1) $\mathsf{viaduct}(E)$ *includes the following viaduct*

$$\forall x_1, ..., x_k\ \alpha_1 \wedge ... \wedge \alpha_n \wedge \neg\beta \rightarrow \texttt{violation}(\texttt{label}(\epsilon))$$

(2) there exists a grounding set Φ *such that*

$$\mathsf{access}(\rho, \Gamma) \cup \mathsf{match}(\rho, \Delta) \vdash \mathsf{ground}(\alpha_1 \wedge ... \wedge \alpha_n \wedge \neg\beta, \Phi)$$

Proof. The following conditions are equivalent: ρ is a violation of $\epsilon \in E$

(i) iff ρ violates ϵ and $\epsilon \in E$
(ii) iff representatives$(\rho, \Gamma, \Delta) \cup \{\epsilon\} \vdash \bot$ and $\epsilon \in E$
(iii) iff $\exists$ a grounding set Φ s.t. representatives$(\rho, \Gamma, \Delta) \vdash$ ground$(\neg$consequent$(\epsilon), \Phi)$ and
representatives$(\rho, \Gamma, \Delta) \vdash$ ground(antecedent$(\epsilon), \Phi)$
and $\epsilon \in E$
(iv) iff $\exists \Phi$ s.t. representatives$(\rho, \Gamma, \Delta) \vdash$ ground(ϵ, Φ)
and antecedent$(\epsilon) \wedge \neg$consequent$(\epsilon) \rightarrow$ violation(label$(\epsilon)) \in$ viaduct(E).

□

The use of viaducts is a method of compiling consistency checking. Proposition 3.4.1 shows that consistency checking is maintained as in the original definition for a violation of an expectation.

The following example illustrates that compilation loses the ability to identify inconsistencies arising from representatives(ρ, Γ, Δ). In other words, there may be a number of possible subsets of representatives$(\rho, \Gamma, \Delta) \cup E$ that are inconsistent but that these cannot be detected if viaduct(E) is used, rather than E. Given that we assume that there are no inconsistencies in representatives(ρ, Γ, Δ), this is not a problem within the EVA framework.

Example 3.4.2. *Let* a(d) *and* b(d) *be in* access(ρ, Γ) *for some report* ρ. *Also let two of the expectations in the set of expectations,* E *be*

$$\forall x\ a(x) \rightarrow c(x)$$
$$\forall x\ b(x) \rightarrow \neg c(x)$$

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. Viaducts turn checking whether a given news report ρ violates an expectation into a decidable problem of classical propositional logic. 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 the problem is considered as an abduction problem, where the existence of a minimal subset of a set of formulae that implies a contradiction is sought, 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.

Using viaducts reduces checking whether a given news report ρ violates an expectation to a decidable problem, which makes implementation possible. A Prolog implementation would incorporate a meta-interpreter (adapted from [SS94]) that evaluates each viaduct in turn.

3.5 Discussion of Chapter 3

This chapter presents the methods used by the EVA framework to both identify and to measure the unexpectedness of information. Information from news reports and background knowledge is considered interesting, with respect to the corpus of expectations, if and only if that information violates one or more of those expectations. The notion of a violation has a clear meaning, resulting, as it does, from the concurrent firing and attacking of an expectation. Searching for inconsistency between representative sets and expectations is a directed activity by means of the viaducts introduced in Section 3.4.

However, there are outcomes other than violation arising from the relation between news, background facts and expectations. Whilst these outcomes may not identify interesting information, they are of use in assessing the strength of expectations. If a representative set does not fire an expectation then that expectation is not relevant to the facts in that representative set and need not be considered any further. If an expectation is both fired and supported by a given representative set then that representative set confirms the expectation. Confirmation increases the validity and accuracy of the expectation by providing evidence that the expectation correctly describes the usual state of the world. In the absence of the closed world assumption a representative set may fire an expectation yet neither attack nor support it. This strengthens the accuracy of the expectation, providing evidence that this expectation does not contradict situations in the real world. Violation of an expectation weakens the evidence that an expectation represents the real world and so decreases accuracy and validity.

These relationships are therefore of interest not only when identifying interesting news reports but also when measuring the reliability of expectations. Central to the EVA framework is the need to evaluate the violations of expectations. This evaluation is a form of measuring of inconsistency. The more valid or more accurate an expectation is, the more significant the inconsistency arising when the expectation is violated. The violation of a highly valid expectation indicates that the reported behaviour is in direct contradiction to that expected given the situation described in the antecedent of that expectation. The violation of a highly accurate expectation merely indicates that the reported behaviour is unusual given the situation described in the antecedent of the expectation.

The measures proposed in this chapter have several advantages: they have clear semantics based on the occurrence of ground instances of the antecedent and consequent in reports and background knowledge. These measures are also based on real world events and thus have a meaningful derivation. The measures can be calculated in constant time, making them ideal for implementation.

The values for expectations should 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, there may be an expectation in the domain of mergers and acquisitions, where it is expected that a company will sell off subsidiaries, indicating a market going through a period of decentralisation. There may also be an opposing expectation that companies will buy other companies, indicating a period of consolidation. Given that every time an expectation is violated its confirmation decreases by some degree and the confirmation of the opposing expectation increases by the same degree, the strength of these expectations acts as 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 can be assigned a weight which decreases with time.

Secondly, applying confirmation theory to information regarding a cohort of entities is more difficult than applying it to singular violations. Recall that, for a cohort violation to take place, there needs to be more than one representative set involved in a relation with a given expectation during a given time. These relationships would be recognised by a human reader as forming a pattern. For cohort violations, it is necessary to consider how to delineate sets of representative sets in order to achieve the following goals:

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

As an interim measure, rather than propose an aggregation of these goals, 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. In the longer term further work may provide clearer guidance on how to apply cohort information to confirmation theoretic measures.

Chapter 4

The set of expectations

The EVA framework, presented in outline in Chapter 2, identifies interesting information by identifying (consistent) representative sets that, together with an expectation about the behaviour of entities in the world, leads to an inconsistency. The set of expectations is therefore central to the EVA framework. This chapter presents a set of definitions that formalise a set of expectations for a given language. This chapter demonstrates that there is an order over expectations that determines the relative strengths of those expectations.

This chapter is structured as follows: Section 4.1 defines the set of expectations and an ordering over those expectations. Section 4.2 demonstrates that there is a relationship between the order of expectations in this set and the order of their coverage, support and attacked values. Finally, Section 4.3 demonstrates that this relationship facilitates the search for desirable expectations.

Results for validity values can be proved dually from the results for accuracy values and results for ssets can be proved dually from the results for vsets. Therefore, proofs for validity will not be given.

The contributions in this chapter are that: antecedents and consequents can be partially ordered by logical entailment (e.g. $\forall \bar{x}\ \alpha \wedge \beta \vdash \forall \bar{x}\ \alpha$) and that antecedents and consequents can be ordered in to such a way that the precedence relation between two formulae determines the order of the values for those formulae.

4.1 Marker formulae and expectations

A set of marker formulae is a set of unground, universally quantified, unconditional formulae for a given language that excludes function symbols. A set of marker formulae is representative of all

possible antecedents and consequents for a given language. That is, for each possible antecedent or consequent there is a marker formula that is its logical equivalent in the set of marker formulae. For any language with a finite number of predicate symbols, no function symbols, an upper limit on the arity of predicates and the length of formulae, there is a finite number of antecedents and consequents.

This section defines sets of marker formulae and the expectations that can be generated from them.

Definition 4.1.1. *Let P be set of predicate symbols, V be a set of variable symbols and n the maximum arity of literals formed from P and V. The set of marker formulae formed from P and V, denoted M is as follows:*

$$M = \{\forall \bar{v}\ \alpha | p \in P,\ v_1, ...v_j \in V\ , 1 \leq j \leq n,\ \bar{v} = \langle v_1, ..., v_j \rangle \text{ and } \alpha = p(v_1, ..., v_j) \text{ or}$$
$$\forall \bar{v_1}\ \beta_1, \forall \bar{v_2}\ \beta_2 \in M \text{ and}$$
$$\forall \bar{v}\ \alpha = \forall \bar{v_1}\ \neg\beta_1 \text{ or } \forall \bar{v}\ \alpha = \forall \bar{v_2}\ \neg\beta_2 \text{ or}$$
$$\forall \bar{v}\ \alpha = \forall \bar{v_1}, \bar{v_2}\ \beta_1 \wedge \beta_2 \text{ or}$$
$$\forall \bar{v}\ \alpha = \forall \bar{v_1}, \bar{v_2}\ \beta_1 \vee \beta_2$$

The above definition results in an infinitely large set of formulae, as there is no limit on the length of conjunctions or disjunctions. However, we will assume some limit is placed on the length of disjunctions and conjunctions in the set M, thus restricting M to a finite set.

Example 4.1.1. *Let $P = \{p_1, p_2\}$. Let $V = \{v_i, v_2\}$. Let $n = 1$. $M =$*

$\{\forall v_1\ p_1(v_1)$,	$\forall v_2\ p_1(v_2)$,
$\forall v_1\ p_2(v_1)$,	$\forall v_2\ p_2(v_2)$,
$\forall v_1\ \neg p_1(v_1)$	...
$\forall v_1\ p_1(v_1) \wedge p_2(v_1)$,	...
$\forall v_1, v_2\ p_2(v_2) \vee p_1(v_1)$,	$\forall v_2\ p_2(v_2) \vee p_1(v_2)\}$

There is an ordering over a set of marker formulae based on the consequence relation ($\vdash$). For any two formulae $\alpha_1, \alpha_2 \in M$, if $\{\alpha_1\} \vdash \alpha_2$ then we say that α_1 is equally or less general than α_2.

Definition 4.1.2. *Let $\preceq$ be the consequence order relation over A such that:*

$$\forall\ \alpha_1\alpha_2 \in M, \alpha_1 \preceq \alpha_2 \textit{ iff } \{\alpha_1\} \vdash \alpha_2$$

We extend the notation as follows:

$$1: \textit{For all } \alpha_1\ \alpha_2 \textit{ if } \alpha_1 \preceq \alpha_2 \textit{ and } \alpha_2 \npreceq \alpha_1 \textit{ then } \alpha_1 \prec \alpha_2$$
$$2: \textit{For all } \alpha_1\ \alpha_2 \textit{ if } \alpha_1 \preceq \alpha_2 \textit{ and } \alpha_2 \preceq \alpha_1 \textit{ then } \alpha_1 = \alpha_2$$
$$3: \textit{For all } \alpha_1\ \alpha_2 \textit{ if } \alpha_1 \npreceq \alpha_2 \textit{ and } \alpha_2 \npreceq \alpha_1 \textit{ then } \alpha_1 || \alpha_2$$

Example 4.1.2. *Let* $\{\forall \mathtt{x}\ \mathtt{profitable(x)}, \forall \mathtt{x}\ \mathtt{profitable(x)} \wedge \mathtt{bank(x)}\} \subseteq M$.

It is the case that $\{\forall \mathtt{x}\ \mathtt{profitable(x)} \wedge \mathtt{bank(x)}\} \vdash \forall \mathtt{x}\ \mathtt{profitable(x)}$*, therefore*

$$\forall \mathtt{x}\ \mathtt{profitable(x)} \wedge \mathtt{bank(x)} \preceq \forall \mathtt{x}\ \mathtt{profitable(x)}$$

It is also the case that $\{\forall \mathtt{x}\ \mathtt{profitable(x)}\} \nvdash \forall \mathtt{x}\ \mathtt{profitable(x)} \wedge \mathtt{bank(x)}$*, therefore*

$$\forall \mathtt{x}\ \mathtt{profitable(x)} \not\preceq \forall \mathtt{x}\ \mathtt{profitable(x)} \wedge \mathtt{bank(x)}$$

Thus $\forall \mathtt{x}\ \mathtt{profitable(x)} \wedge \mathtt{bank(x)} \prec \forall \mathtt{x}\ \mathtt{profitable(x)}$.

Proposition 4.1.1. *Trivially, the pair* $\langle M, \preceq \rangle$ *is a partially ordered set.*

Example 4.1.3. *Let* $\bar{x}$ *be a tuple of variables* $x_1, ..., x_k$ *where k is some positive integer. Let A contain the formulae* $\forall \bar{x}\ \alpha \wedge \beta$*,* $\forall \bar{x}\ \alpha$*,* $\forall \bar{x}\ \beta$*,* $\forall \bar{x}\ \alpha \vee \beta$*. The consequence relation* $\vdash$ *holds between the formulae is as follows:*

$$\{\forall \bar{x}\ \alpha \wedge \beta\} \vdash \forall \bar{x}\ \alpha,$$
$$\{\forall \bar{x}\ \alpha \wedge \beta\} \vdash \forall \bar{x}\ \beta,$$
$$\{\forall \bar{x}\ \alpha\} \vdash \forall \bar{x}\ \alpha \vee \beta$$
$$\{\forall \bar{x}\ \beta\} \vdash \forall \bar{x}\ \alpha \vee \beta$$

So for the pair $\langle M, \preceq \rangle$ *the following orders hold:*

$$\forall \bar{x}\ \alpha \wedge \beta \prec \forall \bar{x}\ \alpha,$$
$$\forall \bar{x}\ \alpha \wedge \beta \prec \forall \bar{x}\ \beta,$$
$$\forall \bar{x}\ \alpha \wedge \beta \prec \forall \bar{x}\ \alpha \vee \beta,$$
$$\forall \bar{x}\ \alpha \prec \forall \bar{x}\ \alpha \vee \beta,$$
$$\forall \bar{x}\ \beta \prec \forall \bar{x}\ \alpha \vee \beta$$

For any language it is possible to generate a finite set of expectations based on the formulae in the set A, where each expectation may not exceed a certain length, or if the inclusion of logically equivalent expectations is not permitted.

Definition 4.1.3. *Let* $\bar{x}, \bar{y}, \bar{z}$ *be tuples of variables. Let A the set of all marker formulae. Let* $\oplus$ *be a function that combines tuples of variables such that* $\langle v_1, ..., v_j \rangle \oplus \langle v_k, ..., v_n \rangle = \langle v_1, ..., v_n \rangle$*. For example,* $\langle w, x \rangle \oplus \langle y, z \rangle = \langle w, x, y, z \rangle$*. Let*

$$E = \{\forall \bar{x}\ \alpha \rightarrow \beta | \forall \bar{y}\ \alpha \in M \text{ and } \forall \bar{z}\ \beta \in M \text{ and } \bar{x} = \bar{y} \oplus \bar{z}\}$$

E is a set of expectations.

The set of expectations is therefore the set that contains every pair of antecedents and consequents, joined by the conditional relation ($\rightarrow$) and universally quantified outermost.

In the following definitions, antecedents and consequents of expectations are drawn from the set of unconditional formulae, M, and expectations are members of the set E. The pair $\langle \preceq, M \rangle$ is the partial ordering over the set M by the consequence relation, $\vdash$.

4.2 Values and the set of expectations

The set of all expectations for a given language, assuming no function symbols, predicates of finite arity and no repeated literals, is finite. But for even relatively restricted languages it is very large. It is useful to be able to predict which expectations are the most useful. Being able to derive the order of the strength values of expectations from the consequence order of those expectations enables such predictions to be made.

For any expectation, ϵ, set of reports, Π, set of access rules, Γ and set of background knowledge, Δ, the value $\mathsf{fired}(\epsilon, \Pi, \Gamma, \Delta)$ is a function of the set of representative set and the antecedent of ϵ. The value $\mathsf{attacked}(\epsilon, \Pi, \Gamma, \Delta)$ is a function of the consequent of ϵ and the set of representative sets. The values $\mathsf{violated}(\epsilon, \Pi, \Gamma, \Delta)$ and $\mathsf{confirmed}(\epsilon, \Pi, \Gamma, \Delta)$ are functions of the antecedent and consequent of ϵ and the set of representative sets. For definitions of the functions $\mathsf{fired}(\epsilon, \Pi, \Gamma, \Delta)$, $\mathsf{attacked}(\epsilon, \Pi, \Gamma, \Delta)$, $\mathsf{violated}(\epsilon, \Pi, \Gamma, \Delta)$ and $\mathsf{confirmed}(\epsilon, \Pi, \Gamma, \Delta)$, refer to Section 3.2.

This section demonstrates that the order of the antecedents and consequents of any pair of expectations determines the fired, confirmed and attacked values of those expectations. Section 4.2.1 on antecedent order and expectation values, proves the link between the order of the antecedents of two expectations ϵ_x and ϵ_y and the order of the values $\mathsf{fired}(\epsilon_x, \Pi, \Gamma, \Delta)$ and $\mathsf{fired}(\epsilon_y, \Pi, \Gamma, \Delta)$. Section 4.2.2 on consequent order and expectation values, demonstrates that the order of the consequents of expectations ϵ_x and ϵ_y is related to the order of the values $\mathsf{attacked}(\epsilon_x, \Pi, \Gamma, \Delta)$ and $\mathsf{attacked}(\epsilon_y, \Pi, \Gamma, \Delta)$ and the order of $\mathsf{supported}(\epsilon_x, \Pi, \Gamma, \Delta)$ and $\mathsf{supported}(\epsilon_y, \Pi, \Gamma, \Delta)$. Section 4.2.3 on consequence order and expectation values, shows that the relative ordering of two expectations ϵ_x and ϵ_y is related to the order of the values $\mathsf{violated}(\epsilon_x, \Pi, \Gamma, \Delta)$ and $\mathsf{violated}(\epsilon_y, \Pi, \Gamma, \Delta)$ and to the order of the values $\mathsf{confirmed}(\epsilon_x, \Pi, \Gamma, \Delta)$ and $\mathsf{confirmed}(\epsilon_y, \Pi, \Gamma, \Delta)$.

4.2.1 Antecedent order and expectation values

The consequence order of two antecedents determines their relative fired values. The fired value of an expectation is a function from the antecedent of that expectation and the set of representative sets to the range of positive integers.

Proposition 4.2.1. *Let E be a set of expectations. For all $\epsilon_j, \epsilon_k \in E$ such that $\{\mathsf{antecedent}(\epsilon_j)\} \vdash \mathsf{antecedent}(\epsilon_k)$ and $\{\mathsf{antecedent}(\epsilon_k)\} \vdash \mathsf{antecedent}(\epsilon_j)$, the set of reports*

that fires ϵ_j, $\mathsf{fset}(\epsilon_j, \Pi, \Gamma, \Delta)$, *is identical to* $\mathsf{fset}(\epsilon_k, \Pi, \Gamma, \Delta)$ *for a given set of reports* Π, *access rules* Γ *and background facts* Δ. *As a result,* $\mathsf{fired}(\epsilon_j, \Pi, \Gamma, \Delta) = \mathsf{fired}(\epsilon_k, \Pi, \Gamma, \Delta)$.

Proof. Let ρ be a report and Δ be a set of background facts, and Γ, a set of access rules. For all $\mathsf{antecedent}(\epsilon_1)$, $\mathsf{antecedent}(\epsilon_2)$, if $\{\mathsf{antecedent}(\epsilon_1)\} \vdash \mathsf{antecedent}(\epsilon_2)$ then $\mathsf{antecedent}(\epsilon_1) \preceq \mathsf{antecedent}(\epsilon_2)$. It follows that for all ρ if there is a grounding set Φ_j such that $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\mathsf{ground}(\epsilon_j, \Phi_j))$ then there is a grounding set Φ_k such that $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\mathsf{ground}(\epsilon_k, \Phi_k))$. Likewise, for all ρ if there is a grounding set Φ_k such that $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\mathsf{ground}(\epsilon_k, \Phi_k))$ then there is a grounding set Φ_j such that $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\mathsf{ground}(\epsilon_j, \Phi_j))$. Therefore

$$\mathsf{fset}(\epsilon_j, \Pi, \Gamma, \Delta) = \mathsf{fset}(\epsilon_k, \Pi, \Gamma, \Delta))$$

By Definition 3.2.1 for all ϵ_n, $\mathsf{fired}(\epsilon_n, \Pi, \Gamma, \Delta) = |\mathsf{fset}(\epsilon_n, \Pi, \Gamma, \Delta)|$, therefore $\mathsf{fired}(\epsilon_j, \Pi, \Gamma, \Delta) = \mathsf{fired}(\epsilon_k, \Pi, \Gamma, \Delta)$. □

For any ordered pair of antecedents, $\mathsf{antecedent}(\epsilon_1) \preceq \mathsf{antecedent}(\epsilon_2)$, it is the case that $\mathsf{antecedent}(\epsilon_2)$ will be fired at least as often as $\mathsf{antecedent}(\epsilon_1)$.

Proposition 4.2.2. *For all* ϵ_j, ϵ_k *such that* $\mathsf{antecedent}(\epsilon_j) \preceq \mathsf{antecedent}(\epsilon_k)$, *it follows that* $\mathsf{fset}(\epsilon_j, \Pi, \Gamma, \Delta) \subseteq \mathsf{fset}(\epsilon_k, \Pi, \Gamma, \Delta)$ *and therefore*

$$\mathsf{fired}(\epsilon_j, \Pi, \Gamma, \Delta) \leq \mathsf{fired}(\epsilon_k, \Pi, \Gamma, \Delta)$$

Proof. Let ρ be a report and Δ be a set of background facts, and Γ, a set of access rules. Given that for all $\mathsf{antecedent}(\epsilon_j)$, $\mathsf{antecedent}(\epsilon_k)$, if $\mathsf{antecedent}(\epsilon_j) \preceq \mathsf{antecedent}(\epsilon_k)$ then $\{\mathsf{antecedent}(\epsilon_j)\} \vdash \mathsf{antecedent}(\epsilon_k)$, it follows that for all ρ if there is a grounding set Φ_j such that $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\mathsf{ground}(\epsilon_j, \Phi_j))$ then there is a grounding set Φ_k such that $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\mathsf{ground}(\epsilon_k, \Phi_k))$. Therefore

$$\mathsf{fset}(\epsilon_j, \Pi, \Gamma, \Delta) \subseteq \mathsf{fset}(\epsilon_k, \Pi, \Gamma, \Delta))$$

By Definition 3.2.1, if $\mathsf{fset}(\epsilon_j, \Pi, \Gamma, \Delta) \subseteq \mathsf{fset}(\epsilon_j, \Pi, \Gamma, \Delta)$ then $\mathsf{fired}(\epsilon_j, \Pi, \Gamma, \Delta) \leq \mathsf{fired}(\epsilon_k, \Pi, \Gamma, \Delta)$. □

Example 4.2.1. *Let* Γ *be a set of access rules and* Δ *be a set of background facts. Let* $\epsilon_1 = \mathtt{airline(x)} \wedge \mathtt{ftse100(x)} \rightarrow \mathtt{profitable(x)}$ *and* $\epsilon_2 = \mathtt{airline(x)} \rightarrow \mathtt{profitable(x)}$. *It follows that* $\mathsf{antecedent}(\epsilon_1) \preceq \mathsf{antecedent}(\epsilon_2)$.

Let $\mathsf{ground}(\epsilon_1, \Phi_1) = \mathtt{airline(ba)} \wedge \mathtt{ftse100(ba)} \rightarrow \mathtt{profitable(ba)}$ *and* $\mathsf{ground}(\epsilon_2, \Phi_2) = \mathtt{airline(ba)} \rightarrow \mathtt{profitable(ba)}$.

Let ρ_1 *be a report such that* representatives$(\rho_1, \Gamma, \Delta) \vdash$ `airline(ba)` $\wedge$ `ftse100(ba)`. *Consequently* ρ_1 *fires* ϵ_1 *and* ϵ_2.

Let ρ_2 *be a report such that* representatives$(\rho_2, \Gamma, \Delta) \vdash$ `airline(ba)`. *Consequently* ρ_2 *fires* ϵ_2 *but not* ϵ_1.

All reports which fire ϵ_1 *are in both* fset(`airline(x)`$\wedge$`ftse100(x)` $\rightarrow$ `profitable(x)`$, \Pi, \Gamma, \Delta)$ *and* fset(`airline(x)` $\rightarrow$ `profitable(x)`$, \Pi, \Gamma, \Delta)$, *but some reports which fire* ϵ_2 *may be in* fset(`airline(x)` $\rightarrow$ `profitable(x)`$, \Pi, \Gamma, \Delta)$ *but not* fset(`airline(x)` $\wedge$ `ftse100(x)` $\rightarrow$ $profitable(x), \Pi, \Gamma, \Delta)$. *Thus* fset$(\epsilon_1, \Pi, \Gamma, \Delta) \subseteq$ fset$(\epsilon_2, \Pi, \Gamma, \Delta)$ *and, by Definition 3.2.1,* fired$(\epsilon_1, \Pi) \leq$ fired(ϵ_2, Π).

4.2.2 Consequent order and expectation values

The attacked value of an expectation is a function from the consequent of that expectation and the set of representative sets to a positive integer. The negation of two logically equivalent consequents will be ground by the same reports. This knowledge is used to direct the generation of a working set of expectations (more details are given in Chapter 5).

Proposition 4.2.3. *Let* E *be a set of expectations. For all* $\epsilon_j, \epsilon_k \in E$ *such that* consequent$(\epsilon_j) \preceq$ consequent(ϵ_k) *and* consequent$(\epsilon_k) \preceq$ consequent(ϵ_j), aset(ϵ_j) *is identical to* aset(ϵ_k). *As a result,* attacked$(\epsilon_j) =$ attacked(ϵ_k).

Proof. Let ρ be a report and Δ be a set of background facts, and Γ, a set of access rules. Given that for all consequent(ϵ_1), consequent(ϵ_2), if consequent$(\epsilon_1) \preceq$ consequent(ϵ_2) then $\{$consequent$(\epsilon_1)\} \vdash$ consequent(ϵ_2), it follows that for all ρ if there is a grounding set Φ_k such that representatives$(\rho, \Gamma, \Delta) \vdash \neg$consequent(ground$(\epsilon_k, \Phi_k)$ then there is a grounding set Φ_j such that representatives$(\rho, \Gamma, \Delta) \vdash \neg$consequent(ground$(\epsilon_j, \Phi_j)$. Likewise, for all ρ if there is a grounding set Φ_j such that representatives$(\rho, \Gamma, \Delta) \vdash \neg$consequent(ground$(\epsilon_j, \Phi_j)$ then there is a grounding set Φ_k such that representatives$(\rho, \Gamma, \Delta) \vdash \neg$consequent(ground$(\epsilon_k, \Phi_k)$. Therefore

$$\text{aset}(\epsilon_j, \Pi, \Gamma, \Delta) = \text{aset}(\epsilon_k, \Pi, \Gamma, \Delta))$$

By Definition 3.2.2 for all ϵ_n, attacked$(\epsilon_n, \Pi, \Gamma, \Delta) = |\text{aset}(\epsilon_n, \Pi, \Gamma, \Delta)|$. Therefore attacked$(\epsilon_j, \Pi, \Gamma, \Delta) =$ attacked$(\epsilon_k, \Pi, \Gamma, \Delta)$. □

For any ordered pair of consequents it is the case that the consequent that is lower in the order will be attacked at least as often as the consequent that is higher in the order.

Proposition 4.2.4. *Let E be a set of expectations. For all $\epsilon_j, \epsilon_k \in E$ such that* $\mathsf{consequent}(\epsilon_j) \preceq \mathsf{consequent}(\epsilon_k)$ $\mathsf{aset}(\epsilon_j) \subseteq \mathsf{aset}(\epsilon_k)$. *As a result,* $\mathsf{attacked}(\epsilon_j) \leq \mathsf{attacked}(\epsilon_k)$.

Proof. Let ϵ_j and ϵ_k be expectations, ρ be a report and Δ be a set of background facts, and Γ a set of access rules. If $\mathsf{consequent}(\epsilon_j) \preceq \mathsf{consequent}(\epsilon_k)$ then $\{\mathsf{consequent}(\epsilon_j)\} \vdash \mathsf{consequent}(\epsilon_k)$. Thus for all ρ if there is a grounding set Φ_k such that $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{ground}(\neg\mathsf{consequent}(\epsilon_k), \Phi_k)$ then there is a grounding set Φ_j such that $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{ground}(\neg\mathsf{consequent}(\epsilon_j), \Phi_j)$. Therefore $\mathsf{aset}(\epsilon_k, \Pi, \Gamma, \Delta) \subseteq \mathsf{aset}(\epsilon_j, \Pi, \Gamma, \Delta)$.

By Definition 3.2.2, if $\mathsf{aset}(\epsilon_k, \Pi, \Gamma, \Delta) \subseteq \mathsf{aset}(\epsilon_j, \Pi, \Gamma, \Delta)$, $\mathsf{attacked}(\epsilon_k, \Pi, \Gamma, \Delta) \leq \mathsf{attacked}(\epsilon_j, \Pi, \Gamma, \Delta)$. □

Example 4.2.2. *Let Γ be a set of access rules and Δ be a set of background facts. Let* $\epsilon_1 = \texttt{airline(x)} \rightarrow \texttt{profitable(x)} \wedge \texttt{ftse100(x)}$ *and* $\epsilon_2 = \texttt{airline(x)} \rightarrow \texttt{profitable(x)}$ *be expectations. Therefore* $\mathsf{consequent}(\epsilon_1) \vdash \mathsf{consequent}(\epsilon_2)$

Let Φ_1 and Φ_2 be grounding sets. Let $\mathsf{ground}(\epsilon_1, \Phi_1) = \texttt{airline(ba)} \rightarrow \texttt{profitable(ba)} \wedge \texttt{ftse100(ba)}$ *and* $\mathsf{ground}(\epsilon_2, \Phi_2) = \texttt{airline(ba)} \rightarrow \texttt{profitable(ba)}$

Let there be a report ρ_1 such that $\mathsf{representatives}(\rho_1, \Gamma, \Delta) \vdash \neg\texttt{ftse100(ba)}$. *Consequently, ρ_1 attacks ϵ_1 but not ϵ_2.*

Let there be a report ρ_2 such that $\mathsf{representatives}(\rho_2, \Gamma, \Delta) \vdash \neg\texttt{profitable(ba)}$. *Consequently, ρ_2 attacks ϵ_1 and ϵ_2.*

Thus all reports that attack ϵ_2 are in both $\mathsf{aset}(\epsilon_1, \Pi, \Gamma, \Delta)$ *and* $\mathsf{aset}(\epsilon, \Pi, \Gamma, \Delta)$, *but some reports that attack ϵ_2 may be in* $\mathsf{aset}(\texttt{airline(x)} \rightarrow \texttt{profitable(x)} \wedge \texttt{ftse100(x)}, \Pi, \Gamma, \Delta)$ *but not* $\mathsf{aset}(\texttt{airline(x)} \rightarrow \texttt{profitable(x)}, \Pi, \Gamma, \Delta)$. *Thus* $\mathsf{aset}(\epsilon_2, \Pi, \Gamma, \Delta) \subseteq \mathsf{aset}(\epsilon_1, \Pi, \Gamma, \Delta)$ *and* $\mathsf{attacked}(\epsilon_2, \Pi) \leq \mathsf{attacked}(\epsilon_1, \Pi, \Gamma, \Delta)$.

4.2.3 Expectation order and expectation values

Expectations with logically equivalent antecedents have identical fsets and expectations with logically equivalent consequents have identical asets. Logically equivalent expectations have identical vsets but do not necessarily have the same fset or asets. Therefore it is not enough to know that ϵ_1 is logically equivalent to ϵ_2 to determine the relationship between the values for ϵ_1 and the values for ϵ_2

Proposition 4.2.5. *Let ϵ_1 and ϵ_2 be two logically equivalent expectations.* $\mathsf{vset}(\epsilon_1, \Pi, \Gamma, \Delta) = \mathsf{vset}(\epsilon_2, \Pi, \Gamma, \Delta)$.

Proof. Let ϵ_j and ϵ_k be expectations, ρ be a report and Δ be a set of background facts, and Γ a set of access rules. If (ϵ_j) is logically equivalent to (ϵ_k) then for all ρ, if there is a grounding set Φ_k

such that representatives$(\rho, \Gamma, \Delta) \vdash$antecedent(ground$\epsilon_k, \Phi_k)) \wedge \neg$consequent(ground$\epsilon_k, \Phi_k))$ then there is a grounding set Φ_j such that representatives$(\rho, \Gamma, \Delta) \vdash$ antecedent(ground$\epsilon_j, \Phi_j)) \wedge$ $\neg$consequent(ground$\epsilon_j, \Phi_j)$). Likewise, for all ρ, if there is a grounding set Φ_j such that

$$\text{representatives}(\rho, \Gamma, \Delta) \vdash \text{antecedent}(\text{ground}\epsilon_j, \Phi_j)) \wedge \neg\text{consequent}(\text{ground}\epsilon_j, \Phi_j))$$

then there is a grounding set Φ_k such that

$$\text{representatives}(\rho, \Gamma, \Delta) \vdash \text{antecedent}(\text{ground}\epsilon_k, \Phi_k)) \wedge \neg\text{consequent}(\text{ground}\epsilon_k, \Phi_k))$$

Therefore $\text{vset}(\epsilon_k, \Pi, \Gamma, \Delta) = \text{vset}(\epsilon_j, \Pi, \Gamma, \Delta)$.

By Definition 3.2.4, if $\text{vset}(\epsilon_k, \Pi, \Gamma, \Delta) = \text{vset}(\epsilon_j, \Pi, \Gamma, \Delta)$, $\text{violated}(\epsilon_k, \Pi, \Gamma, \Delta) = \text{violated}(\epsilon_j, \Pi, \Gamma, \Delta)$. □

For two logically equivalent expectations, ϵ_1 and ϵ_2, the order of $\text{accuracy}(\epsilon_1, \Pi, \Gamma, \Delta)$, $\text{accuracy}(\epsilon_2, \Pi, \Gamma, \Delta)$ is determined solely by the consequence order of the antecedents of ϵ_1 and ϵ_2.

Proposition 4.2.6. *Let E be a set of expectations. For all $\epsilon_m, \epsilon_n \in E$, if ϵ_m is logically equivalent to ϵ_n and* $\text{antecedent}(\epsilon_m) \preceq \text{antecedent}(\epsilon_n)$ *then* $\text{accuracy}(\epsilon_m, \Pi, \Gamma, \Delta) \leq \text{accuracy}(\epsilon_n, \Pi, \Gamma, \Delta)$.

Proof. Let Π be a set of representative sets. Let ϵ_m and ϵ_n be logically equivalent expectations. Let $|\text{vset}(\epsilon_m, \Pi, \Gamma, \Delta)| = \text{vsetsize}$. By Proof 4.2.5, $\text{vset}(\epsilon_m, \Pi, \Gamma, \Delta) = \text{vset}(\epsilon_n, \Pi, \Gamma, \Delta)$. Consequently, $|\text{vset}(\epsilon_n, \Pi, \Gamma, \Delta)| = \text{vsetsize}$.

Let $\text{antecedent}(\epsilon_m) \preceq \text{antecedent}(\epsilon_n)$ then, by Proof 4.2.2, $\text{fset}(\epsilon_m, \Pi, \Gamma, \Delta) \subseteq \text{fset}(\epsilon_n, \Pi, \Gamma, \Delta)$. It follows from Definition 3.2.6 that $\text{accuracy}(\epsilon_m, \Pi, \Gamma, \Delta) \leq \text{accuracy}(\epsilon_n, \Pi, \Gamma, \Delta)$. □

It therefore follows that for a set of logically equivalent expectations, those expectations with the most specific antecedents will be the ones that are the least accurate, as the coverage for such expectations is lower than for expectations with more general antecedents, whilst the violated value remains the same, if the expectations are logically equivalent.

Example 4.2.3. *The following expectations are all logically equivalent:*

$$\epsilon_1 = \forall \mathbf{x}\ \mathbf{p}(\mathbf{x}) \wedge \mathbf{q}(\mathbf{x}) \rightarrow \mathbf{r}(\mathbf{x})$$
$$\epsilon_2 = \forall \mathbf{x}\ \mathbf{q}(\mathbf{x}) \wedge \neg\mathbf{r}(\mathbf{x}) \rightarrow \neg\mathbf{p}(\mathbf{x})$$
$$\epsilon_3 = \forall \mathbf{x}\ \mathbf{p}(\mathbf{x}) \rightarrow \neg\mathbf{q}(\mathbf{x}) \vee \mathbf{r}(\mathbf{x})$$
$$\epsilon_4 = \forall \mathbf{x}\ \neg\mathbf{r}(\mathbf{x}) \rightarrow \neg\mathbf{p}(\mathbf{x}) \vee \neg\mathbf{q}(\mathbf{x})$$

By Proposition 4.2.5, for any set of reports, Π, access rules Γ and background knowledge, Δ,

$\mathsf{vset}(\epsilon_1, \Pi, \Gamma, \Delta)$ *is identical to* $\mathsf{vset}(\epsilon_2, \Pi, \Gamma, \Delta)$, *which is identical to* $\mathsf{vset}(\epsilon_3, \Pi, \Gamma, \Delta)$, *which is identical to* $\mathsf{vset}(\epsilon_4, \Pi, \Gamma, \Delta)$.

By definition 4.1.2, $\mathsf{antecedent}(\epsilon_1) \preceq \mathsf{antecedent}(\epsilon_3)$ *and* $\mathsf{antecedent}(\epsilon_2) \preceq \mathsf{antecedent}(\epsilon_4)$. *According to Proposition 4.2.6,* $\mathsf{fset}(\epsilon_1, \Pi, \Gamma, \Delta) \subseteq \mathsf{fset}(\epsilon_3, \Pi, \Gamma, \Delta)$ *and* $\mathsf{fset}(\epsilon_2, \Pi, \Gamma, \Delta) \subseteq \mathsf{fset}(\epsilon_4, \Pi, \Gamma, \Delta)$.

As a result, $\mathsf{accuracy}(\epsilon_1, \Pi, \Gamma, \Delta) \leq \mathsf{accuracy}(\epsilon_3, \Pi, \Gamma, \Delta)$ *and* $\mathsf{accuracy}(\epsilon_2, \Pi, \Gamma, \Delta) \leq \mathsf{accuracy}(\epsilon_4, \Pi, \Gamma, \Delta)$.

Expectations with less general antecedents pose the strongest test for accuracy for the expectations. That is, it is much more difficult to fire such expectations. Such expectations also have greater explanatory power. That is, the less general the antecedent of an expectation, the more specific a description it is. For example $\forall \mathtt{x}\ \mathtt{british(x)} \wedge \mathtt{airline(x)}$ is more specific than $\forall \mathtt{x}\ \mathtt{british(x)}$. Therefore, less general antecedents are only fired in specific circumstances and, as such, have far greater discrimination between types of situation than general antecedents. It is therefore suggested that expectations with least general antecedents be favoured over those with shorter antecedents when creating the set of expectations.

For an expectation and its contrapositive, the vsets for those expectations are identical (see Proposition 4.2.5). However, as the antecedents are not comparable using the consequence relation, the order of the accuracy and confirmation values for contrapositive expectations is not fixed.

Proposition 4.2.7. *Let* ϵ_1, ϵ_2 *be expectations, such that* ϵ_1 *is the contrapositive of* ϵ_2. *Therefore* $\mathsf{accuracy}(\epsilon_1)$ *is independent of* $\mathsf{accuracy}(\epsilon_2)$, *that is, the order of* $\mathsf{accuracy}(\epsilon_1)$ *and* $\mathsf{accuracy}(\epsilon_2)$ *is not fixed.*

Proof. The following is a counterexample that demonstrates that $\mathsf{accuracy}(\epsilon_1)$ is independent of $\mathsf{accuracy}(\epsilon_2)$. Let $\epsilon_1 = \forall \bar{x}\ \alpha \rightarrow \beta$ and $\epsilon_2 = \forall \bar{x}\ \neg\beta \rightarrow \neg\alpha$.

Let

$\mathsf{representatives}(\rho_1, \Gamma, \Delta) \vdash \{\alpha, \neg\beta\}$
$\mathsf{representatives}(\rho_2, \Gamma, \Delta) \vdash \{\neg\alpha, \neg\beta\}$
$\mathsf{representatives}(\rho_3, \Gamma, \Delta) \vdash \{\alpha\}$
$\mathsf{representatives}(\rho_4, \Gamma, \Delta) \vdash \{\alpha\}$

After the receipt of ρ_2, the values for ϵ_1 and ϵ_2 are as follows:

	accuracy	coverage
ϵ_1	0	1
ϵ_2	0.5	2

After receipt of ρ_4 the values for ϵ_1 and ϵ_2 are as follows:

	accuracy	coverage
ϵ_1	0.66	3
ϵ_2	0.5	2

Therefore the accuracy of ϵ_1 is independent of the accuracy of ϵ_2. □

If two expectations have logically equivalent antecedents, but the consequent of one expectation is more general than the other, the expectation with the more general consequent is as accurate, or more so than the expectation with the less general consequent.

Proposition 4.2.8. *Let* ϵ_1, ϵ_2 *be expectations. Let* $\mathsf{antecedent}(\epsilon_1)$ *be logically equivalent to* $\mathsf{antecedent}(\epsilon_2)$. *Let* $\mathsf{consequent}(\epsilon_1) \preceq \mathsf{consequent}(\epsilon_2)$. *It therefore follows that* $\mathsf{accuracy}(\epsilon_1, \Pi, \Gamma, \Delta) \leq \mathsf{accuracy}(\epsilon_2, \Pi, \Gamma, \Delta)$.

Proof. By Proof 4.2.1, given that $\mathsf{antecedent}(\epsilon_1)$ is logically equivalent to $\mathsf{antecedent}(\epsilon_2)$, $\mathsf{fset}(\epsilon_1, \Pi, \Gamma, \Delta) = \mathsf{fset}(\epsilon_2, \Pi, \Gamma, \Delta)$. By Proof 4.2.4, as $\mathsf{consequent}(\epsilon_1) \preceq \mathsf{consequent}(\epsilon_2)$, $\mathsf{aset}(\epsilon_2, \Pi, \Gamma, \Delta) \subseteq \mathsf{aset}(\epsilon_1, \Pi, \Gamma, \Delta)$ and so $\mathsf{vset}(\epsilon_2, \Pi, \Gamma, \Delta) \subseteq \mathsf{vset}(\epsilon_1, \Pi, \Gamma, \Delta)$.

It follows from Definition 3.2.6 that $\mathsf{accuracy}(\epsilon_1, \Pi, \Gamma, \Delta) \leq \mathsf{accuracy}(\epsilon_2, \Pi, \Gamma, \Delta)$. □

Generalising the antecedent of an expectation whilst keeping the consequent constant results in an unpredictable change in the confirmed and accuracy value for those expectations. This is because the change in antecedent results in both a larger vset and a larger aset, but the increase in the size of these sets is not necessarily proportionate.

Proposition 4.2.9. *Let* E *be a set of expectations. For all* $\epsilon_j, \epsilon_k \in E$ *if* $\mathsf{antecedent}(\epsilon_j) \preceq \mathsf{antecedent}(\epsilon_k)$ *and* $\mathsf{consequent}(\epsilon_j)$ *is logically equivalent to* $\mathsf{consequent}(\epsilon_k)$ *then* $\mathsf{fset}(\epsilon_j, \Pi, \Gamma, \Delta) \subseteq \mathsf{fset}(\epsilon_k, \Pi, \Gamma, \Delta)$ *and* $\mathsf{vset}(\epsilon_j, \Pi, \Gamma, \Delta) \subseteq \mathsf{vset}(\epsilon_k, \Pi, \Gamma, \Delta)$

Proof. By Proof 4.2.3, as $\mathsf{consequent}(\epsilon_1)$ is logically equivalent to $\mathsf{consequent}(\epsilon_2)$, $\mathsf{aset}(\epsilon_2, \Pi, \Gamma, \Delta) = \mathsf{aset}(\epsilon_1, \Pi, \Gamma, \Delta)$. By Proof 4.2.2, given that $\mathsf{antecedent}(\epsilon_1) \preceq \mathsf{antecedent}(\epsilon_2)$, $\mathsf{fset}(\epsilon_1, \Pi, \Gamma, \Delta) \subseteq \mathsf{fset}(\epsilon_2, \Pi, \Gamma, \Delta)$ and so it follows from Definition 3.2.4 that $\mathsf{vset}(\epsilon_2, \Pi, \Gamma, \Delta) \subseteq \mathsf{vset}(\epsilon_1, \Pi, \Gamma, \Delta)$. □

Generalising the antecedent of an expectation leads to an unpredictable change in the accuracy and confirmation values. By Proof 4.2.2, a generalisation in antecedent leads to a larger fset. By Proof 4.2.9, a generalisation in antecedent leads to a larger vset and cset. Therefore, generalising the antecedent of an expectation leads to an increase in both the numerator and the denominator of the equation $\mathsf{accuracy}(\epsilon, \Pi, \Gamma, \Delta) = 1 - \frac{|\mathsf{vset}(\epsilon, \Pi, \Gamma, \Delta)|}{|\mathsf{fset}(\epsilon, \Pi, \Gamma, \Delta)|}$. Consequently $\mathsf{confirmation}(\epsilon_1, \Pi, \Gamma, \Delta)$ is independent of $\mathsf{confirmation}(\epsilon_2, \Pi, \Gamma, \Delta)$ and $\mathsf{accuracy}(\epsilon_1, \Pi, \Gamma, \Delta)$ is independent of $\mathsf{accuracy}(\epsilon_2, \Pi, \Gamma, \Delta)$:

Proposition 4.2.10. *Let E be a set of expectations. For all $\epsilon_j, \epsilon_k \in E$, if* antecedent$(\epsilon_j) \preceq$ antecedent(ϵ_k) *and* consequent(ϵ_j) *is logically equivalent to* consequent(ϵ_k) *then* accuracy$(\epsilon_j, \Pi, \Gamma, \Delta)$ *is independent of* accuracy$(\epsilon_k, \Pi, \Gamma, \Delta)$.

Proof. The following is a counterexample:

Let $\epsilon_1 = \forall \bar{x}\ \alpha \rightarrow \gamma$ and $\epsilon_2 = \forall \bar{x}\ \alpha \wedge \beta \rightarrow \gamma$. Let
representatives$(\rho_1, \Gamma, \Delta) \vdash \{\alpha, \neg\gamma\}$
representatives$(\rho_2, \Gamma, \Delta) \vdash \{\alpha, \beta\}$
representatives$(\rho_3, \Gamma, \Delta) \vdash \{\alpha, \beta, \neg\gamma\}$
representatives$(\rho_4, \Gamma, \Delta) \vdash \{\alpha\}$
representatives$(\rho_5, \Gamma, \Delta) \vdash \{\alpha\}$

After the receipt of ρ_3, the values for ϵ_1 and ϵ_2 are as follows:

	accuracy	coverage
ϵ_1	0.33	3
ϵ_2	0.5	2

After the receipt of ρ_3, the values for ϵ_1 and ϵ_2 are as follows:

	accuracy	coverage
ϵ_1	0.6	5
ϵ_2	0.5	2

Therefore the accuracy of ϵ_1 is independent of the accuracy of ϵ_2. □

If two expectations, ϵ_1 and ϵ_2 differ such that antecedent$(\epsilon_1) \preceq$ antecedent(ϵ_2) and consequent$(\epsilon_1) \preceq$ consequent(ϵ_1), it is tempting to believe that e_2 must be more accurate than e_1. By Proof 4.2.2 the more general antecedent will have a larger fset than the more specific antecedent and by Proof 4.2.4, the more general consequent will have a smaller aset than the more specific consequent. However this need not be the case. This means that it is not possible to determine, *a priori* the order of the accuracy values of those expectations.

Proposition 4.2.11. *For any two expectations, $\epsilon_j, \epsilon_k \in E$, if* antecedent$(\epsilon_j) \preceq$ antecedent(ϵ_k) *and* consequent$(\epsilon_j) \preceq$ consequent(ϵ_k) *then ϵ_k is more general than ϵ_j. That is to say, ϵ_k will be fired more and attacked less than ϵ_j.*

The order of accuracy$(\epsilon_j, \Pi, \Gamma, \Delta)$ *and* accuracy$(\epsilon_k, \Pi, \Gamma, \Delta)$ *cannot be predicted from the consequence order of* antecedent(ϵ_j) *and* antecedent(ϵ_k).

Proof. By Proof 4.2.2, if antecedent$(\epsilon_j) \preceq$ antecedent(ϵ_k) then fired$(\epsilon_j, \Pi, \Gamma, \Delta) \leq$ fired$(\epsilon_k, \Pi, \Gamma, \Delta)$. Therefore the denominator in the formula

$$\text{accuracy}(\epsilon_j, \Pi, \Gamma, \Delta) = 1 - \frac{|\text{violated}(\epsilon_j, \Pi, \Gamma, \Delta)|}{|\text{fired}(\epsilon_j, \Pi, \Gamma, \Delta)|}$$

is smaller than the denominator in

$$\mathsf{accuracy}(\epsilon_k, \Pi, \Gamma, \Delta) = 1 - \frac{|\mathsf{violated}(\epsilon_k, \Pi, \Gamma, \Delta)|}{|\mathsf{fired}(\epsilon_k, \Pi, \Gamma, \Delta)|}$$

However, it is possible that the $\mathsf{vset}(\epsilon_k, \Pi, \Gamma, \Delta) \leq \mathsf{vset}(\epsilon_j, \Pi, \Gamma, \Delta)$, thus also increasing the numerator. As $\mathsf{consequent}(\epsilon_j) \preceq \mathsf{consequent}(\epsilon_k)$, it therefore follows that $\mathsf{attacked}(\epsilon_k, \Pi, \Gamma, \Delta) \leq \mathsf{attacked}(\epsilon_j, \Pi, \Gamma, \Delta)$.

If $\mathsf{aset}(\epsilon_j, \Pi, \Gamma, \Delta) \not\subseteq \mathsf{vset}(\epsilon_k, \Pi, \Gamma, \Delta)$ and $\mathsf{aset}(\epsilon_k, \Pi, \Gamma, \Delta) \not\subseteq \mathsf{vset}(\epsilon_j, \Pi, \Gamma, \Delta)$ then the inequality in size between $\mathsf{aset}(\epsilon_k, \Pi, \Gamma, \Delta)$ and $\mathsf{aset}(\epsilon_j, \Pi, \Gamma, \Delta)$ cannot guarantee that $\mathsf{vset}(\epsilon_k, \Pi, \Gamma, \Delta) < \mathsf{vset}(\epsilon_j, \Pi, \Gamma, \Delta)$.

As the difference in size between $\mathsf{vset}(\epsilon_k, \Pi, \Gamma, \Delta)$ and $\mathsf{vset}(\epsilon_k, \Pi, \Gamma, \Delta)$ is not wholly dependent on the difference in size between $\mathsf{fset}(\epsilon_k, \Pi, \Gamma, \Delta)$ and $\mathsf{fset}(\epsilon_k, \Pi, \Gamma, \Delta)$, the order of $\mathsf{accuracy}(\epsilon_j, \Pi, \Gamma, \Delta)$ and $\mathsf{accuracy}(\epsilon_k, \Pi, \Gamma, \Delta)$ cannot be predicted from the order of $\mathsf{antecedent}(\epsilon_j)$ and $\mathsf{antecedent}(\epsilon_k)$ □

Example 4.2.4. *Let* $\epsilon_1 = \mathtt{airline(x)} \rightarrow \mathtt{profitable(x)}$ *and let* $\epsilon_2 = \mathtt{airline(x)} \wedge \mathtt{british(x)} \rightarrow \mathtt{profitable(x)} \wedge \mathtt{listed(x)}$

Let ρ_1 *be a report such that*

$$\begin{aligned}&\mathsf{representatives}(\rho_1, \Gamma, \Delta) \vdash \\ &\qquad \mathtt{airline(airWales)} \wedge \mathtt{british(airWales)} \wedge \\ &\qquad \mathtt{profitable(airWales)} \wedge \neg\mathtt{listed(airWales)}\end{aligned}$$

As a result, values for the two expectations are given in Table 4.1:

	fired	violated	accuracy
ϵ_1	1	0	1
ϵ_2	1	1	0

Table 4.1: Values for ϵ_1 and ϵ_2 after receipt of ρ_1.

Let ρ_2, ρ_3 *and* ρ_4 *be additional reports such that*

$$\begin{aligned}&\mathsf{representatives}(\rho_2, \Gamma, \Delta) \vdash \\ &\qquad \mathtt{airline(BA)} \wedge \mathtt{british(BA)} \wedge \mathtt{profitable(BA)} \wedge \mathtt{listed(BA)} \\ &\mathsf{representatives}(\rho_3, \Gamma, \Delta) \vdash \\ &\qquad \mathtt{airline(buzz)} \wedge \neg\mathtt{profitable(buzz)} \\ &\mathsf{representatives}(\rho_4, \Gamma, \Delta) \vdash \\ &\qquad \mathtt{airline(swissAir)} \wedge \neg\mathtt{profitable(swissAir)}\end{aligned}$$

The values for the two expectations are shown in Table 4.2:

	fired	violated	accuracy
ϵ_1	4	2	0.5
ϵ_2	2	1	0.5

Table 4.2: Values for ϵ_1 and ϵ_2 after receipt of ρ_4.

Let report ρ_5 *contain facts such that* representatives$(\rho_5, \Gamma, \Delta) \vdash$ `airline(air2000)`$\wedge$ `british(air2000)` $\wedge$ `profitable(air2000)` $\wedge$ `listed(air2000)`. *The updated values for the two expectations are now as in Table 4.3:*

	fired	violated	accuracy
ϵ_1	5	2	0.6
ϵ_2	3	1	0.66

Table 4.3: Values for ϵ_1 and ϵ_2 after receipt of ρ_5.

So we see that, on receipt of ρ_1, ϵ_1 *was the more accurate expectation. By receipt of* ρ_5, ϵ_2 *was the more accurate. This example illustrates that there is no way of predicting the relative strengths of these expectations merely by knowing the relative order of their antecedents and their consequents.*

If two expectations, ϵ_1 and ϵ_2, differ such that antecedent$(\epsilon_1) \preceq$ antecedent(ϵ_2) and consequent$(\epsilon_2) \preceq$ consequent(ϵ_1), by Proof 4.2.2 fset$(\epsilon_1, \Pi, \Gamma, \Delta) \leq$ fset$(\epsilon_2, \Pi, \Gamma, \Delta)$ and by Proof 4.2.4, aset$(\epsilon_2, \Pi, \Gamma, \Delta) \leq$ aset$(\epsilon_1, \Pi, \Gamma, \Delta)$. Again it is not possible to predict the relative accuracy and confirmation values for these expectations.

Proposition 4.2.12. *Let* $\epsilon_j, \epsilon_k \in E$. *If* antecedent$(\epsilon_j) \preceq$ antecedent(ϵ_k), *and* consequent$(\epsilon_k) \preceq$ consequent(ϵ_j) *and* ϵ_j *is not logically equivalent to* ϵ_k, accuracy$(\epsilon_j, \Pi, \Gamma, \Delta)$ *is independent of* accuracy$(\epsilon_k, \Pi, \Gamma, \Delta)$.

Proof. By Proof 4.2.2, if antecedent$(\epsilon_j) \preceq$ antecedent(ϵ_k) then fired$(\epsilon_j, \Pi, \Gamma, \Delta) \leq$ fired$(\epsilon_k, \Pi, \Gamma, \Delta)$. Therefore the denominator in the formula

$$\text{accuracy}(\epsilon_j, \Pi, \Gamma, \Delta) = 1 - \frac{|\text{violated}(\epsilon_j, \Pi, \Gamma, \Delta)|}{|\text{fired}(\epsilon_j, \Pi, \Gamma, \Delta)|}$$

is smaller than the denominator in

$$\text{accuracy}(\epsilon_k, \Pi, \Gamma, \Delta) = 1 - \frac{|\text{violated}(\epsilon_k, \Pi, \Gamma, \Delta)|}{|\text{fired}(\epsilon_k, \Pi, \Gamma, \Delta)|}$$

However, the numerator may grow or shrink, as there is no requirement that if vset$(\epsilon_j, \Pi, \Gamma, \Delta) =$

$\emptyset$ then $\text{vset}(\epsilon_k, \Pi, \Gamma, \Delta) = \emptyset$ also. Therefore it is possible for $\text{vset}(\epsilon_j) = 0$, resulting in $\text{accuracy}(\epsilon_j) = 100\%$ while $\text{vset}(\epsilon_j) > 0$, resulting in $\text{accuracy}(\epsilon_j) < 100\%$ and it is also possible that $\text{vset}(\epsilon_j) > 0$, resulting in $\text{accuracy}(\epsilon_j) < 100\%$ while $\text{vset}(\epsilon_j) = 0$, resulting in $\text{accuracy}(\epsilon_j) = 100\%$ □

Example 4.2.5. *Let* $\epsilon_1 = \forall \mathtt{x}\ \mathtt{airline(x)} \wedge \mathtt{british(x)} \rightarrow \mathtt{profitable(x)}$ *and let* $\epsilon_2 = \forall \mathtt{x}\ \mathtt{airline(x)} \rightarrow \mathtt{profitable(x)} \wedge \mathtt{listed(x)}$.

Let there be a report ρ_1 *such that* $\text{representatives}(\rho_1, \Gamma, \Delta) \vdash \mathtt{airline(airWales)} \wedge \mathtt{british(airWales)} \wedge \mathtt{profitable(airWales)} \wedge \neg \mathtt{listed(airWales)}$

As a result, values for the two expectations are as in table 4.4:

	fired	violated	accuracy
ϵ_1	1	0	1
ϵ_2	1	1	0

Table 4.4: Values for ϵ_1 and ϵ_2 after receipt of ρ_1.

Let there be additional reports ρ_2, *and* ρ_3 *such that* $\text{representatives}(\rho_2, \Gamma, \Delta) \vdash \mathtt{airline(BA)} \wedge \mathtt{british(BA)} \wedge \mathtt{profitable(BA)} \wedge \mathtt{listed(BA)}$ *and* $\text{representatives}(\rho_3, \Gamma, \Delta) \vdash \mathtt{airline(bmi)} \wedge \mathtt{british(bmi)} \wedge \neg \mathtt{profitable(bmi)}$. *The values for the two expectations are once again updated:*

	fired	violated	accuracy
ϵ_1	3	1	0.66
ϵ_2	3	2	0.33

Table 4.5: Values for ϵ_1 and ϵ_2 after receipt of ρ_3.

Let ρ_4, ρ_5, ρ_6 *and* ρ_7 *be a sequence of reports about profitable airlines whose nationality is not known to be British such that:*

$$\text{representatives}(\rho_4, \Gamma, \Delta) \vdash \mathtt{airline(quantas)} \wedge \mathtt{profitable(quantas)}$$

$$\text{representatives}(\rho_5, \Gamma, \Delta) \vdash \mathtt{airline(emirates)} \wedge \mathtt{profitable(emirates)}$$

$$\text{representatives}(\rho_6, \Gamma, \Delta) \vdash \mathtt{airline(alitalia)} \wedge \mathtt{profitable(alitalia)}$$

$$\text{representatives}(\rho_4, \Gamma, \Delta) \vdash \mathtt{airline(easyjet)} \wedge \neg \mathtt{profitable(easyjet)}$$

The values for the two expectations are as shown in Table 4.6:

	fired	violated	accuracy
ϵ_1	3	1	0.66
ϵ_2	7	2	0.715

Table 4.6: Values for ϵ_1 and ϵ_2 after receipt of ρ_7.

So we see that, on receipt of ρ_1, ϵ_1 was the more accurate expectation. By receipt of ρ_7, ϵ_2 was the more accurate. This example illustrates that there is no way of predicting the relative strengths of these expectations merely by knowing the relative order of their antecedents and their consequents.

4.2.4 Special case expectations

There are expectations that, due to their syntax, have values that are predictable. Definitions for and proofs regarding these types of expectations are given below:

Definition 4.2.1. *Let ϵ be an expectation such that* $\{\mathsf{antecedent}(\epsilon)\} \vdash \bot$. *$\epsilon$ is a non-firing expectation.*

Example 4.2.6. *The expectation $\forall x\ \alpha(x) \wedge \neg\alpha(x) \rightarrow \beta(x)$ is a non-firing expectation.*

Proposition 4.2.13. *A non-firing expectation is an expectation whose fired value is always zero and whose accuracy is consequently undetermined.*

Proof. Let ϵ be an expectation such that $\{\mathsf{antecedent}(\epsilon)\} \vdash \bot$.Recall from Definition 2.4.5 that any representative set $\mathsf{representatives}(\rho, \Gamma, \Delta) \nvdash \bot$. Therefore no representative set $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\epsilon)$. As a result, $\mathsf{fired}(\epsilon, \Pi, \Gamma, \Delta) = 0$ for any set of reports, Π, access rules, Γ, and background knowledge, Δ. Recall from Definition 3.2.6 that for any ϵ such that $\mathsf{fired}(\epsilon, \Pi, \Gamma, \Delta) = 0$, $\mathsf{accuracy}(\epsilon, \Pi, \Gamma, \Delta) = \mathit{undetermined}$. □

Definition 4.2.2. *Let ϵ be an expectation such that* $\{\mathsf{consequent}(\epsilon)\} \vdash \bot$. *$\epsilon$ is an automatically violated expectation.*

Example 4.2.7. *The expectation $\forall x\ \alpha(x) \rightarrow \beta(x) \wedge \neg\beta(x)$ is an automatically violated expectation.*

Proposition 4.2.14. *Let ϵ be an automatically violated expectation.* $\mathsf{accuracy}(\epsilon) = 0$ *or is undetermined*

Proof. If ϵ is an automatically violated expectation then $\{\mathsf{consequent}(\epsilon)\} \vdash \bot$ and, therefore, $\{\neg\mathsf{consequent}(\epsilon)\} \vdash \top$. Given that $\emptyset \vdash \top$ and that for any $\mathsf{representatives}(\rho, \Gamma, \Delta)$, $\emptyset \subseteq \mathsf{representatives}(\rho, \Gamma, \Delta)$, $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \neg\mathsf{consequent}(\epsilon)$.

As a result, any representatives(ρ, Γ, Δ) that fires ϵ also violates ϵ. Therefore accuracy$(\epsilon, \Pi, \Gamma, \Delta) =$ 0 if ϵ has been fired, or is undetermined otherwise. □

Definition 4.2.3. *Let ϵ be an expectation such that* $\{\text{antecedent}(\epsilon)\} \vdash \top$. *$\epsilon$ is an automatically firing expectation.*

Example 4.2.8. *The expectation $\forall x\ \alpha(x) \vee \neg\alpha(x) \rightarrow \beta(x)$ is an automatically firing expectation.*

Proposition 4.2.15. *Let ϵ be an automatically firing expectation. For any set of reports, Π, set of access rules, Γ and set of background knowledge, Δ, for all $\rho \in \Pi$ there exists a grounding set Φ such that* representatives$(\rho, \Gamma, \Delta) \vdash$ antecedent(ground(ϵ, Φ)).

Proof. If ϵ is an automatically firing expectation then $\{\text{antecedent}(\epsilon)\} \vdash \top$. Given that $\emptyset \vdash \top$ and that for any representatives(ρ, Γ, Δ), $\emptyset \subseteq$ representatives(ρ, Γ, Δ), representatives$(\rho, \Gamma, \Delta) \vdash$ antecedent(ϵ). As a result, any representatives(ρ, Γ, Δ) fires ϵ. □

Definition 4.2.4. *Let ϵ be an expectation such that* $\{\text{consequent}(\epsilon)\} \vdash \top$. *$\epsilon$ is an inviolable expectation.*

Example 4.2.9. *The expectation $\forall x\ \alpha(x) \rightarrow \beta(x) \vee \neg\beta(x)$ is an inviolable expectation.*

Proposition 4.2.16. *Let ϵ be an inviolable expectation.* accuracy$(\epsilon) = 1$ *or is undetermined.*

Proof. If ϵ is an inviolable expectation then $\{\text{consequent}(\epsilon)\} \vdash \top$, and $\{\neg\text{consequent}(\epsilon)\} \vdash \bot$. Recall from Definition 2.4.5 that any representative set representatives$(\rho, \Gamma, \Delta) \nvdash \bot$ Therefore no representative set representatives$(\rho, \Gamma, \Delta) \vdash \neg$consequent$(\epsilon)$. As a result, violated$(\epsilon, \Pi, \Gamma, \Delta) =$ 0 for any set of reports, Π, access rules, Γ, and background knowledge, Δ. Therefore accuracy$(\epsilon, \Pi, \Gamma, \Delta) = 1$ or is undetermined. □

Definition 4.2.5. *Let ϵ be an expectation such that* $\{\text{antecedent}(\epsilon)\} \vdash \neg\text{consequent}(\epsilon)$. *$\epsilon$ is a self-defeating expectation.*

Example 4.2.10. *The expectation $\forall x\ \alpha(x) \rightarrow \neg\alpha(x)$ is a self-defeating expectation.*

Proposition 4.2.17. *Let ϵ be a self-defeating expectation.* accuracy$(\epsilon) = 0$ *or is undetermined.*

Proof. Let representatives$(\rho, \Gamma, \Delta) \vdash$ antecedent(ϵ). Because the consequence relation is transitive, if antecedent$(\epsilon) \vdash \neg$consequent(ϵ) then representatives$(\rho, \Gamma, \Delta) \vdash \neg$consequent$(\epsilon)$. Therefore for all representatives$(\rho, \Gamma, \Delta) \in$ fset$(\epsilon, \Pi, \Gamma, \Delta)$, representatives$(\rho, \Gamma, \Delta) \in$ vset$(\epsilon, \Pi, \Gamma, \Delta)$. As a result accuracy$(\epsilon, \Pi, \Gamma, \Delta) = 0$ if ϵ is fired, or undetermined otherwise. □

Definition 4.2.6. *Let ϵ be an expectation such that* $\{\text{antecedent}(\epsilon)\} \vdash$ consequent(ϵ). *ϵ is a self-reinforcing expectation.*

Example 4.2.11. *Expectation $\forall x\ \alpha(x) \rightarrow \alpha(x)$ is a self-reinforcing expectation.*

Proposition 4.2.18. *Let ϵ be a self-reinforcing expectation.* $\mathsf{accuracy}(\epsilon) = 1$ *or is undetermined.*

Proof. Let $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\epsilon)$. Because the consequence relation is transitive, if $\mathsf{antecedent}(\epsilon) \vdash \mathsf{consequent}(\epsilon)$ then $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{consequent}(\epsilon)$. Therefore for all $\mathsf{representatives}(\rho, \Gamma, \Delta) \in \mathsf{fset}(\epsilon, \Pi, \Gamma, \Delta)$, $\mathsf{representatives}(\rho, \Gamma, \Delta) \notin \mathsf{vset}(\epsilon, \Pi, \Gamma, \Delta)$. As a result $\mathsf{accuracy}(\epsilon, \Pi, \Gamma, \Delta) = 1$ if ϵ is fired, or undetermined otherwise. □

It is therefore possible to conclude that there expectations that are of no use in determining whether a report is interesting. The values for such expectations are determined wholly by their syntax.

4.3 Reducing the information held on the set of expectations

The set of expectations is very large and therefore it is necessary to ensure that the amount of information that is held concerning each expectation is kept to a minimum. The set of antecedents and consequents for a language $\mathcal{L}$ can be divided in to three subsets, the set of quantified literals, the set of disjunctive formulae and the set of conjunctive formulae. Due to the properties of the expectations in E it is possible to record only the fset, aset and sset values for the set of quantified literals. This section will demonstrate that it is possible to derive an upper bound on the fset, aset and sset for any other expectations in E from these sets by determining the unions or intersections of some of these sets.

There is a set of quantified formulae in M such that all members of the subset are quantified literals. This will be referred to as the *literal layer.*

Definition 4.3.1. *Let M be the set of all unground, universally quantified formulae for a language that includes only predicate symbols and variables. The set of unground, universally quantified literals in M , denoted* $\mathsf{literalLayer}(M)$*, is*

$$\mathsf{literalLayer}(M) = \{\forall \bar{x}\ \alpha, \forall \bar{x}\ \neg\alpha | \forall \bar{x}\ \alpha \textit{ and } \forall \bar{x}\ \neg\alpha \in M \textit{ and } \forall \bar{x}\ \alpha \textit{ is an unground literal}\}$$

Definition 4.3.2. *Let* $\mathsf{literalLayer}(A)$ *be the set of all unground, universally quantified literals up to arity n for a language. The set of conjunctive formulae in A such that no two logically equivalent formulae are in that set, denoted* $\mathsf{conjunctiveFormulae}(M)$*, is as follows:*

$$\begin{aligned}\mathsf{conjunctiveFormulae}(M) = \{&\forall \bar{x}\alpha \in M | \\ &\forall \bar{x}\alpha \vdash \forall \bar{x}_1, ..., \bar{x}_k\ \alpha_1 \wedge ... \wedge \alpha_k \textit{ and} \\ &\forall \bar{x}_1\ \alpha_1, ..., \forall \bar{x}_k\ \alpha_k \in \mathsf{literalLayer}(M)\}\end{aligned}$$

There is a set of quantified formulae in a given language that are logically equivalent to disjunc-

tions of the quantified atoms in literalLayer(M) for that language.

Definition 4.3.3. *Let* literalLayer(M) *be the set of all unground, universally quantified literals up to arity n for a language. The set of disjunctive formulae in A, denoted* disjunctiveFormulae(M), *is as follows:*

$$\begin{aligned}\text{disjunctiveFormulae}(M) = \{&\forall\bar{x}\alpha \in M \mid \\ &\forall\bar{x}\alpha \vdash \forall\bar{x}_1,...,\bar{x}_k\ \alpha_1 \vee ... \vee \alpha_k \text{ and} \\ &\forall\bar{x}_1\ \alpha_1,...,\forall\bar{x}_k\ \alpha_k \in \text{literalLayer}(M)\}\end{aligned}$$

The set M is large even for quite restricted languages. Therefore the less information that needs to be stored regarding the members of the set M, the better. It is not necessary to record fset or aset values for antecedents or consequents that are not members of quantifiedLiterals(M) (see Definition 4.3.1). It is only necessary to record the fsets and asets for the atomic antecedents and consequents, that is, those in literalLayer(M). These can be used to place bounds on the fsets and asets for all disjunctive and conjunctive antecedents and consequents.

For any expectation with a conjunctive antecedent, the fset for that antecedent is the intersection of the fsets of the single literals that are the conjuncts in that antecedent. Note that in the following proposition and proof the tuples $\bar{x}_1...\bar{x}_k$ are assumed to be disjoint. That is, for all $\bar{x}_i...\bar{x}_j$ in $\bar{x}_1...\bar{x}_k$, if a variable is in $\bar{x}_i$ then it is not in $\bar{x}_k$. As such the conjunctive antecedent $\forall\bar{x}\ \alpha_1\wedge...\wedge\alpha_k$ is ground by the same grounding sets that ground $\forall\bar{x}_1\ \alpha_1...\forall\bar{x}_k\ \alpha_k$.

Proposition 4.3.1. *Let* ϵ *be an expectation. Let* antecedent(ϵ) $= \forall\bar{x}\ \alpha_1 \wedge ... \wedge \alpha_k$, *let* consequent($\epsilon$) $= \beta$, $\{\forall\bar{x}_1\ \alpha_1...\forall\bar{x}_j\ \alpha_k\} \subseteq$ quantifiedLiterals(M) *and let* $\bar{x} = \bar{x}_1 \oplus ... \oplus \bar{x}_k$. *Let* β *be any consequent. It follows that* fset($\epsilon, \Pi, \Gamma, \Delta$) = fset($\forall\bar{x}_1\ \alpha_1 \rightarrow \beta, \Pi, \Gamma, \Delta$) $\cap ... \cap$ fset($\forall\bar{x}_k\ \alpha_k \rightarrow \beta, \Pi, \Gamma, \Delta$).

Proof. The reports in fset($\epsilon, \Pi, \Gamma, \Delta$) are exactly those ρ for which there is a grounding set Φ such that representatives(ρ, Γ, Δ) $\vdash$ antecedent(ground(ϵ, Φ)) which is exactly that set of those ρ for which representatives(ρ, Γ, Δ) $\vdash$ ground($\forall\bar{x}_1\ \alpha_1, \Phi$) $\wedge ... \wedge$ ground($\forall\bar{x}_k\ \alpha_k, \Phi$). Thus fset($\epsilon, \Pi, \Gamma, \Delta$) =

$$\text{fset}(\forall\bar{x}_1\ \alpha_1 \rightarrow \text{consequent}(\epsilon), \Pi, \Gamma, \Delta) \cap ... \cap \text{fset}(\forall\bar{x}_k\ \alpha_k \rightarrow \text{consequent}(\epsilon), \Pi, \Gamma, \Delta)$$

□

Example 4.3.1. *Let* Γ *be a set of access rules and* Δ *be a set of background facts. Let* β *be some arbitrary consequent,*

$$\begin{aligned}\epsilon_1 &= \forall\mathtt{x}\ \mathtt{airline(x)} \rightarrow \beta \\ \epsilon_2 &= \forall\mathtt{y}\ \mathtt{ftse100(y)} \rightarrow \beta \\ \epsilon_3 &= \forall\mathtt{x},\mathtt{y}\ \mathtt{airline(x)} \wedge \mathtt{ftse100(y)} \rightarrow \beta\end{aligned}$$

Let Π *be a set of reports* $\{\rho_1, \rho_2, \rho_3, \rho_4\}$ *such that*

$$\begin{aligned}
&\mathsf{representatives}(\rho_1, \Gamma, \Delta) \vdash \texttt{airline(ba)},\\
&\mathsf{representatives}(\rho_2, \Gamma, \Delta) \vdash \texttt{ftse100(ba)},\\
&\mathsf{representatives}(\rho_3, \Gamma, \Delta) \vdash \texttt{airline(ba)} \wedge \texttt{ftse100(ba)}\\
&\mathsf{representatives}(\rho_4, \Gamma, \Delta) \vdash \texttt{airline(ryanair)} \wedge \texttt{ftse100(marksAndSpencer)}
\end{aligned}$$

It follows that

$$\begin{aligned}
&\mathsf{fset}(\epsilon_1, \Pi, \Gamma, \Delta) = \{\rho_1, \rho_3, \rho_4\},\\
&\mathsf{fset}(\epsilon_2, \Pi, \Gamma, \Delta) = \{\rho_2, \rho_3, \rho_4\},\\
&\mathsf{fset}(\epsilon_3, \Pi, \Gamma, \Delta) = \{\rho_3, \rho_4\},\\
&\mathsf{fset}(\epsilon_1, \Pi, \Gamma, \Delta) \cap \mathsf{fset}(\epsilon_2, \Pi, \Gamma, \Delta) = \{\rho_3, \rho_4\}
\end{aligned}$$

However, there are useful antecedents in which variables are repeated across the conjuncts. For example, $\epsilon_4 = \forall \mathtt{x}\ \mathtt{airline(x)} \wedge \mathtt{ftse100(x)} \rightarrow \mathsf{consequent}(\epsilon_4)$. It is the case that $\mathsf{fset}(\epsilon_4, \Pi, \Gamma, \Delta) \subseteq \mathsf{fset}(\epsilon_3, \Pi, \Gamma, \Delta)$ from the example given above. Therefore $|\mathsf{fset}(\epsilon_3, \Pi, \Gamma, \Delta)|$ is an upper bound on the size of $\mathsf{fset}(\epsilon_4, \Pi, \Gamma, \Delta)$.

Example 4.3.2. *Let* Γ *be a set of access rules and* Δ *be a set of background facts. Let* β *be an arbitrary consequent and*

$$\begin{aligned}
\epsilon_1 &= \forall \mathtt{x}\ \mathtt{airline(x)} \rightarrow \beta\\
\epsilon_2 &= \forall \mathtt{y}\ \mathtt{ftse100(y)} \rightarrow \beta\\
\epsilon_3 &= \forall \mathtt{x,y}\ \mathtt{airline(x)} \wedge \mathtt{ftse100(y)} \rightarrow \beta\\
\epsilon_4 &= \forall \mathtt{x}\ \mathtt{airline(x)} \wedge \mathtt{ftse100(x)} \rightarrow \beta
\end{aligned}$$

Let Π *be a set of reports* $\{\rho_1, \rho_2, \rho_3, \rho_4\}$ *such that*

$$\begin{aligned}
&\mathsf{representatives}(\rho_1, \Gamma, \Delta) \vdash \texttt{airline(ba)},\\
&\mathsf{representatives}(\rho_2, \Gamma, \Delta) \vdash \texttt{ftse100(ba)},\\
&\mathsf{representatives}(\rho_3, \Gamma, \Delta) \vdash \texttt{airline(ba)} \wedge \texttt{ftse100(ba)}\\
&\mathsf{representatives}(\rho_4, \Gamma, \Delta) \vdash \texttt{airline(ryanair)} \wedge \texttt{ftse100(marksAndSpencer)}
\end{aligned}$$

It follows that

$$\begin{aligned}
&\mathsf{fset}(\epsilon_1, \Pi, \Gamma, \Delta) = \{\rho_1, \rho_3, \rho_4\},\\
&\mathsf{fset}(\epsilon_2, \Pi, \Gamma, \Delta) = \{\rho_2, \rho_3, \rho_4\},\\
&\mathsf{fset}(\epsilon_3, \Pi, \Gamma, \Delta) = \{\rho_3, \rho_4\},\\
&\mathsf{fset}(\epsilon_4, \Pi, \Gamma, \Delta) = \{\rho_3\},
\end{aligned}$$

For any expectation with a disjunctive antecedent, the fset for that antecedent is the union of the fsets of the single literals that are the disjuncts in that antecedent:

Proposition 4.3.2. *Let* ϵ *be an expectation,* $\mathsf{antecedent}(\epsilon) = \alpha_1 \vee ... \vee \alpha_k$ *where* $\{\forall \bar{x}_1\ \alpha_1 ... \forall \bar{x}_1\ \alpha_k\} \subseteq \mathsf{quantifiedLiterals}(M)$, $\bar{x} = \bar{x}_1 \oplus ... \oplus \bar{x}_k$ *and* $\mathsf{consequent}(\epsilon) = \beta$. *It then follows that*

$$\mathsf{fset}(\epsilon, \Pi, \Gamma, \Delta) = \mathsf{fset}(\forall \bar{x}_1\ \alpha_1 \rightarrow \beta, \Pi, \Gamma, \Delta) \cup ... \cup \mathsf{fset}(\forall \bar{x}_k\ \alpha_k \rightarrow \beta, \Pi, \Gamma, \Delta)$$

Proof. The reports in $\mathsf{fset}(\epsilon, \Pi, \Gamma, \Delta)$ are exactly those ρ for which there is a grounding set Φ such that $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{antecedent}(\mathsf{ground}(\epsilon, \Phi))$ which is exactly that set of those ρ for which $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{ground}(\forall \bar{x}_1\ \alpha_1, \Phi) \vee ... \vee \mathsf{ground}(\forall \bar{x}_k\ \alpha_k, \Phi)$. Thus $\mathsf{fset}(\epsilon, \Pi, \Gamma, \Delta) =$

$$\mathsf{fset}(\forall \bar{x}_1\ \alpha_1 \rightarrow \mathsf{consequent}(\epsilon), \Pi, \Gamma, \Delta) \cup ... \cup \mathsf{fset}(\forall \bar{x}_k\ \alpha_k \rightarrow \mathsf{consequent}(\epsilon), \Pi, \Gamma, \Delta)$$

□

Example 4.3.3. *Let* Γ *be a set of access rules and* Δ *be a set of background facts. Let* β *be an arbitrary consequent and let*

$$\epsilon_1 = \forall \mathtt{x}\ \mathtt{airline(x)} \rightarrow \beta$$
$$\epsilon_2 = \forall \mathtt{y}\ \mathtt{ftse100(y)} \rightarrow \beta$$
$$\epsilon_3 = \forall \mathtt{x,y}\ \mathtt{airline(x)} \vee \mathtt{ftse100(y)} \rightarrow \beta$$

Let $\Phi = \{\mathtt{x} = \mathtt{ba}\}$ *be a grounding set. Let* Π *be a set of reports* $\{\rho_1, \rho_2\}$ *such that*

$$\mathsf{representatives}(\rho_1, \Gamma, \Delta) \vdash \mathtt{airline(ba)},$$
$$\mathsf{representatives}(\rho_2, \Gamma, \Delta) \vdash \mathtt{ftse100(marksAndApencer)}$$

It follows that

$$\mathsf{fset}(\forall \mathtt{x}\ \mathtt{airline(x)} \rightarrow \beta, \Pi, \Gamma, \Delta) = \{\rho_1\},$$
$$\mathsf{fset}(\forall \mathtt{y}\ \mathtt{ftse100(y)} \rightarrow \beta, \Pi, \Gamma, \Delta) = \{\rho_2\},$$
$$\mathsf{fset}(\forall \mathtt{x,y}\ \mathtt{airline(x)} \vee \mathtt{ftse100(y)} \rightarrow \beta, \Pi, \Gamma, \Delta) = \{\rho_1, \rho_2\},$$
$$\mathsf{fset}(\forall \mathtt{x}\ \mathtt{airline(x)} \rightarrow \beta, \Pi, \Gamma, \Delta) \cup \mathsf{fset}(\forall \bar{\mathtt{x}}\ \mathtt{ftse100(x)} \rightarrow \beta, \Pi, \Gamma, \Delta) = \{\rho_1, \rho_2\}$$

Note that the tuples $\bar{x}_1, ..., \bar{x}_2$ are disjoint, but as only one of $\forall \mathtt{x}\ \mathtt{airline(x)}$ or $\forall \mathtt{y}\ \mathtt{ftse100(y)}$ need be fired in order to fire $\forall \mathtt{x,y}\ \mathtt{airline(x)} \vee \mathtt{ftse100(y)}$, it does not matter whether x and y are ground by the same or different constants.

For any expectation with a conjunctive consequent, the aset for that consequent is the union of the asets of the single literals that are the conjuncts in that consequent:

Proposition 4.3.3. *Let* ϵ *be an expectation and let* $\mathsf{consequent}(\epsilon) = \forall \bar{x}\ \alpha_1 \wedge ... \wedge \alpha_k$ *where* $\{\forall \bar{x}_1\ \alpha_1 ... \forall \bar{x}_1\ \alpha_k\} \subseteq \mathsf{quantifiedLiterals}(M)$ *and* $\bar{x} = \bar{x}_1 \oplus ... \oplus \bar{x}_k$ *and let* $\mathsf{antecedent}(\epsilon) =$

$\forall \bar{x}\ \beta$. *It follows that*

$$\mathsf{aset}(\epsilon, \Pi, \Gamma, \Delta) = \mathsf{aset}(\forall \bar{x}\ \beta \rightarrow \alpha_1, \Pi, \Gamma, \Delta) \cup ... \cup \mathsf{aset}(\forall \bar{x}\ \beta \rightarrow \alpha_k, \Pi, \Gamma, \Delta)$$

Proof. The reports in $\mathsf{aset}(\epsilon, \Pi, \Gamma, \Delta)$ are exactly those ρ for which there is a grounding set Φ such that $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \neg\mathsf{consequent}(\mathsf{ground}(\epsilon, \Phi))$ which is exactly that set of those ρ for which $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{ground}(\forall \bar{x}_1\ \neg\alpha_1, \Phi) \vee ... \vee \mathsf{ground}(\forall \bar{x}_k\ \neg\alpha_k, \Phi)$. Thus $\mathsf{aset}(\epsilon, \Pi, \Gamma, \Delta) =$

$$\mathsf{aset}(\mathsf{antecedent}(\epsilon) \rightarrow \alpha_1, \Pi, \Gamma, \Delta) \cup ... \cup \mathsf{aset}(\mathsf{antecedent}(\epsilon) \rightarrow \alpha_k, \Pi, \Gamma, \Delta)$$

□

Example 4.3.4. *Let* Γ *be a set of access rules and* Δ *be a set of background facts. Let* α *be an arbitrary antecedent and let*

$$\epsilon_1 = \forall \mathtt{x}\ \alpha \rightarrow \mathtt{profitable(x)}$$
$$\epsilon_2 = \forall \mathtt{y}\ \alpha \rightarrow \mathtt{highSharePrice(y)}$$
$$\epsilon_3 = \forall \mathtt{x,y}\ \alpha \rightarrow \mathtt{profitable(x)} \wedge \mathtt{highSharePrice(y)}$$

Let $\Phi = \{\mathtt{x} = \mathtt{ba}\}$ *be a grounding set. Let* Π *be a set of reports* $\{\rho_1, \rho_2\}$ *such that*

$$\mathsf{representatives}(\rho_1, \Gamma, \Delta) \vdash \neg\mathtt{profitable(ba)},$$
$$\mathsf{representatives}(\rho_2, \Gamma, \Delta) \vdash \neg\mathtt{highSharePrice(ba)}$$

It follows that

$$\mathsf{aset}(\epsilon_1, \Pi, \Gamma, \Delta) = \{\rho_1\},$$
$$\mathsf{aset}(\epsilon_2, \Pi, \Gamma, \Delta)) = \{\rho_2\},$$
$$\mathsf{aset}(\epsilon_3, \Pi, \Gamma, \Delta)) = \{\rho_1, \rho_2\},$$
$$\mathsf{aset}(\epsilon_1, \Pi, \Gamma, \Delta) \cup \mathsf{aset}(\epsilon_2, \Pi, \Gamma, \Delta) = \{\rho_1, \rho_2\}$$

Note that the tuples $\bar{x}_1, ..., \bar{x}_2$ are disjoint, but as only one of $\forall \mathtt{x}\ \mathtt{profitable(x)}$ or $\forall \mathtt{y}\ \mathtt{highSharePrice(y)}$ need be attacked to attack $\forall \mathtt{x,y}\ \mathtt{profitable(x)} \wedge \mathtt{highSharePrice(y)}$, it does not matter whether x and y are ground by the same or different consequents.

For any expectation with a disjunctive consequent, the aset for that consequent is the intersection of the asets of the single literals that are the disjuncts in that consequent:

Proposition 4.3.4. *Let* ϵ *be an expectation,* $\mathsf{consequent}(\epsilon) = \forall \bar{x}\ \alpha_1 \vee ... \vee \alpha_k$ *where* $\{\forall \bar{x}_1\ \alpha_1 ... \forall \bar{x}_1\ \alpha_k\} \subseteq \mathsf{quantifiedLiterals}(M)$ *and* $\bar{x} = \bar{x}_1 \oplus ... \oplus \bar{x}_k$ *and let* $\mathsf{antecedent}(\epsilon) = \forall \bar{x}\ \beta$. *It follows that* $\mathsf{aset}(\epsilon, \Pi, \Gamma, \Delta) = \mathsf{aset}(\forall \bar{x}\ \beta \rightarrow \alpha_1, \Pi, \Gamma, \Delta) \cap ... \cap \mathsf{aset}(\forall \bar{x}\ \beta \rightarrow \alpha_k, \Pi, \Gamma, \Delta)$.

Proof. The reports in $\mathsf{aset}(\epsilon, \Pi, \Gamma, \Delta)$ are exactly those ρ for which there is a grounding set Φ such that $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \neg\mathsf{consequent}(\mathsf{ground}(\epsilon, \Phi))$ which is exactly that set of those ρ for which $\mathsf{representatives}(\rho, \Gamma, \Delta) \vdash \mathsf{ground}(\forall \bar{x}_1\ \neg\alpha_1, \Phi) \wedge ... \wedge \mathsf{ground}(\forall \bar{x}_k\ \neg\alpha_k, \Phi)$. Thus $\mathsf{aset}(\epsilon, \Pi, \Gamma, \Delta) =$

$$\mathsf{aset}(\mathsf{antecedent}(\epsilon) \rightarrow \alpha_1, \Pi, \Gamma, \Delta) \cap ... \cap \mathsf{aset}(\mathsf{antecedent}(\epsilon) \rightarrow \alpha_k, \Pi, \Gamma, \Delta)$$

□

Note that in the following proposition and proof the tuples $\bar{x}_1 ... \bar{x}_k$ are assumed to be disjoint. That is, for all $\bar{x}_i ... \bar{x}_j$ in $\bar{x}_1 ... \bar{x}_k$, if a variable is in $\bar{x}_i$ then it is not in $\bar{x}_k$. As such the disjunctive consequent $\forall \bar{x}\ \alpha_1 \vee ... \vee \alpha_k$ is ground by the same grounding sets that ground $\forall \bar{x}_1\ \alpha_1$ and...and $\forall \bar{x}_1\ \alpha_k$.

Example 4.3.5. *Let Γ be a set of access rules and Δ be a set of background facts. Let α be an arbitrary antecedent and let*

$$\epsilon_1 = \forall \mathtt{x}\ \alpha \rightarrow \mathtt{profitable(x)},$$
$$\epsilon_2 = \forall \mathtt{y}\ \alpha \rightarrow \mathtt{highSharePrice(x)}$$
$$\epsilon_3 = \forall \mathtt{x,y}\ \alpha \rightarrow \mathtt{profitable(x)} \vee \mathtt{highSharePrice(y)}$$

Let Π be a set of reports $\{\rho_1, \rho_2, \rho_3\}$ such that

$\mathsf{representatives}(\rho_1, \Gamma, \Delta) \vdash \neg\mathtt{profitable(ba)},$
$\mathsf{representatives}(\rho_2, \Gamma, \Delta) \vdash \neg\mathtt{highSharePrice(ba)},$
$\mathsf{representatives}(\rho_3, \Gamma, \Delta) \vdash \neg(\mathtt{profitable(ba)} \vee \mathtt{highshareprice(marksAndSpencer)})$

It follows that

$$\mathsf{aset}(\epsilon_1, \Pi, \Gamma, \Delta) = \{\rho_1, \rho_3\},$$
$$\mathsf{aset}(\epsilon_2, \Pi, \Gamma, \Delta) = \{\rho_2, \rho_3\},$$
$$\mathsf{aset}(\epsilon_3, , \Pi, \Gamma, \Delta) = \{\rho_3\},$$
$$\mathsf{aset}(\epsilon_1, \Pi, \Gamma, \Delta) \cap \mathsf{aset}(\epsilon_2, \Pi, \Gamma, \Delta) = \{\rho_3\}$$

In the above proposition and proof the tuples $\bar{x}_1 ... \bar{x}_k$ are assumed to be disjoint. However, there are useful consequents in which variables are repeated across the disjuncts. For example, in $\epsilon_4 = \mathsf{antecedent}(\epsilon_1) \rightarrow \mathtt{airline(x)} \vee \mathtt{ftse100(x)}$. It is the case that $\mathsf{fset}(\epsilon_4, \Pi, \Gamma, \Delta) \subseteq \mathsf{fset}(\epsilon_3, \Pi, \Gamma, \Delta)$ from the example given above. Therefore $|\mathsf{fset}(\epsilon_3, \Pi, \Gamma, \Delta)|$ is an upper bound on the size of $\mathsf{fset}(\epsilon_4, \Pi, \Gamma, \Delta)$, as the following example demonstrates:

Example 4.3.6. *Let Γ be a set of access rules and Δ be a set of background facts. Let*

$$\epsilon_1 = \forall \mathtt{x}\ \mathsf{antecedent}(\epsilon_1) \rightarrow \mathtt{airline(x)}$$
$$\epsilon_2 = \forall \mathtt{y}\ \mathsf{antecedent}(\epsilon_2) \rightarrow \mathtt{ftse100(y)}$$
$$\epsilon_3 = \forall \mathtt{x,y}\ \mathsf{antecedent}(\epsilon_2) \rightarrow \mathtt{airline(x)} \vee \mathtt{ftse100(y)}$$
$$\epsilon_4 = \forall \mathtt{x}\ \mathsf{antecedent}(\epsilon_2) \rightarrow \mathtt{airline(x)} \vee \mathtt{ftse100(x)}$$

Let Π be a set of reports $\{\rho_1, \rho_2, \rho_3, \rho_4\}$ such that

$$\mathsf{representatives}(\rho_1, \Gamma, \Delta) \vdash \neg\mathtt{airline(ba)},$$
$$\mathsf{representatives}(\rho_2, \Gamma, \Delta) \vdash \neg\mathtt{ftse100(ba)},$$
$$\mathsf{representatives}(\rho_3, \Gamma, \Delta) \vdash \neg\mathtt{airline(ba)} \wedge \neg\mathtt{ftse100(ba)}$$
$$\mathsf{representatives}(\rho_4, \Gamma, \Delta) \vdash \neg\mathtt{airline(ryanair)} \wedge \neg\mathtt{ftse100(marksAndSpencer)}$$

It follows that

$$\mathsf{aset}(\epsilon_1, \Pi, \Gamma, \Delta) = \{\rho_1, \rho_3, \rho_4\},$$
$$\mathsf{aset}(\epsilon_2, \Pi, \Gamma, \Delta) = \{\rho_2, \rho_3, \rho_4\},$$
$$\mathsf{aset}(\epsilon_3, \Pi, \Gamma, \Delta) = \{\rho_3, \rho_4\},$$
$$\mathsf{aset}(\epsilon_4, \Pi, \Gamma, \Delta) = \{\rho_3\},$$

It is therefore possible to derive the upper bound on the fset and aset of any formula in $\mathsf{conjunctiveFormulae}(M)$ or $\mathsf{disjunctiveFormulae}(M)$ as long as the necessary fsets and asets for the formulae in $\mathsf{quantifiedLiterals}(M)$ are known. This significantly reduces the amount of information which needs to be stored. This information can then be used to generate a set of expectations, as demonstrated in Chapter 5.

4.4 Conclusion to Chapter 4

A set of marker formulae is used to represent the set of all antecedents and the set of all consequents. For any pair of expectations $\epsilon_1, \epsilon_2 \in E$, ϵ_1 may have a logically weaker antecedent or consequent, a logically stronger antecedent or consequent, both or neither.

Knowing which of these two expectations has the stronger antecedent and or consequent can also enable us to deduce which will have the greater fired, accuracy and confirmation values. Section 4.2 enumerates the deductions that can be drawn regarding the order of values from the logical order of antecedents and consequents. These deductions allow us to draw some useful conclusions: firstly, that there are some expectations whose values are fixed; secondly that the unknown values for expectations can be approximated from known values from other, logically related expectations and finally; that information need only be kept on some members of M (namely the single literals), in order to be able to deduce the values for all expectations in E.

A summary of the findings in this chapter is as follows:

- The consequence relation between the antecedents of two expectations determines the order of the fired value of those expectations (Section 4.2.1).
- The consequence relation between the consequents of two expectations determines the order of the attacked and supported values for those expectations (Section 4.2.2).
- There are special case expectations whose values are determined by their syntax (Section 4.2.4).
- Bounds on the fsets, ssets and asets of conjunctive and disjunctive formulae can be derived from the fsets, ssets and asets for quantified literals (Section 4.3).

The next chapter will demonstrate how this knowledge that arises from the ordering of expectations can be further harnessed to create a subset of the set of expectations, the set of working expectations, that will be used to analyse interesting news reports.

Chapter 5

Generating working expectations

Chapter 2 introduced the Expectation Violation Analysis (EVA) Framework. Core to this framework is a set of expectations, facts from news reports and a set of background knowledge. Chapter 3 demonstrated that the relationship between a report, relevant background knowledge and an expectation can be used to identify interesting information and that expectations of differing strengths can be used to rank the unexpectedness of information. Chapter 4 demonstrated that it is possible to predict the relative strengths of expectations from their implication order.

However, the set of expectations is very large even for relatively small languages. As a consequence it is unfeasible to consider every expectation with respect to each report in the search for interesting information. It is necessary to identify a subset of the set of expectations that is small enough to be searched. This set is known as the set of working expectations. This chapter presents a novel approach for constructing such expectations based on the evidence in a set of representative sets.

There are several ways a set of working expectations can be obtained: firstly, by having a set of expectations that is generated by a domain expert; secondly, by using a machine learning technique or thirdly, by exploiting the ordering properties of the set of expectations to identify the set of working expectations. The first of these methods, using the skills of an expert together with those of a knowledge engineer, has several drawbacks: the process is time consuming, relies on the subjective judgment of one or more individuals and does not lend itself to automatic updating. The second approach is more feasible: there may be machine learning methods suitable for generating the set of working expectations. However, the third option has the advantage that, by exploiting the inherent properties of the set of expectations, it is possible to learn more about expectations such as: what are the qualities that make a "good" expectation, and how such expectations are distributed though the set of expectations.

This chapter is structured as follows: Section 5.1 defines the set of working expectations and

Section 5.2 defines algorithms that generate the set of working expectations. Finally Section 5.3 concludes this chapter with an analysis of the strengths and weaknesses of this approach to generating the set of expectations that need be considered when identifying interesting news.

5.1 The set of working expectations

This section defines the set of working expectations, a subset of the set of expectations. The set of working expectations must have the following properties:

1. The set of working expectations is small enough to search on receipt of each report
2. Each member of the set of working expectations has a coverage value that is greater than a given threshold
3. Each member of the set of working expectations has an accuracy value that is greater than a given threshold

The first condition ensures that the set of working expectations can be used to identify interesting news in real time. Conditions two and three ensure that the expectations chosen are representative of reported events in the real world. The set of working expectations is generated bottom up from the literals in the language, rather than by removing expectations from the expectation set.

The set coveredAntecedents$(M, \Pi, \Gamma, \Delta, minCoverage)$ is the set of all antecedents with fsets above a given threshold. This threshold is chosen either in response to the need for a given level of accuracy or coverage in a domain or else in response to the need to restrict the size of the set of working expectations.

Definition 5.1.1. *Let Π be a set of news reports, Γ be a set of access rules, Δ be a set of background facts and M be a set of formulae. Let β be an arbitrary consequent. Let $minCoverage$ be the lower threshold on the size of the* fset *of an antecedent. The set of antecedents whose coverage is larger than $minCoverage$, denoted is as follows:*

$$\begin{aligned}&\mathsf{coveredAntecedents}(M, \Pi, \Gamma, \Delta, minCoverage) = \\ &\qquad \{\alpha | \alpha \in M \textit{ and} \\ &\qquad \mathsf{fset}(\alpha \rightarrow \beta, \Pi, \Gamma, \Delta) \geq minCoverage\}\end{aligned}$$

Example 5.1.1. *Let M be a set of universally quantified, unground literals. For the purposes of this example let these be zero-place literals. Let M be the marker formulae that can be derived from the symbols α, β and γ and the symbols $\wedge$, $\vee$ and $\neg$.*

Let the following be the fset*s for the literals in M :*

$$
\begin{aligned}
\mathsf{fset}(\alpha) &= \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5\} \\
\mathsf{fset}(\neg\alpha) &= \{\rho_6, \rho_7, \rho_8, \rho_9, \rho_{10}\} \\
\mathsf{fset}(\beta) &= \{\rho_1, \rho_2, \rho_5, \rho_6\} \\
\mathsf{fset}(\neg\beta) &= \{\rho_3, \rho_4, \rho_7, \rho_8, \rho_9, \rho_{10}\} \\
\mathsf{fset}(\gamma) &= \{\rho_1, \rho_2\} \\
\mathsf{fset}(\neg\gamma) &= \{\rho_3, \rho_4, \rho_5, \rho_6, \rho_7, \rho_8, \rho_9, \rho_{10}\}
\end{aligned}
$$

$\mathsf{coveredAntecedents}(M, \Pi, \Gamma, \Delta, 2) =$

α	$\neg\alpha$	β	$\neg\beta$	γ	$\neg\gamma$
$\alpha \wedge \beta$	$\alpha \wedge \neg\beta$	$\alpha \wedge \gamma$	$\alpha \wedge \neg\gamma$	$\beta \wedge \gamma$	$\beta \wedge \neg\gamma$
$\alpha \wedge \beta \wedge \gamma$	$\alpha \wedge \beta \wedge \neg\gamma$	$\alpha \wedge \neg\beta \wedge \neg\gamma$	$\neg\alpha \wedge \beta \wedge \neg\gamma$	$\neg\alpha \wedge \neg\beta \wedge \neg\gamma$	

It is possible to exploit the property that for all ϵ_1, $\epsilon_2 \in E$, such that $\mathsf{antecedent}(\epsilon_1) \preceq \mathsf{antecedent}(\epsilon_2)$, $\mathsf{fset}(\epsilon_1, \Pi, \Gamma, \Delta) \leq \mathsf{fset}(\epsilon_2, \Pi, \Gamma, \Delta)$. We can use this knowledge to reduce the number of fsets that need to be calculated: if $|\mathsf{fset}(\epsilon_2, \Pi, \Gamma, \Delta)|$ is less than the threshold $minCoverage$ then it is not necessary to generate the set $\mathsf{fset}(\epsilon_1, \Pi, \Gamma, \Delta)$ as $|\mathsf{fset}(\epsilon_1, \Pi, \Gamma, \Delta)| \leq \mathsf{fset}(\epsilon_2, \Pi, \Gamma, \Delta)$.

Example 5.1.2. *Using the figures from Example 5.1.1, there are five two conjunct antecedents that have* fsets *that are smaller than the threshold size $minCoverage$. Three of these are logically inconsistent (and therefore non-firing) antecedents $\alpha \wedge \neg\alpha$, $\beta \wedge \neg\beta$ and $\gamma \wedge \neg\gamma$. The other two are $\neg\alpha \wedge \gamma$ and $\neg\beta \wedge \gamma$. It is therefore not necessary to generate any antecedents that are higher in the consequence order than any of these five.*

If the fsets for each of the possible three conjunct antecedents for this language were to be generated, it would be necessary to create and evaluate 20 fsets. However, from the fsets from single literals it is predictable that for 15 of the 20 antecedents will have an fset smaller than the minimum threshold size, as the fsets for these antecedents will be the intersections of fsets that are themselves smaller than the threshold. Therefore it is only necessary to evaluate five from 20 antecedents, a saving of 75% of the fset calculations which would be performed otherwise.

In the worst case scenario, no fset is smaller than the threshold and so all fsetsmustbecalculated. The best case scenario is that a number of "branches" of the space of antecedents are ruled out early on in the search. Further work is needed to calculate what the average savings are that we can expect from this approach.

The set $\mathsf{goodExpectations}(M, \Pi, \Gamma, \Delta, minCoverage)$ contains all those expectations whose antecedents have an fset value greater than $minCoverage$ and whose consequent is not a conjunct in the antecedent.

Definition 5.1.2. *Let M be a set of unground, universally quantified literals, Π be a set of reports, Γ be a set of access rules and Δ be a set of background facts. Let $minCoverage$ be*

a threshold value, and E be the set of expectations. The set of expectations such that the antecedent of each expectation is in the set coveredAntecedents$(M, \Pi, \Gamma, \Delta, minCoverage)$ *and the consequent of each expectation is a single literal, not appearing in the antecedent is denoted* goodExpectations$(M, \Pi, \Gamma, \Delta, minCoverage)$ *and is as follows:*

$$\begin{aligned}&\text{goodExpectations}(M, \Pi, \Gamma, \Delta, minCoverage) = \\ &\quad \{\forall \bar{x}\bar{y}\ \alpha \rightarrow \beta | \forall \bar{x}\alpha \in \text{coveredAntecedents}(M, \Pi, \Gamma, \Delta, minCoverage)\ and \\ &\qquad \forall \bar{y}\beta \in M\ and \\ &\qquad \forall \bar{x}\alpha \not\vdash \forall \bar{y}\beta\ and \\ &\qquad \forall \bar{x}\alpha \not\vdash \forall \bar{y}\neg\beta\}\end{aligned}$$

Where $\bar{y}$ is a subtuple of $\bar{x}$.

The initial condition ensures that all expectations in goodExpectations$(M, \Pi, \Gamma, \Delta, minCoverage)$ will have coverage that is greater than or equal to $minCoverage$. The second condition ensures that all consequents in goodExpectations$(M, \Pi, \Gamma, \Delta, minCoverage)$ are single unground literals and that these are universally quantified outermost in the expectation. Condition three prevents the inclusion of self-reinforcing expectations (Definition 4.2.6) in goodExpectations$(M, \Pi, \Gamma, \Delta, minCoverage)$ and condition four prevents the inclusion of self-defeating expectations (Definition 4.2.5) in goodExpectations$(M, \Pi, \Gamma, \Delta, minCoverage)$.

Given M, Π, Γ, Δ, $maxCon$, $minCoverage$ and $minAccuracy$, workingExpectations$(M, \Pi, \Gamma, \Delta, minCoverage, minAccuracy)$ returns the working set of expectations. Each expectation in that set is a member of the set returned by goodExpectations$(M, \Pi, \Gamma, \Delta, minCoverage)$ and has an accuracy value of at least $minAccuracy$.

Definition 5.1.3. *Let $minAccuracy$ be a lower threshold on accuracy values, M be a set of unground universally quantified literals, Π be a set of reports, Γ be a set of access rules and Δ be a set of background knowledge. Let $minCoverage$ be a lower threshold on the size of the* fset *for an antecedent. The working set of expectations, denoted* workingExpectations$(M, \Pi, \Gamma, \Delta, minCoverage, minAccuracy)$ *is the subset of* goodExpectations$(M, \Pi, \Gamma, \Delta, minCoverage)$ *such that each expectation has accuracy greater than $minAccuracy$.*

$$\begin{aligned}&\text{workingExpectations}(M, \Pi, \Gamma, \Delta, minCoverage, minAccuracy) = \\ &\quad \{\epsilon | \epsilon \in \text{goodExpectations}(M, \Pi, \Gamma, \Delta, minCoverage) \\ &\qquad and\ \text{accuracy}(\epsilon, \Pi, \Gamma, \Delta) \geq minAccuracy\}\end{aligned}$$

The condition that, for each expectation ϵ in the set returned by workingExpectations, accuracy$(\epsilon) \geq minAccuracy$ ensures that the accuracy of all expectations in the working set is greater than $minAccuracy$.

Recall the necessary properties of the set of working expectations were defined such that the set should be small enough to search and contain only those expectations that have sufficiently high coverage and accuracy values. The set returned by workingExpectations contains only those expectations whose coverage is greater than $minCoverage$ and whose accuracy is greater than $minAccuracy$. It then follows that these values can be set at such a level as to restrict the size of the set of working expectations to one that is small enough to be searched for violations.

Example 5.1.3. *Let the set of unground literals in a language be*

$$M = \{\forall x\ \alpha(x), \forall x\ \neg\alpha(x), \beta(x), \forall x\ \neg\beta(x), \forall x\ \gamma(x), \forall x\ \neg\gamma(x)\}$$

Let the reduced set of antecedents, coveredAntecedents$(M, \Pi, \Gamma, \Delta, minCoverage) =$

$\{\forall x\ \alpha(x)$	$\forall x\ \neg\alpha(x)$	$\forall x\ \beta(x)$
$\forall x\ \neg\beta(x)$	$\forall x\ \gamma(x)$	$\forall x\ \neg\gamma(x)$
$\forall x\ \beta(x) \wedge \alpha(x)$	$\forall x\ \beta(x) \wedge \neg\alpha(x)$	$\forall x\ \beta(x) \wedge \gamma(x)$
$\forall x\ \beta(x) \wedge \neg\gamma(x)$	$\forall x\ \neg\beta(x) \wedge \alpha(x)$	$\forall x\ \beta(x) \wedge \alpha(x) \wedge \gamma(x)$
$\forall x\ \neg\beta(x) \wedge \neg\gamma(x)$	$\forall x\ \beta(x) \wedge \neg\alpha(x) \wedge \neg\gamma(x)$	$\forall x\ \alpha(x) \wedge \neg\gamma(x)$
$\forall x\ \neg\beta(x) \wedge \neg\alpha(x) \wedge \neg\gamma(x)$	$\forall x\ \neg\beta(x) \wedge \neg\alpha(x)$	$\forall x\ \beta(x) \wedge \alpha(x) \wedge \neg\gamma(x)$
$\forall x\ \alpha(x) \wedge \gamma(x)$	$\forall x\ \neg\beta(x) \wedge \alpha(x) \wedge \neg\gamma(x)$	$\forall x\ \neg\alpha(x) \wedge \neg\gamma(x)\ \}$

The set goodExpectations$(M, \Pi, \Gamma, \Delta, minCoverage)$ *and their related values are shown in Table 5.1.3 on page 105. If* $minAccuracy = 0.8$ *and* $minCoverage = 4$ *there are six expectations in Table 5.1.3 that would be in the set returned by* workingExpectations*:*

workingExpectations$(M, \Pi, \Gamma, \Delta, minCoverage, minAccuracy) =$

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

Thus, the working set of expectations is reduced to 14% *of the size of* E *(44 expectations).*

The set workingExpectations(M, Π, Γ, Δ, minCoverage, minAccuracy) is a declarative definition of the expectations that will be searched on receipt of a report. The next section presents algorithms that will generate this set.

5.2 Algorithms for the generation of working expectations

This section presents algorithms to generate the set of working expectations. Section 5.2.1 presents some simple data structures, based on arrays, for representing the data required for the algorithms.

Section 5.2.2 presents some subsidiary algorithms. The main algorithm is presented in the Section 5.2.3.

The algorithm that generates the set of working expectations, `makeWorkingExpectations`, returns an array of expectations that is equivalent to the set of working expectations defined in the previous section. The input to `makeWorkingExpectations` is an array of literals; an array of representative sets; an integer value representing the maximum conjunction size of antecedents; an integer value representing the minimum fset size of an expectation in the set of working expectations; and a floating point value in the range $[0, 1]$ which represents the minimum accuracy of an expectation included in the set of working expectations. The main algorithm calls the functions `degrounding`, `makecoveredAntecedents` and `accuracy` shown in Section 5.2.2.

The `degrounding` function takes a ground formula and returns a deground version. The `accuracy` function receives as input an expectation and a set of reports and returns the accuracy value for that expectation. The `makecoveredAntecedents` function takes as input an array of literals, an array of representative sets, an integer value representing the maximum conjunction size of antecedents and an integer value representing the minimum fset size of an expectation in the set working expectations and returns the array of antecedents whose coverage is greater than or equal to the value `coverage` and whose length is less than or equal to `maxCon`. The data structures used in these algorithms are as shown in the next section.

5.2.1 Data structures

For the sake of simplicity, all data structures are based on arrays. The following data structures are used in the algorithms given in this chapter, but does not preclude the use of other structures, objects for example, in an implementation.

- A **predicate symbol** is represented by a string and for the purposes of these examples will be one of $\texttt{a|b|c}_{\texttt{index}}$, where `index` is an integer greater than zero. A negated predicate is a string that, for the purposes of the following examples will be one of `nota|notb|notc` or else $\texttt{notc}_{\texttt{index}}$.
- A **constant** is represented by a string that, for the purposes of the following examples will be one of `m|n|o` or else $\texttt{m}_{\texttt{index}}$ where $\texttt{index} \geq 1$.
- A **variable** is represented by a string that, for the purposes of the following examples will be one of `x|y|z` or else $\texttt{x}_{\texttt{index}}$.
- A **degrounding symbol** is a unique identifier, $\texttt{v}_{\texttt{index}}$, where $\texttt{index} \geq 1$, that replaces all instances of a constant in a formula thus allowing the identification of formulae which, when deground, are logically equivalent. The next section looks at the process of degrounding in detail.

- A **literal** is represented by an array of variable length $[1, ..., n]$ such that position 1 holds a predicate symbol that may or may not be negated and positions $2, ..., n$ hold either a constant, a variable or a degrounding symbol. Universal quantification is assumed over all variables.
- A **conjunction** is represented by a variable length array of literals $[[literal_1], ..., [literal_n]]$, where $n \geq 1$ such that each literal is a conjunct. Variables are assumed to be universally quantified outermost.
- An **expectation** is represented by a two place array [[conjunction][literal]] where [conjunction] is an antecedent and [literal] is the consequent. Quantification of variables is assumed to be universal and outermost.
- A **representative set** is a variable length array of ground literals, representing the information extracted from a report and the associated information from the set of background facts.
- The array **representative sets** is a variable length array containing one representative set for each report received by the system.
- An **aset**, a **vset** and an **fset** are all variable length arrays $[index_1, ..., index_n]$ where $n \geq 1$ and $index_1, ..., index_n$ are indices of the representative sets array.

Examples of some of the above data structures are presented below:

Example 5.2.1. [b,m] *is a ground, monadic literal equivalent to the literal* b(m). [a, x, y] *is an unground, dyadic literal, representing the literal* $\forall x, y\ a(x, y)$. [nota, x, y] *an unground, negated dyadic literal, representing the literal* $\forall x, y\ \neg a(x, y)$

Example 5.2.2. [[a, x, y][b, x]] *is a conjunction equivalent to the formula* $\forall x, y\ a(x, y) \wedge b(x)$.

Example 5.2.3. [[[a, x, y], [b, x]], [c, y]] *is equivalent to the expectation* $\forall x, y\ a(x, y) \wedge b(x) \rightarrow c(y)$.

Example 5.2.4. [[a, m, n], [b, n], [c, m]] *is a representative set, equivalent to the set* $\{a(m, n), b(n), c(m)\}$.

Example 5.2.5. *Let* repSet *be an array of representative sets:*

$$
\begin{aligned}
&repSet[1] = [[a, m, n], [b, m], [notc, n]] \\
&repSet[2] = [[a, n, o], [b, n]] \\
&repSet[3] = [[notb, n], [c, o]] \\
&repSet[4] = [[b, m], [notc, o]] \\
&repSet[...] = ... \\
&repSet[j] = [[a, m, n], [b, n], [c, m]]
\end{aligned}
$$

Example 5.2.6. *Let* [[[a, x, y], [b, x]][c, y]] *be an expectation. Let* repSet *be the array of representative sets.*

$$\begin{aligned}
\texttt{repSet[1]} &= [[\texttt{a},\texttt{m},\texttt{n}],[\texttt{b},\texttt{m}],[\texttt{notc},\texttt{n}]] \\
\texttt{repSet[2]} &= [[\texttt{a},\texttt{n},\texttt{o}],[\texttt{b},\texttt{n}]] \\
\texttt{repSet[3]} &= [[\texttt{notb},\texttt{n}],[\texttt{c},\texttt{o}]] \\
\texttt{repSet[4]} &= [[\texttt{b},\texttt{m}],[\texttt{notc},\texttt{o}]] \\
\texttt{repSet[5]} &= [[\texttt{a},\texttt{m},\texttt{n}],[\texttt{b},\texttt{n}],[\texttt{c},\texttt{m}]]
\end{aligned}$$

The fset *for the antecedent* [[a, x, y], [b, x]] *is* [1, 2]. *The* aset *for the consequent* [c, y] *is* [1, 4]. *The* vset *for the expectation* [[[a, x, y], [b, x]][c, y]] *is* [1].

5.2.2 Functions used by the working expectation generator

To determine whether an expectation should be a member of the set of working expectations it is necessary to determine the coverage and accuracy values for that expectation. This is achieved by creating the vset and fset for the expectation and its antecedent respectively. It is necessary to create a 'deground' version of each (ground) representative set in order to identify matches with (unground) expectations. Degrounding is achieved by replacing each constant with a degrounding symbol. For each pair of constants c_1, c_2 in the representative set and a symbol v, if $c_1 = c_2$ and the degrounding symbol that replaces $c_1 = v$ then the symbol that replaces $c_2 = v$.

Example 5.2.7. *Let* a *and* b *be predicate symbols. Let* m, n, o *be constants. Let* $\texttt{a}(\texttt{m},\texttt{n},\texttt{o}) \wedge \texttt{b}(\texttt{m})$, $\texttt{a}(\texttt{o},\texttt{n},\texttt{m}) \wedge \texttt{b}(\texttt{o})$ *and* $\texttt{a}(\texttt{m},\texttt{n},\texttt{o}) \wedge \texttt{b}(\texttt{o})$, *be formulae. Let the degrounding symbol for the first constant in each formula be* $\texttt{v}_1$, *the degrounding symbol for the second constant in each formula be* $\texttt{v}_2$ *and the degrounding symbol for the third constant in each formula be* $\texttt{v}_3$. *Replacing the constants in the formula* $\texttt{a}(\texttt{m},\texttt{n},\texttt{o}) \wedge \texttt{b}(\texttt{m})$ *with the degrounding symbols results in the formula* $\texttt{a}(\texttt{v}_1,\texttt{v}_2,\texttt{v}_3) \wedge \texttt{b}(\texttt{v}_1)$. *Replacing the constants in the formula* $\texttt{a}(\texttt{o},\texttt{n},\texttt{m}) \wedge \texttt{b}(\texttt{o})$ *with the degrounding symbols results in the formula* $\texttt{a}(\texttt{v}_1,\texttt{v}_2,\texttt{v}_3) \wedge \texttt{b}(\texttt{v}_1)$. *Replacing the constants in the formula* $\texttt{a}(\texttt{m},\texttt{n},\texttt{o}) \wedge \texttt{b}(\texttt{o})$ *with the degrounding symbols results in the formula* $\texttt{a}(\texttt{v}_1,\texttt{v}_2,\texttt{v}_3) \wedge \texttt{b}(\texttt{v}_3)$. *Therefore it is evident that the first and second formulae are ground versions of the same unground formula, whereas the third formula is a ground version of a different formula.*

In order to create the array of degrounding symbols, it is first necessary to extract all the constants from the literals in the representative set, in order to determine the order in which each constant first appears.

Definition 5.2.1. *Let* repSet *be a representative set.* extractConstants(repSet) *returns an*

array of **all** *constant symbols in* `repSet`.

```
extractConstants(repSet)
 1 constants[]
 2 currentLitteral[]
 3 counter ← 0
 3 for i in 1 to length(repSet)
 5       currentLiteral ← repSet[i]
 6       for j in 2 to length(currentLiteral)
 7             counter ← counter + 1
 8             constants[counter] ← currentLiteral[j]
 9 return constants
```

Example 5.2.8. `extractConstants([[a,m,n,o],[b,m,n]])` *returns* `[m,n,o,m,n]`.

Once the constants have been extracted from `repSet` it is necessary to identify the order in which each constant first appears as this will determine the index of the degrounding symbol that will replace each instance of that constant in the representative set. The function `makeDegroundingSet` takes the array of constants returned by `extractConstants` and returns an array in which each constant appears only once, in the order in which it first occurs in the representative set.

Definition 5.2.2. `makeDegroundingSet(constants)` *creates an array of the unique constant names in the array* `constants`. *Line* 4 *checks to see that a constant is not already present in* `degroundingSet` *before adding it to that set:*

```
makeDegroundingSet(constants)
 1 degroundings ← []
 2 varcount ← 1
 3 for i in 1 to length(constants)
 4       if constants[i] not in degroundings
 5             degroundings[varcount] ← constants[i]
 6             varcount ← varcount + 1
 7 return degroundings
```

Example 5.2.9. `makeDegroundings([m,n,o,m,n])` $=$ `[m,n,o]`.

The function `subRepSet` accepts a representative set and a set of literals and returns a subset of the representative set that contains only those literals with the predicate symbols that appear in the set `literals`.

Definition 5.2.3. *Let* `literals` *be a set of literals and* `repSet` *be a representative set.*

```
subRepSet(repSet, literals)
 1 repCount ← 0
 2 reps ← []
 3 currentLiteral ← []
 4 currentRep ← []
 5 for i in 1 to length(literals)
 6       currentLiteral ← literals[i]
 7       for j in 1 to length(repSet)
 8             currentRep ← repSet[j]
 9                if currentLiteral[1] = currentRep[1] and
length(currentLiteral) = length(currentRep)
10                      repCount ← repcount + 1
11                      reps[repCount] ← currentRep
12 return reps
```

Example 5.2.10. *Let*

$$\texttt{repSet} = [\texttt{a}[\texttt{m}, \texttt{n}, \texttt{o}], \texttt{b}[\texttt{m}, \texttt{o}]]$$

and let

$$\texttt{literals} = [[\texttt{b}, \texttt{x}, \texttt{y}]]$$

$\texttt{subRepSet}([\texttt{a}[\texttt{m}, \texttt{n}, \texttt{o}], \texttt{b}[\texttt{m}, \texttt{o}]], [[\texttt{b}, \texttt{x}, \texttt{y}]])$ *returns* $[[\texttt{b}, \texttt{m}, \texttt{o}]]$

Once the array of degrounding symbols is returned by `makeDegroundingSets`, it is possible to deground the representative set. This is done by replacing each instance of a constant with a degrounding symbol. The degrounding symbol used will be the one with the index that matches the position of that constant in the array of degrounding symbols.

Definition 5.2.4. *Let* `repSets` *be an array of representative sets and* `literals` *an array of literals.* `deground(repSets, literals)` *converts each constant in a representative set to a de-*

grounding symbol. A degrounding symbol is of the form $\mathtt{v_{index}}$.

```
deground(repSets, literals)
 1 for i in 1 to length(repSets)
 2       currentRepset ← subRepSets(repSets[i], literals)
 3       constSet ← extractConstants(currentRepset)
 4       degroundingSet ← makeDegroundingSet(constSet)
 5       for j in 1 to length(currentRepset)
 6            currentLiteral ← currentRepset[j]
 7                       for k in 2 to length(currentLiteral)
 8                       for m in 1 to length(degroundingSet)
 9                            if degroundingSet[m] = currentLiteral[k]
10                                 currentLiteral[k] ← "v_m"
11             currentRepset[j] ← currentLiteral
12       repSets[i] ← currentRepset
13 return repSets
```

Example 5.2.11. *Let* repSet =

$$[[[a, m, n, o], [b, m, n]], [[a, o, m, n], [b, n, m]], [[a, m, n, o], [b, m, o]]]$$

At line 2, subRepSet(repSets[1], [[a, x, y, z], [b, x, y]]) = [[a, m, n, o], [b, m, n]], subRepSet(repSets[2], [[a, x, y, z], [b, x, y]]) = [[a, o, m, n], [b, n, m]] *and* subRepSet(repSets[3], [[a, x, y, z], [b, x, y]]) = [[a, m, n, o], [b, m, o]].

At line 4, when i = 1, degroundingSet = [m, n, o]. *when* i = 2, degroundingSet = [o, m, n]. *When* i = 3, degroundingSet = [m, n, o].

So deground(repSets, [[a, x, y, z], [b, x, y]]) *returns*

$$[[[a, v_1, v_2, v_3], [b, v_1, v_2]], [[a, v_1, v_2, v_3], [b, v_3, v_2]], [[a, v_1, v_2, v_3], [b, v_1, v_3]]]$$

Representative sets must be deground, that is, their constants converted to placeholders, in order to create unground formulae that will make up the expectations in the working set. It is important to preserve the "place" of each constant however, hence the need for degrounding symbols. An example will illustrate this:

Example 5.2.12. *Let there be three representative sets:*

Repset 1 : {takeover(ryanair, buzz), profitable(ryanair)}
Repset 2 : {takeover(boots, unichem), profitable(unichem)}
Repset 3 : {takeover(safeway, morrisons), profitable(safeway)}

Repsets 1 and 3 deground to $\{\texttt{takeover}(v_1, v_2), \texttt{profitable}(v_1)\}$ *(a takeover in which the bidder is a profitable party), whilst repset 2 degrounds to* $\{\texttt{takeover}(v_1, v_2), \texttt{profitable}(v_2)\}$ *(a takeover in which the target is the profitable party). These would result in the generation of quite different expectations.*

Definition 5.2.5. *Let* `degroundRepSets` *be an array of deground representative sets. The order of degrounding symbols in a representative set is analogous to the order of variables in an expectation. Each unique order of degrounding symbols must be identified.*

```
group(degroundRepSets)
 1 uniqueSetCount ← 0
 2 uniqueDegroundRepSets ← []
 3 for i in 1 to length(degroundrepSets)
 4       if degroundRepSets[i] is not in uniqueDegroundRepSets
 5             uniqueSetCount ← uniqueSetCount + 1
 6             uniqueDegroundRepSets[uniqueSetCount] ← degroundRepSets[i]
 7 return uniqueDegroundRepSets
```

Example 5.2.13. *Let* `degroundRepSets` *be an array of deground representative sets.*

$$\texttt{degroundRepSets}[1] = [[\texttt{a}, v_1, v_2, v_3], [\texttt{b}, v_1, v_2]]$$
$$\texttt{degroundRepSets}[2] = [[\texttt{a}, v_1, v_2, v_3], [\texttt{b}, v_3, v_2]]$$
$$\texttt{degroundRepSets}[3] = [[\texttt{a}, v_1, v_2, v_3], [\texttt{b}, v_1, v_2]]$$

So the array returned by `group(deground, repSets)` *is*

$$\texttt{uniqueDegroundRepSets}[1] = [[\texttt{a}, v_1, v_2, v_3], [\texttt{b}, v_1, v_2]]$$
$$\texttt{uniqueDegroundRepSets}[2] = [[\texttt{a}, v_1, v_2, v_3], [\texttt{b}, v_3, v_2]]$$

The set of working expectations is the set of expectations that have high coverage values and high accuracy values. The first step in creating the set of working expectations is to generate the set of antecedents with a coverage value greater than some threshold, and then to create the set of all expectations from these antecedents such that the accuracy of each expectation is greater than some threshold. The algorithms `coveredAntecedents` and `makeGoodExpextations` are presented here, along with supporting functions.

The antecedents in the set of working expectations are formed from conjunctions of literals. `makeConjunctions` takes a set of literals and an integer `conSize` and returns all conjunctions of size `conSize`. The array `literals` contains one literal for each predicate symbol in the domain. The order of variables is immaterial in this function.

Definition 5.2.6. `makeConjunctions(literals, conSize)` *returns an array of all conjunctions of* `literals` *of length* `conSize`. *There are no duplicated elements in the array returned.*

Example 5.2.14. `makeConjunctions([[a,x,y,z],[b,x,y],[c,x]],2)` *returns the array:*

$$\texttt{conjunctions[1]} = [[\texttt{a},\texttt{x},\texttt{y},\texttt{z}],[\texttt{b},\texttt{x},\texttt{y}]]$$
$$\texttt{conjunctions[2]} = [[\texttt{a},\texttt{x},\texttt{y},\texttt{z}],[\texttt{c},\texttt{x}]]$$
$$\texttt{conjunctions[3]} = [[\texttt{b},\texttt{x},\texttt{y}],[\texttt{c},\texttt{x}]]$$

The function `makecoveredAntecedents` calls the functions introduced above in order to generate the set of conjunctions up to a size of `maxCon` and return those with a fset size larger than or equal to the threshold value `coverage`.

The function `makeFset(conjunction,repSets)` returns the fset for `conjunction`. A subconjunction (call it `subconjunction`) of a conjunction (call it `conjunction'`) is any conjunction such that all conjuncts that appear in `subconjunction` also appear in `conjunction'`. We exploit the property that for all `conjunction', subconjunction`, fset(`conjunction'`) $\subseteq$ fset(`subconjunction`) to reduce the number of fsets generated.

The set generated by the function `makecoveredAntecedents(literals,repSets,coverage)` is identical to the set coveredAntecedents($M, \Pi, \Gamma, \Delta, minCoverage$) defined in Definition 5.1.1 and contains all those antecedents whose coverage is greater than the given threshold, `coverage`.

Definition 5.2.7.

```
makecoveredAntecedents(literals, repSets, coverage)
 1 firedConjunctions ← []
 2 firedConjunctionsCount ← 0
 3 degroundConjunctions ← []
 4 uniqueDegroundConjunctions ← []
 5 for h in 1 to length(literals)
 6       if length(makeFset(literals[h], repSets)) ≥ coverage
 7             firedConjunctionsCount ← firedConjunctionsCount + 1
 8             firedConjunctions[firedConjunctionsCount] ← literals[h]
 9 for i in 2 to maxCon
 10       conjunctions[] ← makeConjunctions(literals, i)
 11       for j in 1 to length(conjunctions)
 12             if all subconjunctions of conjunctions[j] of size i − 1 are in
                        firedConjunctions and
                        length(makeFset(conjunctions[j], reports)) ≥ coverage
 13                  firedConjunctionsCount ← firedConjunctionsCount + 1
 14                  firedConjunctions[firedConjunctionsCount] ← conjunctions[i]
 15 coveredAntecedentsCount ← 0
 16 coveredAntecedents ← []
 17 for k in 1 to length(firedConjunctions)
 18       degroundConjunctions ←
           deground(makeFset(firedConjunctions[k], repSets), firedConjunctions[k])
 19 uniqueDegroundConjunctions ← group(degroundConjunctions)
 20 for l in 1 to length(uniqueDegroundExpectations)
 21       coveredAntecedentsCount ← coveredAntecedentsCount + 1
 22       coveredAntecedents[coveredAntecedentsCount] ←
                  uniqueDegroundExpectations[l]
 23 return coveredAntecedents
```

Example 5.2.15. *Let* literals *be an array of literals* literals = [[a, x, y, z], [b, x, y], [c, x]]. *Let* maxCon = 3 *and* minCoverage = 4. *Let the array of representative sets,* repSets =

$$\begin{aligned}
&\texttt{repSets}[1] = [[\texttt{a},\texttt{n},\texttt{o},\texttt{m}],[\texttt{b},\texttt{n},\texttt{o}]]\\
&\texttt{repSets}[3] = [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{m},\texttt{o}]]\\
&\texttt{repSets}[3] = [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{m},\texttt{n}]]\\
&\texttt{repSets}[4] = [[\texttt{a},\texttt{n},\texttt{m},\texttt{o}],[\texttt{b},\texttt{n},\texttt{m}]]\\
&\texttt{repSets}[5] = [[\texttt{b},\texttt{m},\texttt{n}],[\texttt{c},\texttt{o}]]\\
&\texttt{repSets}[6] = [[\texttt{b},\texttt{m},\texttt{o}],[\texttt{c},\texttt{o}]]\\
&\texttt{repSets}[7] = [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{m},\texttt{n}],[\texttt{c},\texttt{o}]]\\
&\texttt{repSets}[8] = [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{n},\texttt{o}],[\texttt{c},\texttt{o}]]
\end{aligned}$$

The loop that begins at line 5 *checks the size of the* fset *for each literal in* `literals`. *Those literals whose* fset*s are larger than or equal to the threshold value* `coverage` *are added to the set* `firedConjunctions`.

At line 14, `firedConjunctions` =

$$\begin{aligned}
&\texttt{firedConjunctions}[1] = [[\texttt{a}, \texttt{x}, \texttt{y}, \texttt{z}]] \\
&\texttt{firedConjunctions}[2] = [[\texttt{b}, \texttt{x}, \texttt{y}]] \\
&\texttt{firedConjunctions}[3] = [[\texttt{c}, \texttt{x}]] \\
&\texttt{firedConjunctions}[4] = [[\texttt{a}, \texttt{x}, \texttt{y}, \texttt{z}], [\texttt{b}, \texttt{x}, \texttt{y}]] \\
&\texttt{firedConjunctions}[5] = [[\texttt{b}, \texttt{x}, \texttt{y}], [\texttt{c}, \texttt{x}]]
\end{aligned}$$

At line 22, `coveredAntecedents` =

$$\begin{aligned}
&\texttt{coveredAntecedents}[1] = [[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3]] \\
&\texttt{coveredAntecedents}[2] = [[\texttt{b}, \texttt{v}_1, \texttt{v}_2]] \\
&\texttt{coveredAntecedents}[4] = [[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_1, \texttt{v}_2]] \\
&\texttt{coveredAntecedents}[5] = [[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_1, \texttt{v}_3]] \\
&\texttt{coveredAntecedents}[6] = [[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_2, \texttt{v}_3]] \\
&\texttt{coveredAntecedents}[7] = [[\texttt{b}, \texttt{v}_1, \texttt{v}_2,], [\texttt{c}, \texttt{v}_3]] \\
&\texttt{coveredAntecedents}[8] = [[\texttt{b}, \texttt{v}_1, \texttt{v}_2,], [\texttt{c}, \texttt{v}_2]]
\end{aligned}$$

The order of variables also effects the consequent of an expectation. If any literal in an expectation is an n-ary predicate where $n > 1$ then there are several possible versions of that consequent. The set of ground representative sets allows us to identify those consequents such that the variable order results in an expectation that is accurate.

First it is necessary to create the set of expectations that have sufficiently large fsets to be in the set of working expectations and then to test those expectations for accuracy. Generating the set of working expectations by first restricting by coverage and then by accuracy is more efficient than if the order were reversed (this will be shown in Section 5.3). The set `coveredAntecedents` is a set of place-dependent antecedents.

The function `makeGoodExpectations` returns a set of expectations which are of the required form and have sufficient coverage to be members of the set of working expectations. The function `makeGoodExpectations` uses `makecoveredAntecedents` to generate an array of deground antecedents. In order to do so, `makeGoodExpectations` identifies those literals that can be used as consequents to form expectations from the antecedents in `coveredAntecedents`. The set of all reports that lead to a violation of an expectation in that set is identified. These representative sets are then deground in order to generate vsets for all variable orderings over those expectations.

The vset for an expectation is the intersection of its aset and its fset. In order to deal with place

dependent version of expectations it is necessary for the `vset` function to return a vset for each place dependent versions of the expectation. `aset` is an array of report identifiers, `fset` is an array of report identifiers and `vset` is the intersection of `fset` and `aset`:

Definition 5.2.8. *Let* `aset` *be an array of report identifiers and* `fset` *be an array of report identifiers.* `makeVset(aset,fset)` *returns an array of the report identifiers common to both sets.*

```
makeVset(aset,fset)
 1 vset ← []
 2 for i in 1 to length(fset)
 3       if fset[i] in aset
 4             add fset[i] to vset
 5 return vset
```

Example 5.2.16. *Let* $\rho_1, ..., \rho_8$ *be reports. Let* $\texttt{fset} = [1,2,4,5,7,8]$. *Let* $\texttt{aset} = [1,3,6,8]$. $\texttt{vset} = [1,8]$.

The function `makeDegroundExpectations([[antecedent],[consequent]],repSets)` returns expectations generated with respect to a set of unique deground versions of representative sets.

Definition 5.2.9. *Let* `[[antecedent],[consequent]]` *be an array that contains a deground conjunction and an unground literal respectively. Let* `repSets` *be the set of representative sets. Let* `makeFset(antecedent)` *be a function that returns the reference to each representative set that fires the antecedent. Let* `makeAset(literal)` *be a function that returns a reference to each representative set that attacks that literal. Let* `vset(aset,fset)` *return an array of references that are present in both* `aset` *and* `fset`. *Let* `degroundLiteral(literal,degroundRepSet)` *return the deground version of* `literal` *that appears in* `degroundRepSet`, *for example, the function* `degroundLiteral([b,x],[[a,`$\texttt{v}_1$`,`$\texttt{v}_2$`],[b,`$\texttt{v}_1$`]])` *returns* $[\texttt{b}, \texttt{v}_1]$. *Let* `contains(element,array)` *return a boolean, true if the element is present in the array and false if it does not.*

The function `makeFset(antecedent,repSets)` *takes an antecedent and the set of representative sets and returns an array of the indices of all representative sets which lead to a given antecedent being fired. The function* `makeAset(consequent,repSets)` *takes a consequent and the set of representative sets and returns the indices of all representative sets which lead to a given consequent being attacked.*

The function `makeVset(aset,fset)` *returns the array of the indices of all representative sets which lead to a given expectation being violated, by generating the intersection of* `fset` *and* `aset`

```
makeDegroundExpectations([[antecedent], [consequent]], repSets)
 1 degroundExpectations ← []
 2 degroundExpectationCount ← 0
 3 groundRepSets ← []
 4 groundRepSetCount ← 0
 5 fset ← makeFset(antecedent, repSets)
 6 aset ← makeAset(consequent, repSets)
 7 vset ← makeVset(fset, aset)
 8 for i in 1 to length(vset)
 9       groundRepSetCount ← groundRepsSetCount + 1
 10      currentIndex ← vset[i]
 10      groundRepSets[groundRepSetCount] ← repSets[currentIndex]
 11 degroundRepSets ← deground(groundRepSets, [[antecedent], [consequent]])
 12 for j in 1 to length(deground, repSets)
 13      currentDegroundLiteral
                        ← degroundLiteral(negated(consequent), degroundRepSets[j])
 14      currentDegroundExpectation ← [[antecedent], negated([currentDegroundLiteral])
 15      if not(contains(currentDegroundExpectation, degroundExpectations))
 16           degroundExpectationCount ← degroundExpectations + 1
 17           degroundExpectations[degroundExpectationCount]
                              ← currentDegroundExpectation
 18 return degroundExpectations
```

Example 5.2.17. *Let* [[antecedent], [consequent]] = [[[a, v_1, v_2, v_3], [b, v_1, v_2]], [notc, x]]. *Let* repSets =

$$\begin{aligned}
&\texttt{repSets}[1] = [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{m},\texttt{n}],[\texttt{c},\texttt{m}]]\\
&\texttt{repSets}[2] = [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{m},\texttt{n}],[\texttt{c},\texttt{o}]]\\
&\texttt{repSets}[3] = [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{m},\texttt{o}],[\texttt{c},\texttt{m}]]\\
&\texttt{repSets}[4] = [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{m},\texttt{n}]]\\
&\texttt{repSets}[5] = [[\texttt{b},\texttt{m},\texttt{n}],[\texttt{c},\texttt{m}]]
\end{aligned}$$

At line 5, makeFset(antecedent, repSets) = [1, 2, 4]. *At line* 6, makeAset(consequent, repSets) = [1, 2, 3, 5]. *At line* 7, makeVset(fset, aset) = [1, 2]

The loop that begins on line 8 *returns the representative sets indexed by the array* vset.

$$\begin{aligned}
&\texttt{groundRepSets}[1] = [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{m},\texttt{n}],[\texttt{c},\texttt{m}]]\\
&\texttt{groundRepSets}[2] = [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{m},\texttt{n}],[\texttt{c},\texttt{o}]]
\end{aligned}$$

At line 11, degroundRepSets *becomes*

$$\texttt{degroundRepSets}[1] = [[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_2, \texttt{v}_3], [\texttt{c}, \texttt{v}_1]]$$
$$\texttt{degroundRepSets}[2] = [[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_2, \texttt{v}_3], [\texttt{c}, \texttt{v}_3]]$$

makeDegroundExpectations($[[[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_2, \texttt{v}_3]], [\texttt{notc}, \texttt{x}]]$, repsSets) *returns:*

$$\texttt{degroundExpectations}[1] = [[[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_2, \texttt{v}_3]], [[\texttt{notc}, \texttt{v}_1]]]$$
$$\texttt{degroundExpectations}[2] = [[[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_2, \texttt{v}_3]], [[\texttt{notc}, \texttt{v}_3]]]$$

The function makeGoodExpectations ensures that no literals in the antecedent, nor any of their negations are used as the consequent of an expectation. In order to do so, a simple function negated is called.

Definition 5.2.10. *The function* negated *takes a literal and, if that literal is a positive literal, then the negated version of that literal is returned. If that literal is a negative literal then the positive version of that literal is returned.*

```
negated(literal)
 1 if literal is negative
 2       return positive literal
 3 else
 4       return negative literal
```

Example 5.2.18. *Let* literal = [a, x]. negated(literal) *returns* [nota, x]. *Let* literal = [nota, x]. negated(literal) *returns* [a, x].

The function makeGoodExpectations firstly generates the set of antecedents with fsets larger than minCoverage, thus excluding from goodExpectations all those expectations with insufficient coverage. The function then generates the deground expectations that have those antecedents and returns these as the array goodExpectations.

Definition 5.2.11. *Let* literals *be the set of all literals,* repSets *be the set of representative sets for each report indexed by the report identifier. Let* maxCon *be the maximum conjunction size of an antecedent and* minCoverage *be a lower bound on an expectation's coverage value.* makeGoodExpectations(literals, repSets, minCoverage) *returns an array of expectations,* goodExpectations *such that for each expectation* ϵ coverage(ϵ) $\geq$ coverage. *At line* 7 *expectations are created by choosing a literal for the consequent such that neither that literal nor its negation are a conjunct in the antecedent.The set generated by the function* makeGoodExpectations(literals, repSets, minCoverage) *returns the set of expectations*

goodExpectations($M, \Pi, \Gamma, \Delta, minCoverage$) *defined in Definition 5.1.2.*

```
makeGoodExpectations(literals, repSets, minCoverage)
 1 expectationCount ← 0
 2 goodExpectations ← []
 3 coveredAntecedents ← makecoveredAntecedents(literals, repSets,
                                                minCoverage)
 4 for i in 1 to length(coveredAntecedents)
 5       for j in 1 to length(literals)
 6             if literals[j] is not in coveredAntecedents[i] and
                     negated(literals[j]) is not in coveredAntecedents[i]
 7                   partDegroundExpectation ← [[coveredAntecedents[i]][literals[j]]]
 8                   degroundExpectations
                             ← makeDegroundExpectations(partDegroundExpectation,
                             repSets)
 9                   for k in 1 to length(degroundExpectations)
10                         expectationCount ← expectationCount + 1
11                                   goodExpectations[expectationsCount]
                                        ← degroundExpectations[k]
12 return goodExpectations
```

Example 5.2.19. *Let* `literals` $= [[a, x, y, z], [b, x, y], [c, x]]$. *Let*

`coveredAntecedents` $= [[[[a, v_1, v_2, v_3], [b, v_1]], [[[a, v_1, v_2, v_3], [b, v_2]], [[[b, v_1], [c, v_1]]]$

`makeGoodExpectations` *returns*

$$[[[[a, v_1, v_2, v_3], [b, v_1]], [c, v_1]],$$
$$[[[a, v_1, v_2, v_3], [b, v_1]], [c, v_2]],$$
$$[[[a, v_1, v_2, v_3], [b, v_2]], [c, v_3]],$$
$$[[[a, v_1, v_2, v_3], [b, v_2]], [c, v_2]],$$
$$[[[b, v_1], [c, v_1]]], [a, v_2, v_1, v_3]]]$$

5.2.3 The process of generating the set of working expectations

The `makeWorkingExpectations` function uses the accuracy of expectations to select those that will be included in the working set. The `calculateAccuracy(repSets, antecedent, consequent)` function is responsible for calculating those accuracy values.

Definition 5.2.12. *Let* `asets` *be an array of arrays of report identifiers. Let* `fsets` *be an array of arrays of report identifiers. Let* `antecedent` *be a conjunction of literals and* `consequent` *be*

a literal. `calculateAccuracy(repSets, antecedent, consequent)` *returns the accuracy for the expectation* `antecedent` $\rightarrow$ `consequent`.

```
calculateAccuracy(repSets, antecedent, consequent)
 1 fset ← makeFset(antecedent, repSets)
 2 vset ← makeVset(aset, fset)
 3 accuracy ← 1 − length(vset)/length(fset)
 4 return accuracy
```

Example 5.2.20. *Let* `literals` *be an array of literals* [[a, x, y, z], [b, x, y], [c, x]].

Let `repSets`=

$$\begin{aligned}
\texttt{repSets}[1] &= [[\texttt{a},\texttt{n},\texttt{o},\texttt{m}],[\texttt{b},\texttt{n},\texttt{o}]]\\
\texttt{repSets}[2] &= [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{m},\texttt{o}]]\\
\texttt{repSets}[3] &= [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{m},\texttt{n}]]\\
\texttt{repSets}[4] &= [[\texttt{a},\texttt{n},\texttt{m},\texttt{o}],[\texttt{b},\texttt{n},\texttt{m}]]\\
\texttt{repSets}[5] &= [[\texttt{b},\texttt{m},\texttt{n}],[\texttt{c},\texttt{o}]]\\
\texttt{repSets}[6] &= [[\texttt{b},\texttt{m},\texttt{o}],[\texttt{c},\texttt{o}]]\\
\texttt{repSets}[7] &= [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{m},\texttt{n}],[\texttt{c},\texttt{o}]]\\
\texttt{repSets}[8] &= [[\texttt{a},\texttt{m},\texttt{n},\texttt{o}],[\texttt{b},\texttt{n},\texttt{o}],[\texttt{c},\texttt{o}]]
\end{aligned}$$

Let `coveredAntecedents`=

$$\begin{aligned}
\texttt{coveredAntecedents}[1] &= [[\texttt{a},\texttt{v}_1,\texttt{v}_2,\texttt{v}_3]]\\
\texttt{coveredAntecedents}[2] &= [[\texttt{b},\texttt{v}_1,\texttt{v}_2]]\\
\texttt{coveredAntecedents}[3] &= [[\texttt{c},\texttt{v}_1]]\\
\texttt{coveredAntecedents}[4] &= [[\texttt{a},\texttt{v}_1,\texttt{v}_2,\texttt{v}_3],[\texttt{b},\texttt{v}_1,\texttt{v}_2]]\\
\texttt{coveredAntecedents}[5] &= [[\texttt{a},\texttt{v}_1,\texttt{v}_2,\texttt{v}_3],[\texttt{b},\texttt{v}_1,\texttt{v}_3]]\\
\texttt{coveredAntecedents}[6] &= [[\texttt{a},\texttt{v}_1,\texttt{v}_2,\texttt{v}_3],[\texttt{b},\texttt{v}_1,\texttt{v}_2],[\texttt{c},\texttt{v}_3]]\\
\texttt{coveredAntecedents}[7] &= [[\texttt{a},\texttt{v}_1,\texttt{v}_2,\texttt{v}_3],[\texttt{c},\texttt{v}_3]]\\
\texttt{coveredAntecedents}[8] &= [[\texttt{b},\texttt{v}_1,\texttt{v}_2],[\texttt{c},\texttt{v}_3]]\\
\texttt{coveredAntecedents}[9] &= [[\texttt{b},\texttt{v}_1,\texttt{v}_2],[\texttt{c},\texttt{v}_2]]\\
\texttt{coveredAntecedents}[10] &= [[\texttt{a},\texttt{v}_1,\texttt{v}_2,\texttt{v}_3],[\texttt{b},\texttt{v}_1,\texttt{v}_2],[\texttt{c},\texttt{v}_3]]\\
\texttt{coveredAntecedents}[11] &= [[\texttt{a},\texttt{v}_1,\texttt{v}_2,\texttt{v}_3],[\texttt{b},\texttt{v}_2,\texttt{v}_3],[\texttt{c},\texttt{v}_3]]
\end{aligned}$$

Recall that `fset` *is an array of indices corresponding to the members of repSets that have fired*

the antecedent. The `fsets` *for these antecedents are*

$$\begin{aligned}
&\texttt{fset}([\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], \texttt{repSets}) = [1, 2, 3, 4, 7, 8] \\
&\texttt{fset}([\texttt{b}, \texttt{v}_1, \texttt{v}_2], \texttt{repSets}) = [1, 2, 3, 4, 5, 6, 7, 8] \\
&\texttt{fset}([\texttt{c}, \texttt{v}_1], \texttt{repSets}) = [5, 6, 7, 8] \\
&\texttt{fset}([[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_1, \texttt{v}_2]], \texttt{repSets}) = [1, 3, 4, 5] \\
&\texttt{fset}([[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_1, \texttt{v}_3]], \texttt{repSets}) = [2, 7, 8] \\
&\texttt{fset}([[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_1, \texttt{v}_2], [\texttt{c}, \texttt{v}_3]], \texttt{repSets}) = [7] \\
&\texttt{fset}([[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{c}, \texttt{v}_3]], \texttt{repSets}) = [7, 8] \\
&\texttt{fset}([[\texttt{b}, \texttt{v}_1, \texttt{v}_2], [\texttt{c}, \texttt{v}_3]], \texttt{repSets}) = [5] \\
&\texttt{fset}([[\texttt{b}, \texttt{v}_1, \texttt{v}_2], [\texttt{c}, \texttt{v}_2]], \texttt{repSets}) = [6] \\
&\texttt{fset}([[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_1, \texttt{v}_2], [\texttt{c}, \texttt{v}_3]], \texttt{repSets}) = [7] \\
&\texttt{fset}([[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_2, \texttt{v}_3], [\texttt{c}, \texttt{v}_3]], \texttt{repSets}) = [8]
\end{aligned}$$

Let `coveredAntecedents[i]` *be* $[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3]$, *which is the antecedent of the expectation It follows that* $[[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{c}, \texttt{v}_3]]$. $\texttt{fset}(\texttt{coveredAntecedents[i]}, \texttt{repSets}) = \{\rho_7, \rho_8\}$. *Let* `literals[j]` $= [\texttt{notb}, \texttt{x}, \texttt{y}]$. *The literal b does not appear in* $[[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{c}, \texttt{v}_3]]$ *and so* $[\texttt{notb}, \texttt{x}, \texttt{y}]$ *can be used as a consequent. Let* $\texttt{aset}(\texttt{literals[j]}, \texttt{repSets}) = \{\rho_7, \rho_8\}$ *therefore*

$$\texttt{intersection}(\texttt{fset}(\texttt{coveredAntecedents[i]}, \texttt{repSets}), \texttt{aset}(\texttt{literals[j]}, \texttt{repSets})) = \{\rho_7, \rho_8\}$$

The reports ρ_7 *and* ρ_8 *are then deground:*

$$\texttt{deground}[7] = [[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_1, \texttt{v}_2][\texttt{c}, \texttt{v}_3]]$$

$$\texttt{deground}[8] = [[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{b}, \texttt{v}_2, \texttt{v}_3][\texttt{c}, \texttt{v}_3]]$$

The deground version of $\texttt{intersection}(\texttt{coveredAntecedents[i]}, \texttt{aset}([\texttt{b}, \texttt{x}, \texttt{y}]))$ *is grouped into two place-distinct expectations. Thus there are two vsets:*

$$\texttt{vset}([[[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{c}, \texttt{v}_3]], [\texttt{notb}, \texttt{v}_1, \texttt{v}_2]]) = [7]$$

$$\texttt{vset}([[[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{c}, \texttt{v}_3]], [\texttt{notb}, \texttt{v}_2, \texttt{v}_3]]) = [8]$$

From this the accuracy of the two expectations can be calculated

$$\textit{The accuracy of } [[[\texttt{a}, \texttt{v}_1, \texttt{v}_2, \texttt{v}_3], [\texttt{c}, \texttt{v}_3]], [\texttt{notb}, \texttt{v}_1, \texttt{v}_2]] = 1 - \frac{1}{2}$$

$$\textit{The accuracy of } [[[\mathtt{a}, \mathtt{v}_1, \mathtt{v}_2, \mathtt{v}_3], [\mathtt{c}, \mathtt{v}_3]], [\mathtt{notb}, \mathtt{v}_2, \mathtt{v}_3]] = 1 - \frac{1}{2}$$

The function `makeWorkingExpectations` returns an array of expectations whose coverage and accuracy is above a given threshold. It uses the preceding algorithms to generate the set of expectations with sufficient coverage, `goodExpectations`. It then returns those expectations in the array returned by `goodExpectations` with a sufficiently high accuracy value.

Definition 5.2.13. *Let* `literals` *be the set of literals in the universe,* `repSets` *be the set of representative sets for each report indexed by the report identifier. Let* `maxCon` *be the maximum conjunction size of an antecedent and* `minCoverage` *and* `minAccuracy` *be a lower bounds on an expectation's coverage and accuracy values.*

`makeWorkingExpectations` *returns an array of expectations,* `workingExpectations` *such that for each expectation* ϵ coverage(ϵ) $\geq$ `minCoverage` *and* accuracy(ϵ) $\geq$ `minAccuracy`. *The array returned by the function*

`makeWorkingExpectations(literals, repSets, minCoverage, minAccuracy)`

is identical to the set workingExpectations($M, \Pi, \Gamma, \Delta, minCoverage, minAccuracy$) *defined in Definition 5.1.3.*

```
makeWorkingExpectations(literals, repSets, minCoverage, minAccuracy)
  1 expectationCount ← 0
  2 workingSet ← []
  3 goodExpectations ← makeGoodExpectations(literals, repSets,
                                            minCoverage)
  4 for i in 1 to length(goodExpectations)
  5       if accuracy(goodExpectations[i]) ≥ minAccuracy
  6             expectationCount ← expectationCount + 1
  7             workingExpectations[expectationCount] ← goodExpections[i]
  8 return workingExpectations
```

Example 5.2.21. *Let the set* `goodExpectations` =

$$\begin{aligned}
\mathtt{goodExpectations}[1] &= [[[\mathtt{a}, \mathtt{v}_1, \mathtt{v}_2, \mathtt{v}_3], [\mathtt{b}, \mathtt{v}_1]], [\mathtt{c}, \mathtt{v}_1]] \\
\mathtt{goodExpectations}[2] &= [[[\mathtt{a}, \mathtt{v}_1, \mathtt{v}_2, \mathtt{v}_3], [\mathtt{b}, \mathtt{v}_1]], [\mathtt{c}, \mathtt{v}_2]] \\
\mathtt{goodExpectations}[3] &= [[[\mathtt{a}, \mathtt{v}_1, \mathtt{v}_2, \mathtt{v}_3], [\mathtt{b}, \mathtt{v}_2]], [\mathtt{c}, \mathtt{v}_3]] \\
\mathtt{goodExpectations}[4] &= [[[\mathtt{a}, \mathtt{v}_1, \mathtt{v}_2, \mathtt{v}_3], [\mathtt{b}, \mathtt{v}_2]], [\mathtt{c}, \mathtt{v}_2]] \\
\mathtt{goodExpectations}[5] &= [[[\mathtt{b}, \mathtt{v}_1], [\mathtt{c}, \mathtt{v}_1]]], [\mathtt{a}, \mathtt{v}_2, \mathtt{v}_1, \mathtt{v}_3]]
\end{aligned}$$

Let `minAccuracy` = 0.8. *Let* `accuracy`$([[[\mathtt{a}, \mathtt{v}_1, \mathtt{v}_2, \mathtt{v}_3], [\mathtt{b}, \mathtt{v}_1]], [\mathtt{c}, \mathtt{v}_1]]) = 0.65$.
$[[[\mathtt{a}, \mathtt{v}_1, \mathtt{v}_2, \mathtt{v}_3], [\mathtt{b}, \mathtt{v}_1]], [\mathtt{c}, \mathtt{v}_1]]$ *is not added to the array* `workingExpectations`.

Let accuracy([[[a, v_1, v_2, v_3], [b, v_1]], [c, v_2]]) = 0.85. expectationCount = 1 *and* workingExpectations = [[[[a, v_1, v_2, v_3], [b, v_1]], [c, v_2]]].

Let the third and fourth expectations have accuracy values of 0.3 *and* 0.6 *respectively. They will not be added to* workingExpectations.

Let accuracy([[[b, v_1], [c, v_1]], [a, v_2, v_1, v_3]]) = 0.9. expectationCount = 2 *and* workingExpectations = [[[[a, v_1, v_2, v_3], [b, v_1]], [c, v_2]], [[[b, v_1], [c, v_1]]], [a, v_2, v_1, v_3]]].

When the function makeWorkingExpectations *terminates, it returns the array* [[[[a, v_1, v_2, v_3], [b, v_1]], [c, v_2]], [[[b, v_1], [c, v_1]], [a, v_2, v_1, v_3]]].

In order to determine which expectations will be members of the set workingExpectations, it is necessary to determine which expectations have an accuracy value greater than or equal to minAccuracy. The function makeWorkingExpectations returns the array of expectations in makeGoodExpectations with accuracy greater than or equal to minAccuracy. Accuracy is a function of the number of times an expectation has been violated versus the number of times it has been fired.

5.3 Conclusion to Chapter 5

The size of the set of expectations, E, makes it impossible to search the whole set for violated expectations on receipt of every report. Generating the set of working expectations is a good solution to this problem. Firstly, the approach of generating the working expectations from the set of literals results in the set of expectations that are most useful for recommending reports. Only those reports that have violated an expectation that has both a high coverage value and a high accuracy value are presented to the user. The members of the set of working expectations are exactly those expectations that are both well covered and highly accurate.

Given that the coverage of an expectation is also used to determine that expectation's accuracy, generating coverage values before generating the violated or supported values is more than generating the violated and supported values first. Note also that restricting the members of the set of working expectations to those expectations that are both highly covered and highly accurate can also ensure that the set of working expectations is restricted to a workable size. Some initial sampling of the set may be necessary to decide the correct thresholds to ensure that the set of working expectations is smaller than the maximum number of expectations that can be searched effectively on receipt of each report.

A critical advantage is that the calculation of accuracy values for expectations can be carried out separately from the application of those expectations to reports. As the timeliness of recommending reports is crucial, removing the delay arising from the calculation of accuracy and coverage

values on receipt of each report is highly desirable.

Additionally, the cost of calculating the accuracy and coverage value for each expectation in the set is high. Generating the working expectations by initially identifying antecedents with high coverage values reduces the number of expectations that must be generated and evaluated. The expense of calculating the accuracy and coverage of an expectation is amortised over the evaluation of several reports.

The set of working expectations is generated according to the requirements of the EVA framework. Should these requirements alter, it is a simple matter to change the definition of the working set. For example, thus far only violations of expectations by a single entity have been considered. However, Chapter 2 introduces the notion of cohort violations; (see Section 3.3.1) that is, violations of expectations that involve more than one entity. The algorithm presented here may be modified to maximise the number of entities involved in the cohort whilst minimising the time frame over which the events occur and maximising the specificity of the entity specification.

A further advantage of this approach is that it is only necessary to consider those orders of variables that occur within expectations that have been fired. Only permutations of variables that appear in a representative formula will appear in the fset for a literal.

Finally, recalculating the accuracy values for expectations at intervals permits the use of only the most recent n reports rather than the set of all reports received. Doing so enables the system to respond to changes in behavioural trends over time by altering the strength of expectations accordingly.

There is a minor concern that the expectations chosen for the set of working expectations are those with high coverage values. As a result of this, an EVA system is able to spot unexpected behaviour in response to common circumstances but not able to spot unexpected behaviour in rare circumstances. However, this problem is not of great concern, as by definition, most reports will be of the common type. A different method may be devised that is able to identify rarely fired but useful expectations, but this is beyond the scope of this thesis.

Section 5.2.2 defines an algorithm that is used to deground representative sets, in order that they may be related to unground expectations. This is adequate, given that the expectations as defined in the EVA framework are unground and universally quantified. However, it may be the case that expectations that are partially ground are of use in certain circumstances. For example, a constant in an expectation could restrict the applicability of that expectation to a unique entity, a specific location, a particular industry sector and so on. Plotkin discusses a method of generalising formulae in [Plo70] such that generality is determined by the subsumption relation between two formulae. As such, it is possible to generalise from a ground, or partially ground, formula to a number of other partially ground or an unground formula.

Example 5.3.1. *Let* $\texttt{profitable(britishairways)} \wedge \texttt{sector(britishairways, aviation)}$ *be a formula. The degrounding algorithm defined in this chapter would render the deground*

version of this formula as:

$$\forall \mathtt{x}, \mathtt{y}\ \mathtt{profitable}(\mathtt{x}) \wedge \mathtt{sector}(\mathtt{x}, \mathtt{y})$$

However, the generalisation method suggested by Plotkin permits three possible generalisations:

$$\forall \mathtt{x}\ \mathtt{profitable}(\mathtt{x}) \wedge \mathtt{sector}(\mathtt{x}, \mathtt{aviation})$$
$$\forall \mathtt{x}\ \mathtt{profitable}(\mathtt{britishairways}) \wedge \mathtt{sector}(\mathtt{britishairways}, \mathtt{x})$$
$$\forall \mathtt{x}, \mathtt{y}\ \mathtt{profitable}(\mathtt{x}) \wedge \mathtt{sector}(\mathtt{x}, \mathtt{y})$$

Were the EVA framework to require us to consider partially ground expectations, Plotkin's approach to generalisation would be ideal. However, it is not the intention to examine partially ground expectations in this thesis.

It is worth noting that partially ground expectations can never have higher coverage values than a fully deground version of the same expectation. Therefore the method for generating the set of working expectations proposed here would discriminate against partially ground expectations and would perhaps need to be modified to overcome this drawback. This is a potential area for further work.

Once generated, the set of working expectations does not take into account any changes in the set of representative sets or the background knowledge. Should the predicate and constant symbols in the language change, then the set of working expectations would need to be recreated afresh. Likewise, the set of working expectations should be regenerated often enough to take account of any changes in the relative frequency of reported events.

The algorithms presented here provide one possible way of identifying working sets of expectations. Many of these algorithms suffer from the fact that they rely on several nested loops. However, as the simulations in the next chapter demonstrate, a prototype system based on these algorithms is able to generate sets of working expectations in a practical fashion.

In conclusion, the working set of expectations is a set of expectations that is small enough to be searchable and yet contains only those expectations that are strong enough to identify interesting news. Generating the set of working expectations in this manner is a good solution to the problem created by the large size of E.

ϵ	fset(ϵ)	vset(ϵ)	accuracy(ϵ)
$\forall x\ \alpha(x) \rightarrow \beta(x)$	$\{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5\}$	$\{\rho_3, \rho_4\}$	0.6
$\forall x\ \alpha(x) \rightarrow \neg\beta(x)$	$\{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5\}$	$\{\rho_1, \rho_2, \rho 5\}$	0.4
$\forall x\ \alpha(x) \rightarrow \gamma(x)$	$\{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5\}$	$\{\rho_3, \rho_4, \rho 5\}$	0.4
$\forall x\ \alpha(x) \rightarrow \neg\gamma(x)$	$\{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5\}$	$\{\rho_1, \rho_2\}$	0.6
$\forall x\ \neg\alpha(x) \rightarrow \beta(x)$	$\{\rho_6, \rho_7, \rho_8, \rho_9, \rho_{10}\}$	$\{\rho_7, \rho_8, \rho_9, \rho_{10}\}$	0.2
$\forall x\ \neg\alpha(x) \rightarrow \neg\beta(x)$	$\{\rho_6, \rho_7, \rho_8, \rho_9, \rho_{10}\}$	$\{\rho_6\}$	0.8
$\forall x\ \neg\alpha(x) \rightarrow \gamma(x)$	$\{\rho_6, \rho_7, \rho_8, \rho_9, \rho_{10}\}$	$\{\rho_6, \rho_7, \rho_8, \rho_9, \rho_{10}\}$	0
$\forall x\ \neg\alpha(x) \rightarrow \neg\gamma(x)$	$\{\rho_6, \rho_7, \rho_8, \rho_9, \rho_{10}\}$	$\{\}$	1
$\forall x\ \beta(x) \rightarrow \alpha(x)$	$\{\rho_1, \rho_2, \rho_5, \rho_6\}$	$\{\rho_6\}$	0.75
$\forall x\ \beta(x) \rightarrow \neg\alpha(x)$	$\{\rho_1, \rho_2, \rho_5, \rho_6\}$	$\{\rho_1, \rho_2, \rho_5\}$	0.25
$\forall x\ \beta(x) \rightarrow \gamma(x)$	$\{\rho_1, \rho_2, \rho_5, \rho_6\}$	$\{\rho_5, \rho_6\}$	0.5
$\forall x\ \beta(x) \rightarrow \neg\gamma(x)$	$\{\rho_1, \rho_2, \rho_5, \rho_6\}$	$\{\rho_1, \rho_2\}$	0.5
$\forall x\ \neg\beta(x) \rightarrow \alpha(x)$	$\{\rho_3, \rho_4, \rho_7, \rho_8\}$	$\{\rho_7, \rho_8\}$	0.5
$\forall x\ \neg\beta(x) \rightarrow \neg\alpha(x)$	$\{\rho_3, \rho_4, \rho_7, \rho_8\}$	$\{\rho_3, \rho_4\}$	0.5
$\forall x\ \neg\beta(x) \rightarrow \gamma(x)$	$\{\rho_3, \rho_4, \rho_7, \rho_8\}$	$\{\rho_3, \rho_4, \rho_7, \rho_8\}$	0
$\forall x\ \neg\beta(x) \rightarrow \neg\gamma(x)$	$\{\rho_3, \rho_4, \rho_7, \rho_8\}$	$\{\}$	1
$\forall x\ \gamma(x) \rightarrow \alpha(x)$	$\{\rho_1, \rho_2\}$	$\{\}$	1
$\forall x\ \gamma(x) \rightarrow \neg\alpha(x)$	$\{\rho_1, \rho_2\}$	$\{\rho_1, \rho_2\}$	0
$\forall x\ \gamma(x) \rightarrow \beta(x)$	$\{\rho_1, \rho_2\}$	$\{\}$	1
$\forall x\ \gamma(x) \rightarrow \neg\beta(x)$	$\{\rho_1, \rho_2\}$	$\{\rho_1, \rho_2\}$	0
$\forall x\ \neg\gamma(x) \rightarrow \alpha(x)$	$\{\rho_3, \rho_5, \rho_8, \rho_9, \rho_{10}\}$	$\{\rho_3, \rho_5\}$	0.4
$\forall x\ \neg\gamma(x) \rightarrow \neg\alpha(x)$	$\{\rho_3, \rho_5, \rho_8, \rho_9, \rho_{10}\}$	$\{\rho_8, \rho_9, \rho_{10}\}$	0.6
$\forall x\ \neg\gamma(x) \rightarrow \beta(x)$	$\{\rho_3, \rho_5, \rho_8, \rho_9, \rho_{10}\}$	$\{\rho_3, \rho_8, \rho_9, \rho_{10}\}$	0.2
$\forall x\ \neg\gamma(x) \rightarrow \neg\beta(x)$	$\{\rho_3, \rho_5, \rho_8, \rho_9, \rho_{10}\}$	$\{\rho_5\}$	0.8
$\forall x\ \beta(x) \wedge \alpha(x) \rightarrow \gamma(x)$	$\{\rho_1, \rho_2, \rho_5\}$	$\{\rho_5\}$	0.66
$\forall x\ \beta(x) \wedge \alpha(x) \rightarrow \neg\gamma(x)$	$\{\rho_1, \rho_2, \rho_5\}$	$\{\rho_1, \rho_2\}$	0.33
$\forall x\ \beta(x) \wedge \neg\alpha(x) \rightarrow \gamma(x)$	$\{\rho_6\}$	$\{\rho_6\}$	0
$\forall x\ \beta(x) \wedge \neg\alpha(x) \rightarrow \neg\gamma(x)$	$\{\rho_6\}$	$\{\}$	1
$\forall x\ \neg\beta(x) \wedge \alpha(x) \rightarrow \gamma(x)$	$\{\rho_3, \rho_4\}$	$\{\rho_3, \rho_4\}$	0
$\forall x\ \neg\beta(x) \wedge \alpha(x) \rightarrow \neg\gamma(x)$	$\{\rho_3, \rho_4\}$	$\{\}$	1
$\forall x\ \neg\beta(x) \wedge \neg\alpha(x) \rightarrow \gamma(x)$	$\{\rho_7, \rho_8, \rho_9, \rho_{10}\}$	$\{\rho_7, \rho_8, \rho_9, \rho_{10}\}$	0
$\forall x\ \neg\beta(x) \wedge \neg\alpha(x) \rightarrow \neg\gamma(x)$	$\{\rho_7, \rho_8, \rho_9, \rho_{10}\}$	$\{\}$	1
$\forall x\ \beta(x) \wedge \gamma(x) \rightarrow \alpha(x)$	$\{\rho_1, \rho_2\}$	$\{\}$	1
$\forall x\ \beta(x) \wedge \gamma(x) \rightarrow \neg\alpha(x)$	$\{\rho_1, \rho_2\}$	$\{\rho_1, \rho_2\}$	0
$\forall x\ \beta(x) \wedge \neg\gamma(x) \rightarrow \alpha(x)$	$\{\rho_5, \rho_6\}$	$\{\rho_5\}$	0.5
$\forall x\ \beta(x) \wedge \neg\gamma(x) \rightarrow \neg\alpha(x)$	$\{\rho_5, \rho_6\}$	$\{\rho_6\}$	0.5
$\forall x\ \neg\beta(x) \wedge \neg\gamma(x) \rightarrow \alpha(x)$	$\{\rho_3, \rho_4, \rho_7, \rho_8, \rho_9\}$	$\{\rho_7, \rho_8, \rho_9\}$	0.4
$\forall x\ \neg\beta(x) \wedge \neg\gamma(x) \rightarrow \neg\alpha(x)$	$\{\rho_3, \rho_4, \rho_7, \rho_8, \rho_9\}$	$\{\rho_3, \rho_4\}$	0.6
$\forall x\ \alpha(x) \wedge \gamma(x) \rightarrow \beta(x)$	$\{\rho_1, \rho_2\}$	$\{\}$	1
$\forall x\ \alpha(x) \wedge \gamma(x) \rightarrow \neg\beta(x)$	$\{\rho_1, \rho_2\}$	$\{\rho_1, \rho_2\}$	1
$\forall x\ \alpha(x) \wedge \neg\gamma(x) \rightarrow \beta(x)$	$\{\rho_3, \rho_4, \rho_5\}$	$\{\rho_3, \rho_4\}$	0.33
$\forall x\ \alpha(x) \wedge \gamma(x) \rightarrow \neg\beta(x)$	$\{\rho_3, \rho_4, \rho_5\}$	$\{\rho_5\}$	0.66
$\forall x\ \neg\alpha(x) \wedge \neg\gamma(x) \rightarrow \beta(x)$	$\{\rho_6, \rho_7, \rho_8, \rho_9, \rho_{10}\}$	$\{\rho_7, \rho_8, \rho_9, \rho_{10}\}$	0.2
$\forall x\ \neg\alpha(x) \wedge \gamma(x) \rightarrow \neg\beta(x)$	$\{\rho_6, \rho_7, \rho_8, \rho_9, \rho_{10}\}$	$\{\rho_6\}$	0.8

Table 5.1: Expectations and the firing and attacking sets for example 5.1.3

Chapter 6

Simulation of the working expectation generator

Chapter 2 introduced the EVA framework. Chapters 3 and 4 defined expectations. Chapter 5 demonstrated that the set of expectations can be reduced to a subset: the set of working expectations. This set is central to the EVA framework as it is the set of expectations that is used to identify interesting news items. It is therefore imperative that this set can be created in a manner that is computationally viable and that the set itself is manageable and meaningful (these terms will be elaborated on below).

A prototype working expectation generator was built following the definition and algorithms presented in Chapter 5. This prototype formed the basis for a series of simulations. These simulations demonstrate that it is possible to generate a set of working expectations that is *manageable*, that is, the size of the set of working expectations is small enough to be easily searched. The simulations also demonstrate that these sets of working expectations are *meaningful*. In order for these sets of working expectations to be meaningful, identical inputs should result in identical outputs and "similar" inputs should result in "similar" outputs. Measures of similarity are defined in the next section. Additionally, the simulations demonstrate that the process of generating the set of working expectations is *computationally viable*, that is, the input needed and time taken to generate the set of working expectations is reasonable.

There are four simulations whose results will be presented in this chapter. These simulations address the questions of manageability, meaningfulness and viability raised above.

- Simulation One (Section 6.2) demonstrates that, given a large enough set of representative sets, the set of working expectations converges to a *canonical set*, which will be defined in that section. It is necessary that, given similar sets of representative sets and identical

threshold values as inputs, the system should generate similar sets of working expectations. This is because the members of the set of working expectations are entirely dictated by the membership of the set of representative sets and the thresholds chosen. Simulation One shows that for populations of representative sets with the same predicate symbol distribution the set of working expectations is, for most distributions, identical or highly similar.

- Simulation Two (Section 6.3) provides an indication of the similarity between of working sets of expectations given sets of representative sets of various cardinalities. This provides an understanding of the amount of evidence (in the form of representative sets) required to generate sets of working expectations with high precision and recall values. High precision and recall values suggest that the set of working expectations is a good approximation to the *canonical set* of working expectations.

- Simulation Three (Section 6.4) demonstrates the effect of increasing the complexity of the universe by increasing the number of degrounding symbols. The more complex the universe, the greater the size of the set of expectations. As the size of the set of expectations grows, the number of representative sets required to result in convergence to a canonical set also increases. Simulation Three examines the effect of increasing the number of degrounding symbols on the ability of the system to generate the set of working expectations.

- Simulation Four (Section 6.5) demonstrates a method for biasing the set of working expectations towards those expectations with long antecedents. Expectations with more specific (i.e. longer) antecedents provide a more useful indication of interesting behaviour. However, expectations with shorter antecedents have equal or greater fired values than longer ones (see Proposition 4.2.6 for an explanation). Therefore, all things being equal, the set of working expectations is more likely to contain expectations with n conjuncts antecedents than with $n + 1$ conjuncts. Simulation Four demonstrates that a stepped coverage threshold is capable of biasing the set of working expectations such that it contains expectations with longer antecedents.

The decision was taken to carry out these simulations using artificially created representative sets, rather than extracting news from real reports. The motivation behind this decision is twofold: artificially created data may be better understood than genuine data. The process of generating the representative sets ensures familiarity with the characteristics of that data, such as the distribution of predicate symbols, the arity of predicates, the cardinality of representative sets and so on. It is possible to test the prototype's performance under a variety of different scenarios by varying the characteristics of the sets of representative sets. These advantages outweigh the potential loss of information that results from not using real reports in the simulations. The results gathered from an analysis of business news reports from Reuters and The Economist magazine in Section 6.1.7 suggest that real data from the business domain is similar to one of the probability distributions used to create artificial representative sets. As such, we can expect similar performance when using reports from the business domain to those that result from using simulated data.

This chapter is structured as follows: Section 6.1 presents the algorithms used to run the simulations and defines the measures that are applied to the results. Section 6.1.5 elaborates on the reasons for using automatically generated representative sets and presents the different characteristics that these representative sets may have. This section also highlights the dangers of ontological bias, where the choice of the structure of the language can profoundly affect the performance of the working expectation generator. This section also demonstrates that the distribution of predicates for a corpus of business news conforms closely to the Zipf distribution. Sections 6.2 to 6.5 present the simulations introduced above and their results. Section 6.6 presents a summary of the findings and argues that the working expectation generator as defined in Chapter 5 is computationally viable and creates meaningful and manageable results.

6.1 Definitions and methods

The definitions of terms, predicates, literals and groundings should be taken as those given in Section 2.2. A representative set is a set of literals containing information extracted from a report and from the background knowledge, as given in Definition 2.4.5. This section defines the method and tools used in carrying out the simulations presented in this chapter. Two sets of real news stories were used at various points during this work. The Economist corpus is a set of 470 weekly digests of, on average, 19 reports per digest, taken from the period January 2000 to August 2004. The Reuters corpus is a set of 4,049 business news articles from the period September 1996 to August 1997.

The computing environment in which the simulations were run was a PC running Windows 2000 on a 1.8 GHz Pentium 4 processor with 1 gigabyte of RAM. The time for a single run varied from simulation to simulation but, in the worst recorded case, took no more than seven minutes to run. The less challenging simulations took less than one minute per run.

6.1.1 The representative set generator

In order to test the performance of the working expectation generator it is necessary to provide some input in the form of representative sets. For the justification for generating the representative sets automatically see Section 6.1.5. In this way it was possible to examine the effect of different probability distributions (Section 6.1.2) and different degrees of complexity in the universe.

Sets of representative sets are generated by a system that accepts four parameters: the number of representative sets to be generated; the number of constant symbols in the language; the arity of literals and the predicate symbol distribution. Representative sets are sets of ground atoms. The representative set generator creates representative sets using the Java random number generator. Each parameter of each representative set is determined using the random number generator

according to the following probabilities:

- Number of literals in representative set: fixed probabilities shown in Table 6.1. These probabilities were chosen to be representative of the size of news reports from the Economist and Reuters corpora.
- Whether the literal is negated or positive : fixed probabilities: P(positive)=0.5, P(negated)=0.5
- Constant symbol, where, for some n, each constant is in the set $\{c_1, ..., c_n\}$: fixed probability, $\frac{1}{n}$
- Predicate symbol : probability is dependent on the predicate symbol distribution chosen by the experimenter. Probabilities are given in Table 6.2.

The number of representative sets and the distribution are sent to the representative set generator as input. For each representative set to be generated a random number is chosen that fixes the number of literals in the representative set. For each literal, the predicate symbol is chosen using a random number generator. The distribution chosen by the experimenter determines which predicate symbol is generated by each random number. Further random numbers determine whether the literal is negated and the subscripts of the constants.

Number of literals	2	3	4	5	6
Probability	0.2	0.4	0.15	0.15	0.1

Table 6.1: Fixed probabilities governing the number of literals in each representative set.

The following algorithm is responsible for generating the set of representative sets:

Definition 6.1.1. *Let* `cRandomNumber` *be a random number. Let the function* `integer(cRandomNumber, maxConstants)` *convert a floating point number an integer in the range* $1, ..., \texttt{maxConsts}$.

Let `sRandomNumber` *be a random number and* `pRandomNumber` *be a random number. Let* `getSetSize(sRandomNumber)` *return an integer that will determine the number of literals in the representative set, according to the representative set size distribution in Table 6.1. Let* `getPredSymbol(pRandomNumber, predicateDistribution)` *return a predicate symbol according to the distributions shown in Section 6.1.2.*

Let each representative set be an object (informally, a collection of different data types) $\{\texttt{atomSign}, \texttt{predSymbol}, \texttt{constants}\}$ *where* `atomSign` *is either null or takes the value* $\neg$, `predSymbol` *is one of the predicate symbols in the universe and* `constants` *is an array of strings, each of which is a constant symbol from* $c_1, ..., c_n$.

The function `generatePi(piSize, maxConstants, predicateDistribution, arity)` *returns a set of* `piSize` *representative sets whose predicate symbols are chosen according to the distribu-*

tion `predicateDistribution` *and whose constants are numbered with an integer in the range* 0, ..., `maxConstants`. *The arity of predicates is* `arity`.

```
generatePi(piSize, maxConstants, predicateDistribution, arity)
 1 repsets[]
 2 counter ← 0
 3 for i in 1 to piSize
 4       repsets[i] ← generateRepset(maxConstants, predicateDistribution, arity)
 5 return repsets;
```

The function `generateRepset(maxConstants, predicateDistribution, arity)` *returns a single representative set. Let* `concat(string, integer)` *concatenate a string and an integer such that* `concat("c", 1)` *returns* "c1". *Let* "¬" *be the negation symbol.*

```
generateRepSet(maxConstants, predicateDistribution, arity)
 1 repset[]
 2       sRandomNumber ← newRandomNumber()
 3       repSetSize ← getSetSize(sRandomNumber)
 4       for i in 1 to setSize
 5             do
 6                negRandomNumber ← newRandomNumber()
 7                if negRandomNumber ≤ 0.5
 8                      negSymbol =" "
 9                else
10                       negSymbol =" ¬"
11                pRandomNumber ← newRandomNumber()
12                predicateSymbol ←
                     getPredSymbol(pRandomNumber, predicateDistribution)
13                for j in 1 to arity
14                       cRandomNumber ← newRandomNumber()
15                       constSymbols[k] ←
                            concat("c", integer(cRandomNumber, maxConstants))
16             while {literalSign, predicateSymbol, constSymbols} in repset
17 return repset
```

6.1.2 Predicate symbol distributions

Representative sets are created by the representative set generator (see Section 6.1.1). Each representative set is a set of ground literals. The representative set generator must create these literals

from the set of predicate symbols, $\mathcal{P} = \{p_1, ..., p_n\}$, the negation operator, "$\neg$", and the set of constant symbols, $\mathcal{C} = \{c_1, ..., c_n\}$. Predicate symbol distributions are used to determine the probability with which each predicate symbol occurs in the set of representative sets.

Predicate Symbol:	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
One Up	0.51	0.07	0.07	0.07	0.07	0.07	0.07	0.07
Five Up	0.17	0.17	0.17	0.17	0.17	0.05	0.05	0.05
Three Up	0.15	0.15	0.15	0.11	0.11	0.11	0.11	0.11
Big Three	0.3	0.3	0.3	0.02	0.02	0.02	0.02	0.02
Uniform	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
Pseudo Gaussian	0.002	0.021	0.136	0.341	0.341	0.136	0.021	0.002
Flat Gaussian	0.05	0.1	0.15	0.2	0.2	0.15	0.1	0.05
Pseudo Zipf	0.4	0.2	0.13	0.1	0.08	0.06	0.02	0.01

Table 6.2: Predicate symbol distributions for universe of 8 predicate symbols (exact figures).

There are eight predicate symbol distributions presented in Table 6.2 and Figures 6.1 and 6.2: the Uniform distribution has a single level of probability, distributions One up, Three up, Five Up and Big Three have two levels of probability (these will be referred to as stepped distributions), the Pseudo Gaussian and Flat Gaussian distributions have four levels of probability and the Pseudo Zipf distribution has eight levels of probability. The Pseudo Gaussian, Flat Gaussian and Pseudo Zipf will be referred to as the natural distributions. These distributions were chosen as it was felt that, although the values chosen are somewhat arbitrary, the distributions are varied enough to adequately represent the range of possible distributions of predicate symbols in representative sets.

The figures given in Table 6.2 are the exact probabilities of each predicate symbol occurring, whether in a positive or negative ground predicate. The negation symbol has an even chance of being applied to a predicate. Therefore, if $\langle c_1, ..., c_n \rangle$ is a possible tuple of constants then the probability of the ground predicate $p_1(c_1, ..., c_n)$ being generated is the same as the probability of the ground predicate $\neg p_1(c_1, ..., c_n)$ being generated.

Example 6.1.1. *Let the probability of predicate symbol* p_1 *being generated be* 0.5. *Let the probability of the tuple of constant symbols* $\langle c_1, c_2 \rangle$ *being generated be* 0.3. *the probability of the negation symbol being applied to a predicate is* 0.5. *The probability of the predicate* $p_1(c_1, c_2)$ *being generated is therefore* $0.5 \times 0.3 \times 0.5 = 0.075$. *Likewise, the probability of the predicate* $\neg p_1(c_1, c_2)$ *being generated is therefore* $0.5 \times 0.3 \times 0.5 = 0.075$.

The Uniform distribution is the distribution in which each predicate symbol is equiprobable, so there will be little or no difference between the coverage value of antecedents of the same length and little or no difference between the attacked value of each constant. In a set of expectations of equal antecedent length, there should be no expectation that has a significantly higher accuracy or coverage value than any other.

Example 6.1.2. *Let* $|\mathcal{C}| = 2$. *The possible values for* $\bar{c}$ *are then* (c_1, c_1), (c_1, c_2), (c_2, c_1), *and*

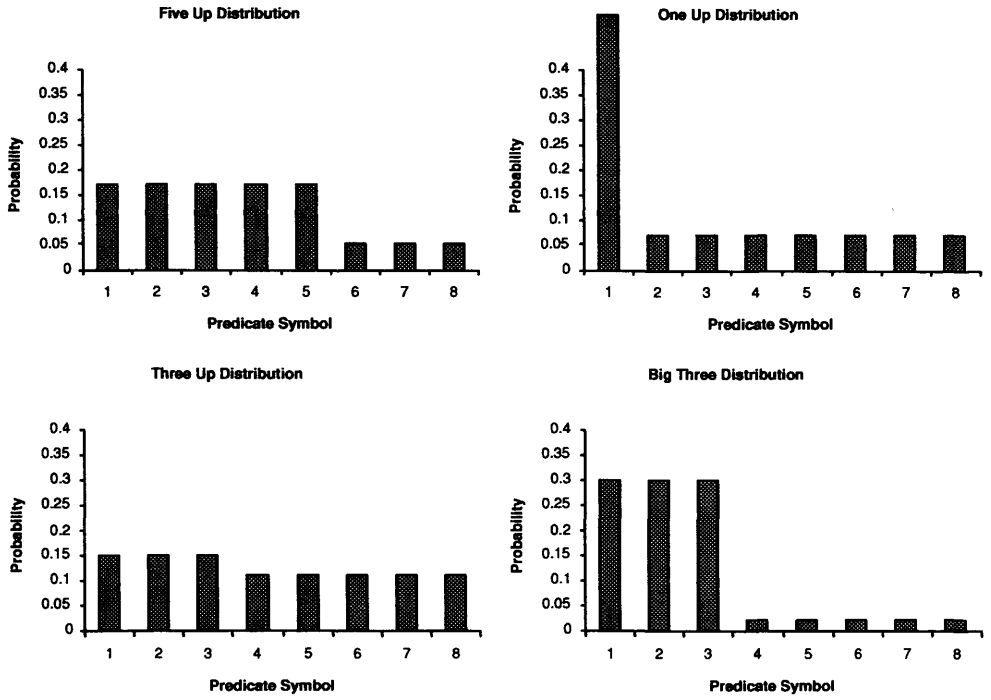


Figure 6.1: The stepped distributions of predicate symbols as given in Table 6.2.

(c_2, c_2). *Each tuple is equiprobable. The probability of each ground literal being negated is* 0.5. *The probability of* $p_1 = 0.125$. *Let the probability of a predicate symbol* p *be* probability(p), *the probability of a tuple of constants* t *be* probability(t) *and the probability of a negation symbol* n *(where* $n = [\neg, null]$*) be* probability(n). *The probability of any one literal (for example* $\neg p_1(t)$*) is* probability$(\neg) \times$ probability$(p_1) \times$ probability(t), *which equals* $0.5 \times 0.125 \times 0.25 = 0.0156$.

The One Up distribution is the simplest of the stepped distributions, where one predicate symbol (p_1) is much more likely to be generated than any other. Consequents whose predicate symbol is p_1 are therefore much more likely to be attacked than others. Antecedents for which p_1 is the predicate symbol of one of the conjuncts are also more likely to be fired than those for which p_1 is not the predicate symbol of one of the conjuncts.

Example 6.1.3. *Let* $|\mathcal{C}| = 2$. *The possible values for* $\bar{c}$ *are then* (c_1, c_1), (c_1, c_2), (c_2, c_1), *and* (c_2, c_2). *Each tuple is equiprobable. The probability of each literal being negated is* 0.5. *In the One Up distribution, the probability of* $p_1 = 0.51$. *Let the probability of a predicate symbol* p *be* probability(p), *the probability of a tuple of constants* t *be* probability(t) *and the probability of a negation symbol* n *(where* $n = [\neg, null]$*) be* probability(n). *The probability of any one literal (for example* $\neg p_1(t)$*) is* probability$(\neg) \times$ probability$(p_1) \times$ probability(t), *which equals* $0.5 \times 0.51 \times 0.25 = 0.0643$.

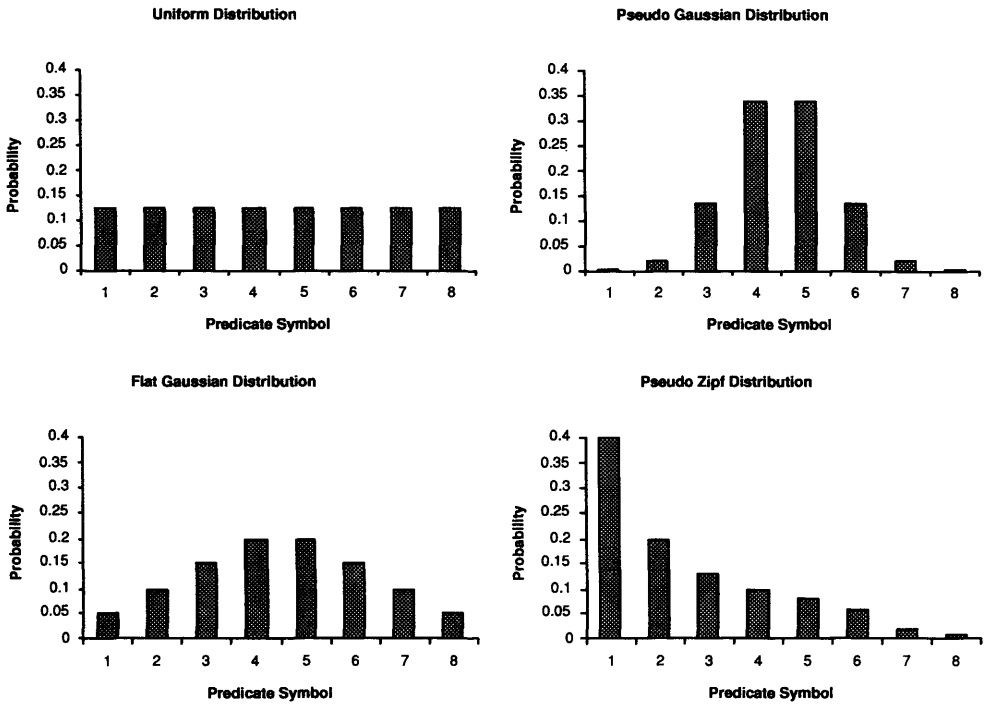


Figure 6.2: The natural distributions of predicate symbols as given in Table 6.2.

The Three Up distribution has a slightly higher probability of predicate symbols p_1, p_2 or p_3 occurring ($P = 0.15$) than $p_4, ..., p_8$ ($P = 0.11$). Therefore those expectations whose antecedents contain p_1, p_2 or p_3 or a combination thereof as conjuncts are more likely to occur than those that do not. However, the difference in these probabilities is small, which means that this distribution is susceptible to the effect of small differences in the output of the random number generator.

The Three Up distribution is more complex than the One Up distribution as there is a much higher probability of certain combinations of predicate symbols occurring than others.

Example 6.1.4. *For the Three Up distribution, the probability of the predicate symbols* p_1 *and* p_2 *occurring together in a two literal representative set is* 0.0225 *whereas the probability of* p_1 *and* p_4 *occurring together in a two literal representative set is* 0.0165 *and the probability of* p_4 *and* p_5 *occurring together in a two literal representative set is* 0.0121.

The Five Up predicate symbol distribution is another stepped distribution. The difference between the levels of probability is more pronounced than for the Three Up distribution. There are also strong probabilities of co-occurrence between $p_1, ..., p_5$.

Example 6.1.5. *For the Five Up distribution, the probability of predicate symbols* p_1 *and* p_2 *occurring together in a two literal representative set is* 0.0289 *whereas the probability of* p_1 *and*

p_6 occurring together in a two literal representative set is 0.0085 *and the probability of p_6 and p_7 occurring together in a two literal representative set is* 0.0025.

The Big Three distribution was included to highlight the effect of a larger difference between the levels of probability assigned to predicates. The difference between the levels of probability for $p_1, ..., p_3$ and $p_4, ..., p_8$ in the Big Three distribution is much greater than the difference in the Three Up distribution. This distribution should result in a small number of well covered antecedents and five rarely attacked consequents.

The probabilities of co-occurrences are highly variable:

Example 6.1.6. *For the Big Three distribution, the probability of predicate symbols p_1 and p_2 occurring together in a two literal representative set is* 0.09 *whereas the probability of predicate symbols p_1 and p_4 occurring together in a two literal representative set is* 0.006 *and the probability of p_4 and p_5 occurring together in a two literal representative set* 0.0004.

The Pseudo Gaussian distribution is natural in that it mimics, as closely as practical, the normal or Gaussian distribution. The normal distribution has been found to provide a reasonable approximation of the distribution of data in many situations and is often used in statistical analysis when the true distribution of a dataset is unknown. It is not possible for the predicate distribution to be identical to the Gaussian distribution as, under the Gaussian distribution the curve is continuous and tends towards zero but extends infinitely. For implementation purposes it was necessary to have eight probability values that sum to one in order that each random number generated is attributable to one of the eight outcomes. There are four levels of probability in this distribution, with a large difference between the levels.

Example 6.1.7. *Let $|\mathcal{C}| = 2$. The possible values for $\bar{c}$ are then (c_1, c_1), (c_1, c_2), (c_2, c_1), and (c_2, c_2). Each tuple is equiprobable. The probability of each literal being negated is* 0.5. *In the Pseudo Gaussian distribution, the probability of $p_1 = 0.002$. Let the probability of a predicate symbol p be* probability(p), *the probability of a tuple of constants t be* probability(t) *and the probability of a negation symbol n (where $n = [\neg, null]$) be* probability(n). *The probability of any one literal (for example $\neg p_1(t)$) is* probability($\neg$) $\times$ probability(p_1) $\times$ probability(t), *which equals* $0.5 \times 0.002 \times 0.25 = 0.0003$.

The probabilities of co-occurrences are highly variable as the following example demonstrates:

Example 6.1.8. *For the Pseudo Gaussian distribution, the probability of predicate symbols p_4 and p_5 occurring together in a two literal representative set is* 0.1163 *(to 4dp) whereas the probability of predicate symbols p_1 and p_8 occurring together in a two literal representative set is vanishingly small, at* 4×10^{-6}.

The Flat Gaussian distribution is a symmetrical distribution but in the differences between the levels of the probabilities are much smaller than in the Gaussian distribution. Because the difference between the probability levels is less pronounced than in the Pseudo Gaussian distribution,

this distribution is more susceptible to variations in the output of the random number generator. There is also a smaller spread of probabilities of co-occurrence than under the Pseudo Gaussian distribution:

Example 6.1.9. *For the Flat Gaussian distribution, the probability of predicate symbols p_4 and p_5 occurring together in a two literal representative set is* 0.04 *whereas the probability of predicate symbols p_1 and p_8 occurring together in a two literal representative set is much lower, at* 0.0025.

The Pseudo Zipf distribution is another natural distribution, based on the Zipf distribution, used in domains such as web content distribution ([SGD+02]), natural language processing ([Sen98]) and economics ([Foc01]). For implementations sake a simplified form is used. The spread of probabilities of co-occurrence is wide:

Example 6.1.10. *For the Pseudo Zipf distribution, the probability of predicate symbols p_1 and p_2 occurring together in a two literal representative set is* 0.08 *whereas the probability of predicate symbols p_7 and p_8 occurring together in a two literal representative set is much lower, at* 0.0002.

These distributions will be referred to throughout the chapter. They are used in the generation of representative sets that are used as input to the prototype during the simulations. A variety of predicate distributions are used in order to test the behaviour of the working set generator under a variety of conditions.

In order to determine the which predicate distribution was most representative of the news in the Economist and the Reuters corpora, a number of templates were created for use in the Biorat information extraction tool (see[CBLJ04]). Section 6.1.7 examines the distributions within these corpora. Briefly, they share the characteristics of the Pseudo Zipf distribution used in this simulation. Likewise, the size of the set of the space of all expectations is similar to that in some of the simulations presented here. It is possible to be confident that the working expectation generator would perform similarly on the data extracted from these corpora as it would on the simulated data, as the characteristics are very similar. However, the simulations also provide us with a range of data that demonstrate the sorts of expectation spaces that would be suited to the use of the working expectation generator, regardless of the domain.

The effect of noise on the working expectation generator has not been examined. In order to determine the rates of noise in the corpora available it is necessary to undertake a manual examination of the reports therein. Due to time constrains this was not possible and the question remains open for future work.

6.1.3 Measures of convergence: recall and precision

In order to evaluate the working expectation generator it is necessary to understand its performance. The measure of this performance must be related to the purpose of the working expectation generator, which is to generate a set of working expectations that is a meaningful representation of the universe described in the set of representative sets and the background knowledge. Therefore, if the working set of expectations is truly representative of the events described in the representative sets, sets of representative sets describing similar sets of events should result in similar sets of working expectations.

If this is the case, two simulations, described by the same language and distribution, may result in different sets of representative sets and background knowledge but should still result in similar sets of working expectations. The degree of similarity between these sets can be measured by their convergence. When two or more sets of working expectations have identical membership then they have converged:

Definition 6.1.2. *Let $R_1, ..., R_n$ be sets of representative sets and $\kappa \in \mathbb{N}$ be a coverage threshold and α in the range $[0, 1]$ be an accuracy threshold. Let* $\mathsf{workingset}(R, \kappa, \alpha)$ *be the set of expectations whose coverage based on the set R is greater than κ and whose accuracy based on the set R is greater than α.*

$\mathsf{workingset}(R_j, \kappa, \alpha) = \mathsf{workingset}(R_k, \kappa, \alpha)$ *iff:*

$$\mathsf{workingset}(R_j, \kappa, \alpha) \backslash \mathsf{workingset}(R_k, \kappa, \alpha) = \emptyset$$

and

$$\mathsf{workingset}(R_k, \kappa, \alpha) \backslash \mathsf{workingset}(R_j, \kappa, \alpha) = \emptyset$$

$\mathsf{workingset}(R_1, \kappa, \alpha), ..., \mathsf{workingset}(R_n, \kappa, \alpha)$ *are* **fully converged** *iff:*

$$\mathsf{workingset}(R_1, \kappa, \alpha) = ... = \mathsf{workingset}(R_n, \kappa, \alpha)$$

Example 6.1.11. *Let* $\mathsf{workingset}(R_1, \kappa, \alpha) = \{\epsilon_1, \epsilon_2, \epsilon_3\}$ *and* $\mathsf{workingset}(R_2, \kappa, \alpha) = \{\epsilon_1, \epsilon_2, \epsilon_3\}$. $\mathsf{workingset}(R_1, \kappa, \alpha) \backslash \mathsf{workingset}(R_2, \kappa, \alpha) = \emptyset$. *Therefore* $\mathsf{workingset}(R_1, \kappa, \alpha)$ *and* $\mathsf{workingset}(R_2, \kappa, \alpha)$ *are fully converged.*

However, at lower levels of evidence (that is, fewer representative sets) the working expectations generator is more prone to the effects of small changes in representative sets. As shown in Section 6.1.2, the difference between the probability of the most commonly produced representative set and the least commonly produced representative may be extremely large. In this case, even at small levels of evidence, there will be a set of common expectations that occur in all sets of working expectations and a set of expectations that occur in no sets of working expectations.

However, less differentiated distributions lead to a greater variety of working expectations. There is less chance of full convergence. It is, therefore, necessary to have a measure of this partial convergence.

Definition 6.1.3. *Let $R_1, ..., R_n$ be n sets of representative sets and $\kappa \in \mathbb{N}$ be a coverage threshold and α in the range $[0, 1]$ be an accuracy threshold. Let* $\mathsf{workingset}(R, \kappa, \alpha)$ *be the set of expectations whose coverage based on the set R is greater than κ and whose accuracy based on the set R is greater than α. If*

$$\mathsf{workingset}(R_1, \kappa, \alpha) \cap ... \cap \mathsf{workingset}(R_n, \kappa, \alpha) \neq \emptyset$$

then $\mathsf{workingset}(R_1, \kappa, \alpha)...\mathsf{workingset}(R_n, \kappa, \alpha)$ *are* **partially converged.**

Let

$$\mathsf{aveSetSize} = \frac{|\mathsf{workingset}(R_1, \kappa, \alpha)| + ... + |\mathsf{workingset}(R_n, \kappa, \alpha)|}{n}$$

Let

$$\mathsf{convergedSetSize} = |\mathsf{workingset}(R_1, \kappa, \alpha) \cap ... \cap \mathsf{workingset}(R_n, \kappa, \alpha)|$$

The function

$$\mathsf{convergence} = \frac{\mathsf{convergedSetSize}}{\mathsf{aveSetSize}}$$

Example 6.1.12. *Let* $\mathsf{workingset}(R_1, \kappa, \alpha) = \{\epsilon_1, \epsilon_2, \epsilon_3\}$ *and let* $\mathsf{workingset}(R_2, \kappa, \alpha) = \{\epsilon_1, \epsilon_2, \epsilon_4\}$. $\mathsf{workingset}(R_1, \kappa, \alpha) \cap \mathsf{workingset}(R_2, \kappa, \alpha) = \{\epsilon_1, \epsilon_2\}$.

As a result, $\mathsf{aveSetSize} = 3$, $\mathsf{convergedSetSize} = 2$ *and* $\mathsf{convergence} = 0.6667$ *(4 decimal places)*

Very large numbers of reports may be required to reach a state of full convergence. However, as later results will demonstrate, there is a trend such that the greater the number of representative sets supplied to the working expectation generator, the greater degree of convergence there will be.

Whilst not requiring total convergence, the canonical set of working expectations is a set of expectations drawn from several executions of the working expectation generator using the same values for α and κ and independently generated representative sets of the same distribution. The outputs from these executions are compared and the common members identified. If there is sufficient commonality then the common members form the canonical set.

Definition 6.1.4. *Let $R_1, ..., R_n$ be sets of representative sets, α be a lower bound on accuracy*

and κ be a lower bound on coverage. Let $\mathsf{workingset}(R_1, \kappa, \alpha), ..., \mathsf{workingset}(R_n, \kappa, \alpha)$ *be the working sets of expectations generated with respect to $R_1, ..., R_n$, α and κ. Let the* **common set** *of expectations,* $\mathsf{commonSet}(R_1...R_n, \kappa, \alpha)$ *be a set of expectations such that each expectation is in at least* 90% *of the sets* $\mathsf{workingset}(R_1, \kappa, \alpha), ..., \mathsf{workingset}(R_n, \kappa, \alpha)$.

$\mathsf{commonSet}(R_1...R_n, \kappa, \alpha) = \mathsf{canonicalSet}(R_1...R_n, \kappa, \alpha)$ *if and only if*

$$|\mathsf{commonSet}(R_1...R_n, \kappa, \alpha)| \geq 0.9 \times \frac{|\mathsf{workingset}(R_1, \kappa, \alpha)| + ... + |\mathsf{workingset}(R_n, \kappa, \alpha)|}{n}$$

Otherwise there is no canonical set for the set of working sets. This ensures that common sets that are unrepresentative of the individual working sets, because they are small, are excluded from consideration. If the cardinality of the canonical set is less than 90% *of the average working set size, then it will not be used as a basis of comparison.*

Whilst the 90% bound is chosen arbitrarily, it ensures that the canonical set of working expectations contains at least 90% of the expectations generated and these expectations are in at least 90% of the sets of working expectations generated. This allows us to be confident that the canonical set is a good representation of the real world in that it includes those expectations that at least 90% of working sets agree should be present and at the same time insists that the sets of working expectations should be consistent, on average, 90% of the time. Note that a canonical set is required only for the purposes of the simulation. An EVA system need only create a single working set of expectations in order to identify interesting news reports, as long as it is possible to be confident that the domain has characteristics that lead to high levels of convergence.

The canonical set is used to examine the performance of the working expectation generator. The recall measure determines how many false negatives (with respect to the canonical set of expectations) are excluded from the set of working expectations and the precision measure determines how many false positives are included in the set of working expectations.

Definition 6.1.5. *Let $R_1, ..., R_m$ be m sets of representative sets of a given predicate symbol distribution and maximum arity, R_n be a set of representative sets created using the same predicate symbol distribution, α be a threshold on accuracy and κ be a threshold on coverage.*

The number of canonical expectations is,

$$\mathsf{canonical}(R_1, ..., R_m, \kappa, \alpha) = |\mathsf{canonicalset}(R_1, ..., R_m, \kappa, \alpha)|$$

The number of retrieved expectations is, $\mathsf{retrieved}(R_n, \kappa, \alpha) = |\mathsf{workingset}(R_n, \kappa, \alpha)|$.

The number of those canonical expectations that have been retrieved is,
$\mathsf{retcan}(R_1, ..., R_m, R_n, \kappa, \alpha) = |\mathsf{canonicalset}(R_1, ..., R_m, \kappa, \alpha) \cap \mathsf{workingset}(R_n, \kappa, \alpha)|$.

The recall value, the proportion of retrieved expectations that are canonical, is

$$\mathsf{recall}(R_1, ..., R_m, R_n, \kappa, \alpha) = \frac{\mathsf{retcan}(R_1, ..., R_m, R_n, \kappa, \alpha)}{\mathsf{canonical}(R_1, ..., R_m, \kappa, \alpha)}$$

The precision value, that is, the number of retrieved expectations that are canonical is,

$$\mathsf{precision} = \frac{\mathsf{retcan}(R_1, ..., R_m, R_n, \kappa, \alpha)}{\mathsf{retrieved}(R_n, \kappa, \alpha)}$$

The working expectation accepts as input a set of representative sets (hereafter referred to as R), a real number threshold value for accuracy, α and an integer threshold for coverage, κ. The system returns a set of expectations whose accuracy is above the threshold α and coverage is above the threshold κ, given R.

The set R is created using a representative set generator (see Section 6.1.1). The set of representative sets is generated to a given size and distribution. The working expectation generator was fully tested prior to deployment. The output for each predicate symbol distribution was plotted to ensure that the atoms in R were generated with the correct frequency.

6.1.4 Testing the working set generator

In order to carry out the simulations, it was necessary to ensure that the working expectation generator was a faithful reproduction of the definitions in Chapter 5. Testing was carried out to uncover any discrepancies that may suggest either a) that the implementation was incorrect with respect to the definitions in Chapter 5 or that b) the sets in Chapter 5 were incorrectly defined.

In this chapter the definition of the working expectation generator is the same as that in Chapter 5. The first stage is to create the set $\mathsf{coveredAntecedents}(M, \rho, \Gamma, \Delta)$ according to Definition 5.1.1. Secondly the set $\mathsf{goodExpectations}(M, repSets, maxCon, minCoverage)$ is created according to the Definition 5.1.2. Finally the set $\mathsf{workingExpectations}(M, repSets, maxCon, minCoverage, minAccuracy)$ is generated according to Definition 5.1.3.

- **Literals** In Chapter 5 the data type for a literal is an array containing a predicate symbol that may be negated and one or more terms. In the Java implementation used as the basis for the simulations in this chapter, the data type is an object. The class is used by both the representative set generator and the working expectation generator. The object's attributes

are:

```
public class literal {
        boolean negated; true if is a negative atom, false otherwise
        String predSymbol; the predicate symbol
        String[] terms; either constants or variables
        }
```

In order to ensure that the set of possible expectations is a manageable size, the universe was restricted to 8 predicate symbols and their negations: 16 unground predicates in total. In the first run, the language was restricted to monadic predicates and one degrounding symbol. In later runs the language was expanded to include dyadic predicates and up to four degrounding symbols.

- **Representative sets** are as defined in Chapter 5. The class is used by both the representative set generator and the working expectation generator. A representative set is an object with the following attributes:

```
public class representativeSet{
        long identifier;
        atom[] representativeSet;
        }
```

The variable `identifier` is used as an indexing variable, in order that fsets and asets can be arrays of `longs` rather than arrays of representative sets.

The set of representative sets is a static object, `Repsets`, that acts as a container for representative sets. The class is used by both the representative set generator and the working expectation generator. The attribute definition is as follows:

```
public class Repsets {// the set of all representative sets
        representativeSet[] Repsets; // an array of representative sets
        int counter; //position of the highest representative set in Repsets
        int portion; //the size of Repsets
}
```

Representative sets are pushed on to the end of the array `Repsets`. The variable `counter` records the position of the most recent representative set to be added to `Repsets`. Representative sets, once added, are never removed in this implementation. The variable `portion` is used as a parameter to define the size of the array of representative sets, `Repsets` and can be increased during execution.

The probability distributions are only relevant to the generation of representative sets and, consequently, do not appear in the working expectation generator. The code for the predicate symbol generator that implements the Big Three distribution over 8 predicate symbols

is as follows:

```
public static int makeBigThreeIndex(float indexRand){
          if(indexRand<0.3)
               return 0;
          else if(indexRand<0.6)
               return 1;
          else if(indexRand<0.9)
               return 2;
          else if(indexRand<0.92)
               return 3;
          else if(indexRand<0.94)
               return 4;
          else if(indexRand<0.96)
               return 5;
          else if(indexRand<0.98)
               return 6;
          else
               return 7;
}
```

The integer returned is the index for the predicate symbol of a literal in the representative set being generated. The comparison operators in the switch statements compare a random number in the interval [0, 1] with the cumulative probability of each predicate symbol index being generated.

- **Threshold values** for coverage and accuracy, as defined in Chapter 5 were supplied as parameters to the working expectation generator. Cutoffs varied from 0 to 1 for the accuracy threshold and 0, 5,000 and 10,000 representative sets for coverage. Each set of threshold values was used as input to 10 runs of the system.

- **Output** The set of representative sets and the set(s) of working expectations were written to files for further analysis. Below is an example of the output from the representative set generator:

  `Report`
  `listed(constant0, constant1)lnotsharerise(constant1, constant0)`

  `Report`
  `lnotsharerise(constant1, constant0)buyer(constant1, constant0)`

  `Report`
  `lnotsharerise(constant1, constant0)lnotsharerise(constant0, constant1)`
  `bankrupt(constant0, constant1)`

  `Report`

lnotsharerise(constant0, constant1)buyer(constant1, constant0)
lnotnewjobs(constant1, constant0)lnotsharerise(constant1, constant0)
bankrupt(constant0, constant1)

The following is an extract of the output from the working expectation generator. The @ character delimits entries. The text entry is the expectation that has been generated; the first numeric value is the coverage for that expectation and the second numeric value is the accuracy of that expectation:

$\texttt{lnotsharerise}(v_0, v_1) \wedge \texttt{lnotbuyer}(v_1, v_0) \rightarrow \texttt{sharerise}(v_1, v_0)$@351@0.93162394
$\texttt{lnotsharerise}(v_0, v_1) \wedge \texttt{lnotbuyer}(v_1, v_0) \rightarrow \texttt{newjobs}(v_1, v_0)$@351@0.92022794
$\texttt{lnotsharerise}(v_0, v_1) \wedge \texttt{lnotbuyer}(v_1, v_0) \rightarrow \texttt{lnotbankrupt}(v_1, v_0)$@351@1.0
$\texttt{lnotsharerise}(v_0, v_1) \wedge \texttt{lnotbuyer}(v_1, v_0) \rightarrow \texttt{lnotsharerise}(v_1, v_0)$@351@1.0
$\texttt{lnotsharerise}(v_0, v_1) \wedge \texttt{lnotbuyer}(v_1, v_0) \rightarrow \texttt{lnotnewjobs}(v_1, v_0)$@351@1.0
$\texttt{lnotnewjobs}(v_0, v_1) \wedge \texttt{listed}(v_1, v_0) \rightarrow \texttt{listed}(v_0, v_1)$@363@1.0
$\texttt{lnotnewjobs}(v_0, v_1) \wedge \texttt{listed}(v_1, v_0) \rightarrow \texttt{lnotbuyer}(v_0, v_1)$@363@1.0
$\texttt{lnotnewjobs}(v_0, v_1) \wedge \texttt{listed}(v_1, v_0) \rightarrow \texttt{lnotbankrupt}(v_0, v_1)$@363@1.0
$\texttt{lnotnewjobs}(v_0, v_1) \wedge \texttt{listed}(v_1, v_0) \rightarrow \texttt{lnotsharerise}(v_0, v_1)$@363@1.0

In order to test the working expectation generator, it was run with a set of representative sets and the values $\alpha = 0$ and $\kappa = 0$, thus generating all possible expectation given the language. These expectations and their values were recorded on a spreadsheet and the values examined to ensure that they were correct with respect to the set of representative sets.

Those expectations whose accuracy and coverage values are greater than α and κ respectively are the expectations that should be the members of the set of working expectations according to the definitions in Chapter 5. The working expectation generator was run again with the values of α and κ and the set of working expectation generated compared with those identified manually.

In all cases (all distributions, monadic and dyadic predicates, number of degrounding symbols between 1 and 8, various cutoff points) the working expectation generator generated all and only those expectations predicted by the manual analysis. This indicates that the system functions generates the set of working expectations as defined in Chapter 5.

It was also necessary to determine that these expectations were indeed indicative of the expectations that may be held by an individual with access to the information presented in the representative sets used as input. Manual analysis was carried out to ensure that only the expectations that were truly representative of the world as described in those representative sets were included. Of course, this relies on the assumption that the expectations that have the highest accuracy and coverage are most likely to be analogues of the expectations that a well informed person may hold. See the discussion of interest in Chapter 1 for the reasons behind this assumption. To determine whether this is actually the case would require work with human subjects, something that is beyond the scope of this thesis.

6.1.5 Artificial representative sets

The simulations presented in this chapter use data from the representative set generator rather than "real" data. The first advantage of this approach is that there is no need to format the data as it is created in structured format, whereas real data would require converting to structured text format before it could be used. Secondly, the use of news reports world require the development of a set of related background facts, which is not necessary for the generation of representative sets. Thirdly, generating the data ensures that the characteristics of those data (arity, predicate distribution etc.) are well understood. Finally, the most significant advantage of using generated representative sets rather than real data is the ability to generate a range of types of representative sets reflecting universes of different complexity as defined by the language and the distribution of predicate symbols.This allows conclusions to be drawn concerning the suitability of an Expectation Violation Analysis (EVA) system as a method for spotting interesting news in a variety of domains.

The disadvantage of using simulated representative sets is that there is no indication as to whether the results obtained are representative of the results that would be obtained from real news reports. To counter this, analysis of the business news reports from the Economist and Reuters was undertaken. This analysis indicates that the results for the simulations that use representative sets generated according to the Pseudo Zipf distribution are representative of the results we would expect using news from these two sources.

In order for an EVA system to be useful it is necessary there is a subset of all possible expectations that are significantly more likely to be fired and less likely to be violated than others in the domain. This is most likely when the probability of the co-occurrence of certain predicate symbols in representative sets is high.

The following simulations demonstrate that distributions in which certain co-occurrences of certain predicate symbols are significantly more probable than others tend to result in better performance of the working expectation generator. The Pseudo Gaussian and Pseudo-Zipf probability distributions are more suited to the generation of a set of working expectations than distributions such as the Uniform distribution. This has implications for the suitability of the EVA method to different domains. Statistical analysis of news and background data can demonstrate the type of distribution of predicate symbols in that domain and, by examining which of the predicate symbol distributions examined in this chapter it most closely represents, then it is possible to decide on the suitability of the method to that domain. That news from the Reuters and Economist corpora has a Zipf-style distribution will be shown in Section 6.1.7. This demonstrates that the framework supports a practical system that can be applied to news from these sources.

6.1.6 The effect of knowledge representation on the predicate distribution

The choices made regarding knowledge representation affect the distribution of predicate symbols. If few predicates of high arity are chosen then the distribution will be closer to uniform than if predicates of a lower arity are chosen, as the following examples will demonstrate:

Let there be a report: "*Loss making dutch airline, Buzz, has recently been acquired by the profitable irish carrier, Ryanair for GBP 15.6million*" and a report "*Tobacco giant R.J. Reynolds' acquisition of cigarette manufacturer Brown and Williamson was approved by the FTC today, the 26th June, and will proceed at a cost to R.J. Reynolds of USD2.6bn*"

A language that has few predicate symbols and many terms per predicate may represent the above reports as follows:

Example 6.1.13. *Representative set*

$$\rho_1 = \{\texttt{takeover(ryanair, profitable, irish, buzz, unprofitable, dutch, null, GBP15.6m, null)}\}$$

and representative set

$$\rho_2 = \{\texttt{takeover(r.j.reynolds, null, null, brown\&williamson, null, null, 26/06/05, USD2.6bn, FTC)}\}$$

The above example results in a distribution of predicate symbols that is uniform or near uniform. This is a consequent of the fact that each report will result in one and only one predicate symbol. That predicate symbol occurs in the representative set for each report.

A language that has many predicate symbols and favours few terms in predicates may render the reports as follows:

Example 6.1.14. *Representative set*

$$\rho_1 = \{\texttt{buyer(ryanair), target(buzz), price(GBP15.6mil), profit(ryanair), } \neg\texttt{profit(buzz) country(ryanair, ireland), country(klm, netherlands)}\}$$

and representative set

$$\rho_2 = \{\texttt{buyer(r.j.reynolds)target(brown\&williamson), date(26/06/05), price(USD2.6bn), regulator(FTC)}\}$$

The renderings in Example 6.1.14 will result in more complex distributions than the rendering in 6.1.13 due to the greater number of possible permutations in the predicate symbols resulting from reports. It would, therefore, be desirable to insist on a normal form for the representation of data. This normal form should be generative, and should result in an ontology that balances simplicity (fewer predicate symbols) with discriminatory powers (more predicate symbols). This would ensure that two domains could be compared as like with like, provided the ontology for each is in the normal form.

A method of data normalisation, Boyce-Codd Normal Form (BCNF) is widely used in relational database design (for example see [AA93], pp193–208). It is a generative method and results in a compact data structure that reduces the likelihood of inconsistent data. While this method of normalisation is not designed for the purposes of designing an ontology, it does address data structure. We require a set of facts that is as compact and yet as expressive as possible and BCNF addresses this need. I will therefore suggest that this be the normalisation chosen and that the question of whether a normalisation could be devised that is better suited to the working expectation generator remains open for future work.

In order to apply BCNF to the ontology for a domain it is necessary that we consider predicate symbols to be the names of relations (database tables) and constants to be attributes of those relations. Briefly, BCNF consists of five increasingly demanding normal forms. First normal form (1NF) requires that in each relation (predicate) there be no repeating columns of data. All repeating columns must be removed to another relation. For data to be in second normal form (2NF) it must be in (1NF) and must also be free of repeated rows of data. For data to be in third normal form (3NF) it must be in 2NF and also have no value in any relation that is not dependent on the key. Fourth normal form structures are those that meet the requirement for 3NF and that also have no multivalued (many-to-many) relationships.

There exists a fifth level of normalisation that removes redundancy that is not removed by any of the previous normal forms. The fourth and fifth normal forms are rarely used.

I would suggest the use of 3NF as the normal form for the design of ontologies. It may be that there is a normal form that is more suited to the EVA framework, but developing this form is beyond the scope of this thesis.

6.1.7 Predicate symbols distributions in the Reuters and Economist corpora

A corpus of 4049 articles relating solely to business news and dating from September 1st 1996 to August 1st 1997 were selected from the Reuters news feed archive. Each of these articles dealt with a single "story", that is, there was a clear delineation of the event under discussion and other entities and events were only discussed if relevant to the central story. A corpus of

470 news digests were selected from The Economist's e-mail feed. These digests date from 29th January 2000 to 20th August 2004. Each of these digests contains several report summaries of between one and five sentences. The mean number of summaries per digest is 19.4 with a standard deviation of 4.8. Each summary relates to one story and approximately half the summaries relate to business while the other half relate to politics and world current affairs in general. Both the Reuters and the Economist feed have global coverage.

A subset of these corpora was used to determine a set of predicates required to describe the news stories. A set of 16 predicate symbols was defined according to BCNF. These predicate symbols were adequate to describe over 90% of the stories in the subsets of both corpora and a significant amount of the background information provided in the Reuters news reports [1]. An information extraction tool, BioRAT [CBLJ04] was used to extract occurrences of information that could be represented by these predicates from the Reuters and Economist corpora.

The results are shown in Figure 6.3. The distribution of the frequency of predicate symbols roughly conforms to Zipf's Law. Considered separately, both the Economist and the Reuters corpus also conform to a Zipf distribution, although the order of frequency of the predicate symbols is different, for example: in the Reuters corpus, the "Sell" predicate is ranked 8th whereas in the Economist corpus the same predicate is ranked 10th. This appears to be related to the different time frames covered by the two corpora. As the results from the simulations will show, the method of generating working expectations simulated here is highly appropriate to this distribution.

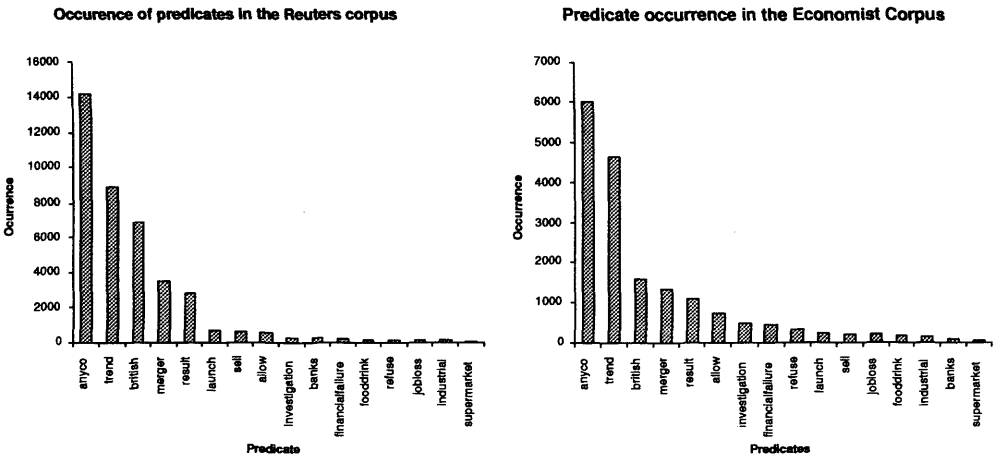


Figure 6.3: The distribution of predicates in the Reuters and Economist corpora.

[1] Because of the abridged nature of the digests, very little background information was available in the Economist corpus.

6.1.8 The effect of other distributions

Thus far it has been assumed that all constant symbols in representative sets are equiprobable. The same assumption has been made about the negation symbol. In reality it may be that the constant symbols and negation are drawn from non-uniform distributions. However, by assuming these to be uniform, the predicate symbol distribution provides a good approximation of the overall distribution of atoms as will be demonstrated below.

As the probability of an atom $\neg p_1(c_1, c_2)$ occurring is $\mathsf{prob}(p_1) \times \mathsf{prob}(\neg) \times \mathsf{prob}(c_1, c_2)$, the predicate symbol distribution alone provides a good approximation of the overall distribution of literals. The assumption is made that the probability distributions for predicate symbols, constant symbols and negation are independent of one another. The probability of each tuple of constants is constant for each predicate symbol. Likewise the probability of negation is constant for each predicate symbol.

Proposition 6.1.1. *Let the distributions of constants, predicate symbols and negation be independent. Let $P(p_1)$ be the probability of predicate symbol p_1 occurring. Let $P(p_2)$ be the probability of predicate symbol p_2 occurring. Let $P(p_1) \geq P(p_2)$. For any tuple of constants $\bar{c}$, $P(p_1(\bar{c})) \geq P(p_2(\bar{c}))$.*

Proof. Let the probability of an atom $p_1(\bar{c})$ be $P(p_1) \times P(\bar{c}) \times (1 - P(\neg))$. For the atoms $p_1(\bar{c})$ and $p_2(\bar{c})$, $P(\bar{c}) \times (1 - P(\neg)$ is constant.

Let $P(\bar{c}) \times (1 - P(\neg)) = n$. $P(p_1(\bar{c})) = P(p_1) \times n$ and $P(p_2(\bar{c})) = P(p_2) \times n$. Therefore if $P(p_1) \geq P(p_2)$, $P(p_1(\bar{c})) \geq P(p_2(\bar{c}))$. □

For any independent distributions of predicate symbols, constant symbols and negation, the distribution of atoms will be an amalgam of all three distributions.

Example 6.1.15. *Let the constant and predicate symbol distributions be independent of one another. Figure 6.4 shows the distribution of atoms where a predicate symbol distribution and a constant symbol distribution are combined. The figure on the left shows the effect of combining a Pseudo Zipf predicate distribution and a Pseudo Gaussian constant distribution, whereas the figure on the right shows the effect of combining a Pseudo Zipf predicate distribution and a Pseudo Zipf constant distribution.*

Figure 6.4 clearly demonstrates that the underlying predicate symbol distribution is a good indicator of the overall probability distribution of the atoms in the representative sets, without having the additional complexity of considering the distributions of constant or negation symbols.

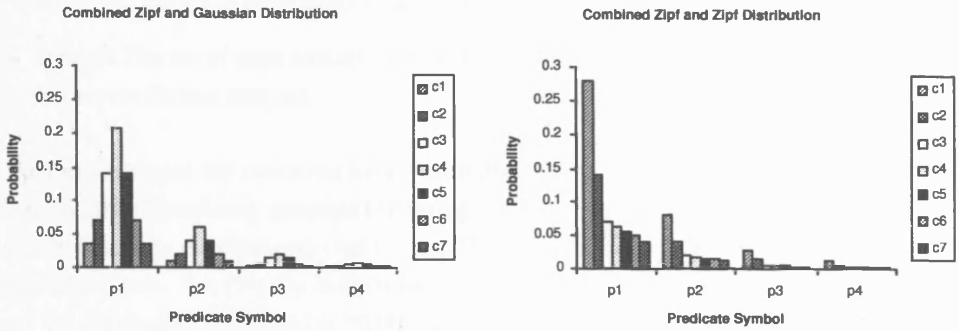


Figure 6.4: Distributions of atoms under the Zipf (predicate) and Gaussian (constant) distributions, and distributions of atoms under the Zipf (predicate) and Zipf (constant) distributions. The clusters of bars are atoms with a common predicate symbol. The bars of identical shading are atoms with common constant symbols.

6.2 Simulation 1: Measuring convergence in the set of working expectations.

This simulation demonstrates whether, given sufficient evidence, the set of working expectations for a given predicate symbol distribution and given threshold values converges to a canonical set of expectations (see Definition 6.1.4 for a definition of the canonical set).
Success condition: that the simulation shows the degree to which the set of working expectations converges for each predicate symbol distribution.

The parameters of the simulation were as follows:

- **Atoms** In order to restrict the set of possible expectations to a manageable size, the universe was restricted to 8 predicate symbols and their negations. In the first run, the language was restricted to monadic predicates and one degrounding symbol. In later runs the language was expanded to include dyadic predicates and two degrounding symbols.
- **Representative sets** The simulations were run with 10,000 representative sets and tested for convergence to a canonical set. If convergence had not occurred, then the simulation was run with 20,000 representative sets. If convergence still had not occurred then the simulation was run with 50,000 representative sets.
- **Predicate symbol distributions** There were 10 runs for each of four natural and four stepped predicate symbol distributions (See section 6.1.2 for details).
- **Threshold values** The threshold values were different for each distribution and were chosen to restrict the size of the set of working expectations to as near to 100 as possible in

order that the set of expectations should be easily analysed.

- **Output** The set of representative sets and the set(s) of working expectations were written to files for further analysis.

In order to determine the canonical set for each distribution, the working expectation generator was given 10 independently generated "training sets" of representative sets. For all distributions except the Uniform distributions, One Up and Three Up, convergence occurred at 10,000 sets of representative sets. The One Up distribution converged at 20,000 sets of representative sets, the Three Up distribution converged at 50,000 sets of representative sets. The universal distribution does not converge at 50,000 sets of representative sets and does not exhibit any tendency to converge. It is not expected that the uniform distribution would lead to convergence as no combination of n unground literals would occur significantly more frequently than any other combination of n unground literals in the output from the representative set generator.

Once the canonical sets had been obtained, the working expectation generator was then run again with 10 independently generated sets "test sets" of 10,000 representative sets for each of the non-uniform distributions. The output from the working expectation generator for each of these 10 runs was compared with the canonical set to determine the recall and precision values.

Table 6.3 gives the average recall and precision measures over the 10 runs for each distribution where representative sets consist of unary ground predicates. There is no canonical set, and therefore no recall or precision figures, for the Uniform distribution due to the non-convergence of working expectations based on representative sets with the Uniform distribution.

Distribution	Recall	Precision
One up	1.00	0.50
Three up	0.87	0.99
Big Three	1.00	1.00
Five up	1.00	1.00
Pseudo Gaussian	0.96	0.97
Flat Gaussian	1.00	0.99
Pseudo Zipf	1.00	1.00

Table 6.3: Average recall and precision results for the working expectation generator: unary predicates, 10 runs over 10 independently generated sets of 10,000 representative sets.

Recall was complete or near complete (96% or above) for all distributions except the Three Up distribution (Figure 6.3). Precision was complete or nearly complete (97% or above) for all but the One Up distribution (Figure 6.3). Therefore, for most distributions, 10,000 representative sets are sufficient to create the canonical set with a high degree of confidence.

The simulations of binary predicate representative sets were restricted to four distributions only: three up, Big Three, Pseudo Gaussian and Pseudo Zipf. The Uniform distribution was excluded as it does not converge and therefore cannot provide meaningful recall or precision values. The Three

Up and Big Three distributions were chosen as representatives of the stepped distributions and the Pseudo Zipf and Pseudo Gaussian were chosen as representatives of the natural distributions.

Distribution	Recall	Precision
Pseudo Gaussian	1.00	0.99
Three up	1.00	0.98
Big Three	1.00	1.00
Pseudo Zipf	0.98	0.92

Table 6.4: Recall and precision results for the working expectation generator: binary predicates, 10 runs over 10 independently generated sets of 10,000 representative sets.

For all four distributions, convergence occurred at 10,000 sets of representative sets. The recall results in Table 6.4 clearly show that, for all four of the above distributions, the canonical set is identifiable from a set of 10,000 representative sets. The precision results in Table 6.4 show that the majority of those expectations generated are those that are members of the canonical set.

Conclusion: This simulation demonstrates that the canonical set is identifiable and that there is a high degree of precision and recall with respect to the canonical set for most distributions. The next simulation examines whether recall and precision can be maintained with fewer representative sets provided as input.

6.3 Simulation 2: Determining the effect of using fewer representative sets

For this simulation a canonical set of expectations was created by running the working expectation generator with sufficiently large sets of representative sets to ensure convergence to a canonical set over ten runs. The purpose of the simulation is to determine how many representative sets are required for each distribution, such that the working expectation generator generates the expectations in that canonical set with precision and recall values $> 90\%$.
Success condition: the simulation will show that, for each distribution, performance degrades as the size of the set of representative sets decreases and to what degree there is a degradation.

The parameters of the simulation were as follows:

- **Atoms** The universe was restricted to 8 predicate symbols, all of which could be negated. In the first run, the language was restricted to monadic predicates and one degrounding symbol. Later the language was extended to dyadic predicates and two degrounding symbols. Even in this relatively restricted language the upper bound of $|E| \simeq 2.36 \times 10^{11}$ where antecedents and consequents are restricted to a maximum length of five conjuncts or disjuncts.

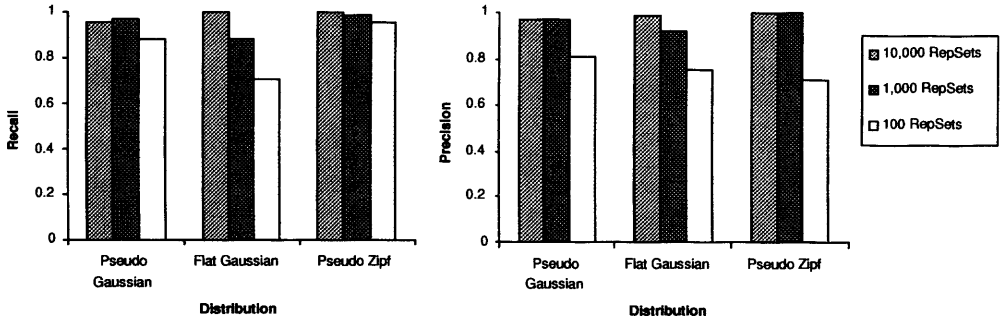
- **Canonical Set** The working expectation generator was run with increasingly large numbers of representative sets (min $R = 10,000$, max $R = 50,000$) until a cutoff point was identified that resulted in convergence on a canonical set of working expectations of cardinality of an near 100 as possible.

- **Representative sets** The simulations were run with 10,000 representative sets , 1,000 representative sets and 100 representative sets . Each simulation consisted of 10 runs using ten independently generated sets of representative sets.

- **Predicate symbol distributions** The experiments were run 10 times over each of seven predicate symbol distributions: three natural and four stepped predicate symbol distributions. See Section 6.1.2 for details of predicate symbol distributions.

- **Threshold values** Threshold values were chosen to ensure that the canonical set of working expectations should have as near to 100 members as possible.

- **Output** The set of representative sets and the set(s) of working expectations were written to comma separated value files for further processing.



	Recall at n RepSets			Precision at n RepSets		
Distribution	n=10,000	n=1,000	n=100	n=10,000	n=1,000	n=100
Pseudo Gaussian	0.96	0.97*	0.88	0.97	0.97*	0.81
Flat Gaussian	1.00	0.88	0.71	0.99	0.92	0.75
Pseudo Zipf	1.00	0.99	0.96	1.00	1.00*	0.71

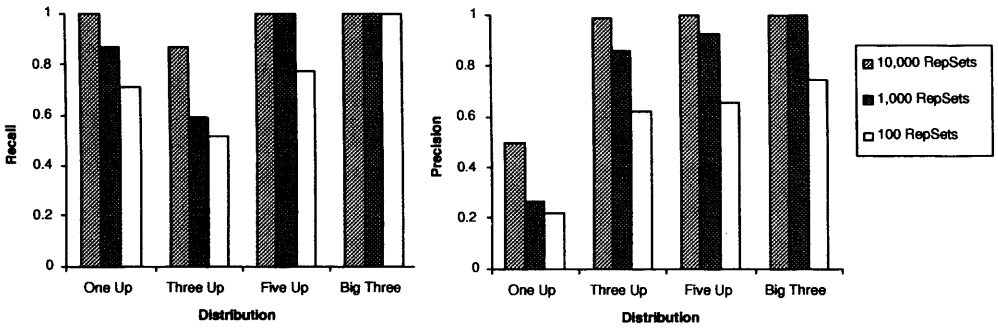
Figure 6.5: Recall and precision results for the working expectation generator: unary predicates only; natural distributions (*Change from previous value is not significant at p=0.005 when tested with the two-tailed t-test)

The results in Figure 6.5 show that the recall of working expectations generated from the set of representative sets created using the natural distributions decreases as the size of the set of representative sets decreases. The decrease in recall for the Pseudo Gaussian distribution between 10,000 sets of representative sets and 1,000 sets of representative sets is not significant at $p = 0.005$ but all other decreases in performance are significant at $p = 0.005$. Nevertheless, recall is

still high (88% for the Pseudo Gaussian distribution and 96% for the Pseudo Zipf distribution) at 100 sets of representative sets.

When provided with representative sets that are subject to the Pseudo Gaussian and the Pseudo Zipf distributions, the working expectation generator is resistant to lack of evidence. When provided with representative sets that are subject to the Flat Gaussian distribution on the other hand, the working expectation generator performs less well. At $1,000$ sets of representative sets recall is 88% but at 100 sets of representative sets recall has dropped significantly (at $p = 0.005$) to 71%. The working set generator is expected to miss many of the canonical expectations at low levels of evidence.

The results in Figure 6.5 also show the precision of working expectations generated from a set of representative sets that are subject to the natural distributions. These values also tend to decrease as the number of representative sets decreases. However, the decrease in precision for the representative sets that are subject to the Pseudo Gaussian and Pseudo Zipf distribution between 10,000 representative sets and 1,000 representative sets is not significant at $p = 0.005$. The decrease in precision for each distribution is such that at 100 representative sets precision is 81% or less. The decrease in precision between 1,000 representative sets and 100 representative sets is significant (p=0.005) for all distributions. Therefore at low levels of evidence it is not possible to place much confidence in the ability of the working expectation generator generating only canonical expectations.



	Recall at n RepSets			Precision at n RepSets		
Distribution	n=10,000	n=1,000	n=100	n=10,000	n=1,000	n=100
One up	1.00	0.87	0.71	0.50	0.26	0.22
Three up	0.87	0.59	0.52*	0.99	0.86	0.62
Five up	1.00	1.00*	0.77	1.00	0.93	0.66
Big Three	1.00	1.00*	1.00*	1.00	1.00*	0.75

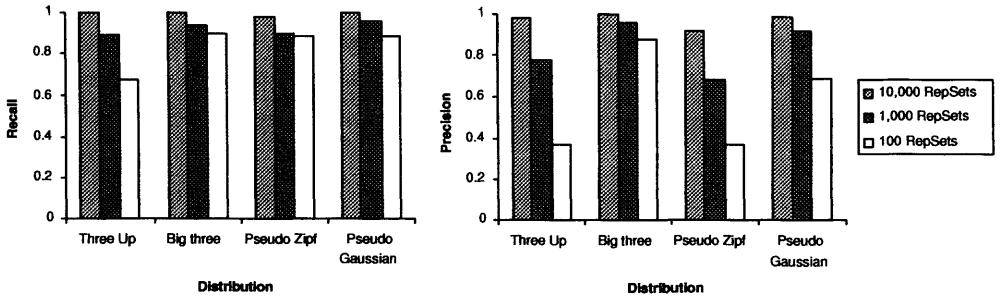
Figure 6.6: Precision and recall results for the working expectation generator: unary predicates only; stepped distributions. (* Change from previous result is not significant at $p = 0.005$ when tested with the two-tailed t-test)

The results in Figure 6.6 show that recall of working expectations generated from a set R created

using the stepped distributions tends to decrease as the number of representative sets decreases, with the exception of the Big Three distribution, whose performance remains constant with 100% recall. With input of 1,000 representative sets, the Three Up distribution is showing very poor results for recall. With input of 100 representative sets, all of the stepped distributions, with the exception of the Big Three distribution, have recall of less than 80%.

The results in Figure 6.6 also show the precision of working expectations generated from a set of representative sets created using a stepped distribution. This also decreases as the number of representative sets decreases. When the representative sets are subject to the One Up distribution, precision results are poor at all numbers of representative sets. There is a marked decrease in performance as the number of representative sets is reduced for the One Up distribution, resulting in a final precision value of 22%. The One Up and Three Up distributions also show a significant (p=0.005) decrease in precision values as the number of representative sets decreases. At 1,000 representative sets, the precision values for both the Three Up and Five Up distribution are still sufficiently high enough to be confident in the ability of the working expectation generator due to the high number of false positive results. However, with only 100 representative sets as input, the precision values for had all of the stepped distributions had fallen significantly (p=0.005) and were below the confidence threshold of 90%.

In a simple universe with unary predicates, the Pseudo-Gaussian, Pseudo Zipf, Big Three and Five Up distributions are most resistant to a lack of evidence.



	Recall at n RepSets			Precision at n RepSets		
Distribution	n=10,000	n=1,000	n=100	n=10,000	n=1,000	n=100
Three Up	1.00	0.89	0.68	0.99	0.92	0.69
Big three	1.00	0.94	0.81	1.00	0.96	0.88
Pseudo Zipf	0.98	0.90	0.88	0.92	0.68	0.37
Pseudo Gaussian	1.00	0.96	0.88	0.98	0.78	0.37

Figure 6.7: Recall and precision results for the working expectation generator: binary predicates; all distributions

Figure 6.7 shows the effect of reducing the number of representative sets on the recall and precision in a universe that contains only binary predicates. These simulations were carried out on two stepped distributions (Big Three and Three Up) to allow a comparison of the effects of large

and small differences in stepped distributions, and also on the Pseudo Gaussian and Pseudo Zipf distribution. These were felt to be sufficient to give an overall picture of the behaviour of the working expectation generator as the Pseudo Gaussian, Pseudo Zipf and Big Three distributions are the most resistant to degradation of performance, whilst the Three Up distribution is more prone to degradation and is a good representation of the stepped distributions.

Figure 6.7 shows a significant degradation in recall as the number of representative sets is reduced. However, for all distributions, 1,000 sets of representative sets results in recall ≥ 0.89. When the number of representative sets is reduced to 100, the recall for all distributions has fallen below the confidence threshold of 90%, but recall values for the natural distributions are still close to that threshold, at 88%. Therefore, even at very low levels of evidence, the working expectation generator is able to generate a useful proportion of the canonical set of expectations in universes where the predicates follow a natural distribution.

Figure 6.7 also shows a significant degradation in precision as the number of representative sets is reduced. For the Pseudo Gaussian and Big Three distributions, input of 1,000 sets of representative sets results in precision values ≥ 0.92, therefore the number of false positives is low. Input of 100 representative sets results in very low precision values for all but the Big Three distribution.

Conclusion: For most distributions there is a clear degradation in both recall and precision as the number of representative sets used as input to working expectation generator decreases. This degradation is not linear and the performance of the working expectation generator is "robust", maintaining good performance even at small numbers of representative sets under certain distributions. Of the distributions tested, the least robust performance occurs when representative sets are subject to the Three Up distribution. This can be contrasted with the relatively high levels of performance displayed when representative sets are subject to the Big Three distribution. I hypothesise that the difference is due to the difference in the variance between the levels of probability. The larger difference between the probabilities in the Big Three distribution (three predicates at P=0.3 vs five predicates at P=0.02) results in greater differences between the coverage and accuracy values of expectations than the smaller interval of the Three Up distribution (three predicates at P=0.15 vs five predicates at P=0.11). This in turn results in a canonical set that is less likely to be perturbed by random fluctuations in the occurrence of predicates symbols in representative sets.

Recall values are, in general, high even with as few as 100 representative sets. However, precision at such a low number of representative sets is very low and so using so few representative sets is undesirable if false positive results are to be minimised. At 1,000 sets of representative sets however, recall and precision results are good for all except the Three Up distribution. This demonstrates that, for a language for which the upper bound on $|E| \simeq 2.36 \times 10^{11}$, as few as 1,000 representative sets is sufficient to identify a set of working expectations that bears close relation to the canonical set of expectations. The next simulation demonstrates the effect of increasing the upper bound on $|E|$.

n degrounding symbols	Size of E
n=2	5.99×10^{10}
n=3	4.33×10^{15}
n=4	3.46×10^{19}
n=5	7.94×10^{20}
n=6	4.63×10^{22}
n=7	1.35×10^{24}
n=8	2.41×10^{25}

Table 6.5: The maximum size of E for a language with n degrounding symbols, 7 predicate symbols and a maximum unconditional formula size of 5 literals.

Recall that the news in the Reuters and the Economist corpora closely follows the Pseudo Zipf distribution. The performance of the working expectation generator is robust when presented with representative sets that are subject to the Pseudo Zipf distribution. As a result, relatively few articles from these corpora would be necessary in order to generate a set of working expectations that is representative of the canonical set of expectations.

6.4 Simulation 3: Determining the effect of a more expressive language

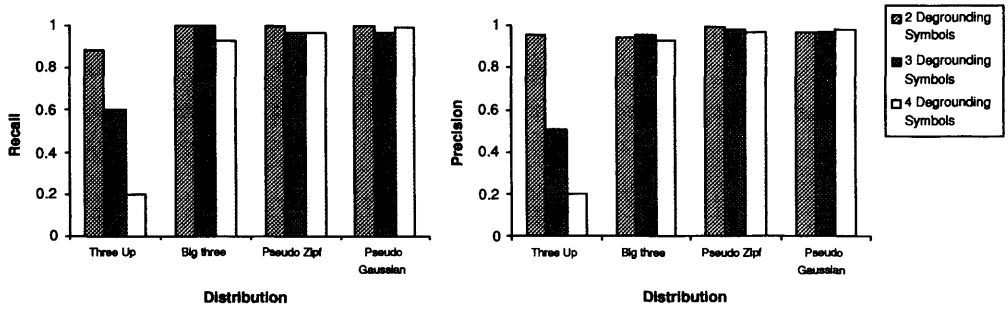
This simulation determines how the ability of the working expectation generator to generate expectations with high levels of precision and recall is affected by the complexity of the universe, as determined by the maximum size of $|E|$.
Success condition: the simulation must indicate the degradation of performance associated with the increasing complexity of the universe.

The parameters of the simulation were as follows:

- **Atoms** The universe was restricted to 7 predicate symbols and their negations. The language was restricted to dyadic predicates. Simulations were run with a maximum number of degrounding symbols in the range of 2 to 8. Table 6.5 gives the upper bounds on $|E|$ if antecedents and consequents are restricted to a maximum of five conjuncts or disjuncts in length.
- **Canonical Set** The working expectation generator was run with increasingly large numbers of representative sets (max $50,000$) until a canonical set of expectations was generated .
- **Representative sets** The simulations were run with 10 independently generated sets of 10,000 representative sets.
- **Predicate symbol distributions** The experiments were run 10 times over each of 4 pred-

icate symbol distributions: the Pseudo-Gaussian distribution and Pseudo-Zipf distribution were used to represent the natural distributions. The Three Up and Big Three distributions were used to represent the stepped predicate symbol distributions to examine the effect of different degrees of variance between probabilities.

- **Threshold values** Threshold values were chosen to ensure that the canonical set of working expectations should have as near to 100 members as possible.
- **Output** The set of representative sets and the set(s) of working expectations were written to comma separated value files for further processing.



Distribution	Recall with n Degrounding Symbols			Precision with n Degrounding Symbols		
	n=2	n=3	n=4	n=2	n=3	n=4
Three Up	0.88	0.6	0.20	0.95	0.51	0.20
Big Three	1.00	1.00*	0.93	0.94	0.95*	0.93
Pseudo Zipf	1.00	0.97	0.97*	0.99	0.98*	0.97*
Pseudo Gaussian	1.00	0.97	0.99	0.97	0.97*	0.98*

Figure 6.8: Recall and precision results for the working expectation generator: up to 4 degrounding symbols, input of 10,000 representative sets (*Change from previous value is not significant at p=0.005 when tested with the two-tailed t-test)

Figure 6.8 summarises the effect of increasing the number of degrounding symbols in the universe from two to four on recall. Once again, it is the Three Up distribution that is most prone to degradation of performance, managing recall of only 0.2 for a universe with four degrounding symbols. The Big Three distribution shows minor but significant (at p=0.005) reduction in performance for a universe with four degrounding symbols, but the recall value is still high at 0.93. The natural distributions show minor changes, but no great degradation in performance.

Figure 6.8 also summarises the effect of increasing the number of predicate symbols in the universe from two to four on precision. As with recall, it is the Three Up distribution that is most prone to degradation of performance, managing only precision of 0.2 for a universe with four degrounding symbols. The Big Three distribution shows minor, insignificant (at p=0.005) reduction in performance for a universe with four degrounding symbols, but the recall value is still

high at 0.93. The degradations in the results for the natural distributions are not significant at p=0.005. The Pseudo Gaussian distribution shows a minor increase in precision as the number of degrounding symbols rises from three to four, but this increase is insignificant at $p = 0.005$ and should most probably be regarded as a statistical anomaly.

	Recall with n Degrounding Symbols						
Distribution	n=2	n=3	n=4	n=5	n=6	n=7	n=8
Pseudo Gaussian	1.00	0.97	0.99	1.00	0.97	0.95*	0.97*
Pseudo Zipf	1.00	0.97	0.97*	0.98*	0.95	0.96*	0.96*
	Precision with n Degrounding Symbols						
Distribution	n=2	n=3	n=4	n=5	n=6	n=7	n=8
Pseudo Gaussian	0.97	0.97*	0.98*	0.98*	0.94	0.96*	0.95*
Pseudo Zipf	0.99	0.98*	0.96*	0.97*	0.96*	0.95*	0.96*

Table 6.6: Recall and precision results for the working expectation generator, two to eight degrounding symbols, input of 10,000 representative sets (*Change from previous value is not significant at $p = 0.005$ when tested with the two-tailed t-test)

Table 6.6 summarises the effect of increasing the number of degrounding symbols to eight. These simulations were run on the natural distributions only, as the performance for the Three Up distributions with > 4 degrounding symbols has degraded to such an extent that it is impossible to create a canonical set of working expectations with an input of 50,000 representative sets. Overall, recall and precision never fall below 0.94 for either of these distributions. Once again, some recall and precision values actually rise as the number of degrounding symbols increases. However, these increases are insignificant at $p = 0.005$ and should also be regarded as a statistical anomalies.These results demonstrate that, when presented with representative sets that are subject to one of the natural distributions, the working expectation generator is exceptionally robust in the face of increasing complexity.

Conclusion: The recall and precision values for three of the four distributions (Big Three, Pseudo Gaussian and Pseudo Zipf) are only slightly affected, if at all, by the increase in the number of degrounding symbols. For these distributions, increasing the complexity of the universe by increasing the number of degrounding symbols is not a significant problem. However, for the stepped distribution with the small difference in probability levels (Three Up), the degradation in performance is large and significant. Therefore not all distributions are guaranteed to be robust in the face of complex universes.

However, when input is subject to one of the non-uniform natural distributions, the performance of the working expectation generator is very robust. Even in universes with up to eight degrounding symbols, recall and precision never fall below 0.94 for both the Pseudo Gaussian and Pseudo Zipf distribution. Consequently, even for universes with a large number of degrounding symbols, a good performance can be expected under the natural distributions. Further investigation is needed to determine why the performance of the working expectation generator is more robust when presented with representative sets that are subject to the natural distributions than when

presented with representative sets that are subject to the stepped distributions. One possible explanation again takes into account the fact that the natural distributions have the greatest variation in the levels of probability (Pseudo Gaussian: max=0.341, min=0.002, Pseudo Zipf: max=0.4, min=0.01, Big Three: max=0.3, min=0.02, Three Up: max=0.15, min=0.15). The greater the variation in probability values, the greater the difference between coverage and accuracy values for expectations and therefore the easier it is for the working expectation generator to discriminate between members of the canonical set and non members of the canonical set.

6.5 Simulation 4: Favouring conjunctive expectations

This simulation evaluates the effect of favouring expectations with longer antecedents. The system is modified such that antecedents of longer length have a lower fired threshold value than those of a shorter length.
Success condition: The sensitivity of each predicate symbol distribution to changes in the threshold value that favour longer antecedents is determined.

- **Atoms** The universe was restricted to 5 predicate symbols and their negations, dyadic predicates and three degrounding symbols.
- **Canonical Set** No canonical set was required, as the object of the simulation is to identify the proportions of the sets of working expectations that are of 1 to 6 conjuncts in length.
- **Representative sets** The simulations were run with 10,000 representative sets.
- **Predicate symbol distributions** The experiments were run 10 times with input that was subject to the Pseudo-Gaussian distribution and 10 times with input that was subject to the Pseudo-Zipf distribution.
- **Threshold values** Threshold values were $\alpha = 0.9$ and $\kappa = [2, 500, 5, 000]$.
- **Output** The set of representative sets and the set(s) of working expectations were written to comma separated value files for further processing.

In previous simulations, the sets of working expectations generated consisted almost exclusively of expectations with single conjunct antecedents. No expectations were generated with antecedents of more than two conjuncts. This is because, as demonstrated in Proposition 4.2.2, the longer the antecedent, the less likely it is to be fired. Reducing the coverage threshold for expectations with longer antecedents increases the chances of those expectations being generated by the working expectation generator. Tables 6.7 to 6.10 demonstrate that the larger the reduction factor, the greater the likelihood of expectations with longer antecedents being generated, from no five-conjunct antecedents at the reduction factor of 2 to between 7% and 16% of expectations having five conjunct antecedents at a reduction factor of 10. N.B. There are no working expectations at $\kappa = 5,000$

Overall % of antecedents with	Reduction factor of n			
	n=2	n=4	n=8	n=10
1 literal	NA	0%	0%	0%
2 literals	NA	62%	19%	13%
3 literals	NA	38%	33%	37%
4 literals	NA	0%	29%	39%
5 literals	NA	0%	19%	11%

Table 6.7: Percentage of all working expectations with one to five literals in the antecedent where the reduction factor is n, $\kappa = 5,000$ and $\alpha = 0.9$. Pseudo Zipf distribution. N.B: No expectations generated at $n = 2$

with a reduction factor of 2.

Overall % of antecedents with	Reduction factor of n			
	n=2	n=4	n=8	n=10
1 literal	60%	13%	4%	2%
2 literals	40%	38%	18%	17%
3 literals	0%	35%	30%	35%
4 literals	0%	14%	36%	39%
5 literals	0%	0%	12%	7%

Table 6.8: Percentage of all working expectations with one to five literals in the antecedent where the reduction factor is n, $\kappa = 2,500$ and $\alpha = 0.9$. Pseudo Zipf distribution..

Overall % of antecedents with	Reduction factor of n			
	n=2	n=4	n=8	n=10
1 literal	100%	2%	1%	1%
2 literals	0%	60%	20%	17%
3 literals	0%	38%	31%	30%
4 literals	0%	0%	28%	36%
5 literals	0%	0%	20%	16%

Table 6.9: Percentage of all working expectations with one to five literals in the antecedent where the reduction factor is n, $\kappa = 5,000$ and $\alpha = 0.9$. Pseudo Gaussian distribution.

Conclusion: The results of this simulation demonstrate that expectations with longer antecedents can be generated by reducing the threshold for coverage for expectations with longer antecedents. The downside of this approach is that it requires a change to the algorithm used to generate the antecedents. The original algorithm exploits the monotonicity of coverage values to reduce the number of antecedents that need to be generated. Reducing the threshold for longer expectations means that the monotonicity of coverage values cannot be exploited. For a universe with 7 predicate symbols, dyadic predicates only, five degrounding symbols and a maximum antecedent

Overall % of antecedents with	Reduction factor of n			
	n=2	n=4	n=8	n=10
1 literal	64%	15%	4%	5%
2 literals	36%	44%	16%	15%
3 literals	0%	41%	31%	30%
4 literals	0%	0%	36%	40%
5 literals	0%	0%	3%	10%

Table 6.10: Percentage of all working expectations with one to five literals in the antecedent where the reduction factor is n, $\kappa = 2,500$ and $\alpha = 0.9$. Pseudo Gaussian distribution.

length of 5 conjuncts, by exploiting the monotonicity of coverage values, the minimum number of expectations that will be generated and evaluated is 78,400, whereas if monotonicity is not exploited, the minimum number of expectations that will be generated and evaluated is 2.93×10^{12} (see Table 6.5. Therefore, biasing the working expectation generator towards longer antecedents comes at the price of much higher computational demands.

6.6 Discussion of Chapter 6

This chapter presents a set of simulations and their results. Section 6.1 presents the definitions and methods used in the simulation. This section also presents the representative set generator that is used to generate the input to the simulations. These representative sets are generated, rather than being extracted from news reports, in order that the attributes of these representative sets (probability distribution of the predicate symbols, arity, number of degrounding symbols) are known and can be varied. Among the attributes to be varied are the predicate symbol probability distributions. There are eight predicate symbol probability distributions that are used in the simulations: four "natural" and four "stepped" distributions. A study of corpora of reports from the Economist and Reuters shows that the probabilities of predicate symbols in news concerning business from these sources tend to follow the Zipf distribution.

Simulation One demonstrates that given input subject to different predicate symbol distributions, the working expectation generator performs differently in terms of how well it can create sets of expectations that converge to a canonical set. The Uniform distribution for example does not converge at all. Once a canonical set has been created, input subject to the Three Up distribution leads the working expectation generator to perform badly in terms of both recall and precision. The output from the working expectation generator has poor precision when the input is subject to the One Up distribution.

Simulation One suggests that performance appears to be correlated with the spread of probability values in the predicate symbol distribution. The hypothesis that a greater range of probability

values leads to better performance is strengthened by the observation that input subject to the Big Three distribution leads to better performance than representative sets subject to the Three Up distribution. However, the Flat Gaussian distribution leads to better performance than the Pseudo Gaussian distribution, regardless of the fact that the Flat Gaussian distribution has less variation in probability values. This suggests that the spread of probability values is not the only factor that determines performance.

Simulation Two demonstrates that reducing the number of representative sets available as input to the working expectation generator has a detrimental effect on the precision and recall values for input from all distributions. As in Simulation One, all distributions are not equally affected. The Big Three, Pseudo Zipf and Pseudo Gaussian distributions in particular lead to less degradation than other distributions. This again supports the hypothesis that a large variance in probability values is helpful when determining the set of working expectations.

Simulation Three examines the effect of increasing the size of the set of expectations on the performance of the working expectation generator. Larger expectation spaces lead to some degradation in performance, which is again more noticeable in the distributions with the least variation in probability. The decline in performance is insignificant for input subject to the Pseudo Gaussian and Pseudo Zipf distributions. This suggests that, for these distributions at least, even very large expectation spaces do not present a problem.

The expectations in the set of working expectations are biased towards those with short antecedents. This is due to the monotonic relationship between the antecedent ordering and coverage values. Simulation Four demonstrates that it is possible to bias the working expectation generator towards longer antecedents, at the cost of reduced computational efficiency. Nevertheless, the working expectation generator still functions adequately in relatively small sets of expectations. Future work may determine whether there are heuristics that can create expectations with longer antecedents without sacrificing computational efficiency.

In summary, the results of these simulations show that some distributions, particularly those with highly variable probability values, lead to better results from the working expectation generator than others. The spreads of probabilities for the Economist and Reuters corpora are similar to the Pseudo Zipf distribution, which has proved particularly robust to increasing sizes of E and decreasing numbers of representative sets. It is therefore possible to be optimistic that the working expectation generator would work well on business news from these sources, as well as other news that displays a probability distribution that is widely variable.

Further work may explore several different avenues including:

Whether there is a better heuristic for generating sets of working expectations that are biased towards longer antecedents? The current method must evaluate all of the expectations that have conjunctive antecedents and single literal consequents. Even for relatively small universes, this results in a very large space to explore. There may be other, more efficient, ways of searching

these expectations.

What would be the effect of "typing" constants and variables? For example: literals with the predicate symbol `merger` could be restricted to constant symbols drawn from the set of constant symbols that denote companies, such as `merger(glaxowelcome, smithklinebeecham)`, and forbidden from accepting constant symbols that denote dates, places, people etc. By restricting constants and variables in predicates to ones of a particular type, this will reduce the size of E, by reducing the permutations of constants and variables that need be considered for each predicate symbol. Whilst adding typing information to the language would increase its complexity, would the concomitant decrease in the size of E be sufficient to justify doing so?

What is the effect of a floating threshold for accuracy and coverage to generate the working set of expectations? Is it possible, after initially polling a sample of some n expectations to determine the cutoff points for some top $m\%$ of expectations?

How many representative sets are needed to alter the members of a working set in the light of changes in the predicate symbol distribution of the representative sets? If the set of working expectations begins in one stable state (that is, fully trained to correctly represent the predicate symbol distribution of the representative sets), how many representative sets with the new predicate symbol distribution of the representative sets are needed before the set of working expectations reflects the news distribution?

These issues are beyond the scope of this thesis and they present promising avenues for further research.

Chapter 7

Narratives in EVA

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 most domains reports form stereotypical narratives where each report tells part of an ongoing story. In the mergers and acquisitions domain for example, a narrative may begin with rumours of an impending bid, continue with news of a bid being made, then go on through the negotiations until the bid is finally agreed on or rejected. All reports are part of at least one narrative and all narratives contain at least one report.

Narratives are characterised by entities and changes in the states of those entities. However, 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. The event calculus is exploited in order to capture these narratives.

Whilst some expectations are applicable to all entities in all states, others are only applicable to entities in certain states. For example, companies in the state of bankruptcy are expected not to launch takeover bids. In order to apply this expectation it is necessary to know whether an entity is in the state of being bankrupt. This requires an event model.

In order to represent and reason with narratives, it is necessary to model states and changes of state. Certain phrases are indicative of 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 modelling consists of three parts: state models, event records, and the event calculus. The result of event modelling is an event model for a domain that may be populated using information in news reports. An event model can therefore be regarded as an up-to-date repository of information about a set of stories. A given domain may have a number of different event models capturing different kinds of stories in the domain, though for simplicity, individual event models only are considered here.

There are several approaches to dealing with temporal information using logic (for a review see [MS02]). In this chapter, Section 7.1 introduces the situation calculus and the event calculus, which are two such approaches. Section 7.2.1 introduces state rules and defines a may in which these can be used to generate a state model. Section 7.3 defines events and the way in which events and state rules together result in an event model. Section 7.4 discusses the way in which the event model can be used to support the identification of interesting information within the EVA framework. Section 7.5 defines *state interpolation operators*. These are an extension to the Kowalski and Sergot event calculus that allow for the identification of unrecorded states. Section 7.6 is an extended example of an event model in the domain of business news.

7.1 The situation calculus and the event calculus

Reasoning about time and sequence using logic is used in planning and in explanation generation tasks and is also useful in the EVA framework. Knowledge of which states hold and at what time is required, as some states prevent, permit or demand certain types of behaviour from entities in a domain. For example, companies in the state of being bankrupt are prevented from launching takeover bids for other companies.

The situation calculus, introduced by McCarthy and Hayes in 1969 [MH69] is recognised as the first attempt to address reasoning about states and time using logic. This was followed by the event calculus, introduced by Kowalski and Sergot in 1985 [KS85]. The situation calculus and the event calculus are both logical formalisms for relating events, states and timeperiods. As originally defined, the situation calculus reasons about hypothetical, branching sequences of events which result in states that affect all entities. The situation calculus is a powerful method for reasoning about several possible sequences of events that could have led to a particular situation. However, the situation calculus, as originally defined, was prone to the frame problem [Mor90]. The frame problem is the difficulty of ensuring that states only change as a consequence of related events. Hanks and McDermott characterised the frame problem with the example of the Yale Shooting Problem (YSP) in [HM86]. Informally put, the YSP states that if a gun is loaded, then the gun is held for a while, then the gun is fired at Fred, that Fred should die. In the original formulation of the situation calculus it is necessary to explicitly state that holding on to the gun for a period of time does not cause the gun to become unloaded. Evidently, the need to state the effects that actions do not have quickly makes using the situation calculus cumbersome.

Kowalski and Sergot introduced the event calculus primarily as a way of avoiding the frame problem. The event calculus as originally defined dealt only with non branching sequences of events. The event calculus is also monotonic, that is, updates can only add knowledge. It is therefore possible to assimilate events into an ordered narrative, even if they are received out of chronological order. This is very useful in domains where information may emerge in an unordered way. The event calculus consists of three ontologies: event types; events and states. All events are of an event type: for example, the takeover of Safeway by Morrisons is an event of the takeover type. Events initiate and terminate states: for example, the takeover of Safeway by Morrisons initiates the state of Safeway being owned by Morrisons and terminates the state of Safeway being an independent company.

In contrast with the situation calculus, in the event calculus events, rather than timepoints, are primitives. That is to say, time frames are defined by the events that begin and end them, rather than events being defined by the times at which they occur. This is in contrast to the original definition of the situation calculus in which timepoints are primitive and situations are defined by the timepoints at which they obtain. This distinction means that the event calculus is able to reason about concurrent events and events whose timepoints are not known, whereas the situation calculus, as originally defined, does not.

Much has been made of the similarities and differences between the situation calculus and the event calculus in recent years. Both formalisms relate events to states and both can be implemented in a declarative language, such as Prolog. However, the event calculus, as originally defined, lacked the ability to reason with multiple branches of possible events. The event calculus was also prone to over-commitment because of its use of negation as failure [Rei78]. This meant that, in the event calculus, events that were unreported were deemed not to have happened. The original formulation of the situation calculus on the other hand was not appropriate for constructing narratives or for dealing with concurrent events.

Over time, work has been carried out to address the relative shortcomings of the situation calculus and the event calculus. In [Sch90], Schubert presented a monotonic solution to the frame problem in the situation calculus. In [PR93], Pinto and Reiter added the ability to construct linear paths in situation calculus models by using events as primitives (as in the event calculus). This gave the situation calculus the narrative capabilities of the event calculus. In [MS94] Miller and Shanahan presented another approach to constructing narratives whilst conserving the primacy of timepoints. As a result, two of the main drawbacks of the situation calculus were eliminated.

However, work also continued on the event calculus. In [MS96] Miller and Shanahan added the ability to reason about continuous change to the event calculus by means of incorporating axioms from the differential calculus. Because of the event calculus' ability to deal with concurrency of events, the formalism presented in [MS96] is ideal for modelling complex dynamic systems. In [Sha97], Shanahan proves the event calculus as defined in [MS96] can be used as a planner that is both sound and complete. In [MS02], Miller and Shanahan present extended definitions

of the event calculus that include the ability to reason about events with duration, time lines with explicit beginning and end points, precondition rules for events and gradual change in state. These features may be adopted to represent particular features of particular domains.

In [Pro96] Provetti adds the ability to construct branching sequences of possible events to the event calculus. Provetti demonstrates that, now that the event calculus can deal with branching sequences of events in a way that has a sound and complete Prolog implementation, the event calculus now subsumes all versions of the situation calculus. In [KS97] Kowalski and Sadri demonstrate theoretically that, when restricted to timepoints that are defined by states and events, the event calculus is more powerful than the Situation calculus.

Inarguably, the differences between the situation calculus and event calculus have been eroded and in the process, the shortcomings of both approaches have been addressed. However, as EVA does not require the ability to reason about multiple potential histories nor the ability to reason about continuous change, the original formulation of the event calculus, as defined in [KS85] will be used as a basis for the event model in the EVA framework. This formalism has a simple, clear semantics and is easily implemented in Prolog. However, the commitments made by the use of the Negation as Failure operator are too strong for the proposes of the EVA framework. Therefore, a modification is presented here that allows the event model to interpolate missing events where these have gone unreported.

7.2 The event calculus and the EVA framework

The EVA framework proposed in this thesis enables the identification of interesting news by looking for information that is inconsistent with expectations. Some of these expectations are only applicable to entities in certain states. For example, the expectation that company X will float on the stock exchange is applicable only to companies that are not already listed on the stock exchange.

The original form of the event calculus, as presented in [KS85] is ideal for reasoning about states, events and entities. The event calculus defines rules and metapredicates that can be used to build up models of series of events. There are three ontologies in the event calculus: event types, events and states. Definitions of these ontologies will be presented in this section.

Section 7.2.1 demonstrates that the set of event types can define all possible state transitions for a domain. The event type definition is supplemented in this thesis with a state model. A state model is a directed graph whose nodes correspond to states and whose arcs correspond to events. The relations between nodes and arcs are defined by the event types. The state model is therefore the definition of all possible sequences of events.

7.2.1 State rules and state models

In the original definition of the event calculus, presented in [KS85], time periods are identified by the events that begin and end them. For each domain, it is possible to define a set of rules in which the head has the predicate symbol *initiates* or *terminates*. These rules define the relationships between events and states for each domain.

Definition 7.2.1. *Let states be a set of unground literals, events be a set of labels that uniquely identify individual events. Let $s \in$ states and $e \in$ events. Let α be an unconditional formula and $\bar{x}$ be a tuple of the variables in s and α.*

$\forall \bar{x}\ \alpha \rightarrow \mathsf{initiates}(e, s)$ *is an initiates rule and* $\forall \bar{x}\ \alpha \rightarrow \mathsf{terminates}(e, s)$ *is a terminates rule.*

Example 7.2.1. *The rule, "A company announcing its bankruptcy initiates a state where that company is bankrupt" could be represented as:*

$$\forall \mathtt{c} \in \mathtt{companies},\ \mathtt{e} \in \mathtt{events},\ \mathtt{act(e, bankruptcy)} \wedge \mathtt{object(e, c)} \rightarrow \mathsf{initiates}(\mathtt{e}, \mathtt{bankrupt(c)})$$

Example 7.2.2. *The rule, "An enterprise is no longer state-owned once it has been privatised" can be represented as:*

$$\forall \mathtt{c} \in \mathtt{companies},\ \mathtt{e} \in \mathtt{events}\ \mathtt{act(e, privatised)} \wedge \mathtt{object(e, c)} \rightarrow \mathsf{terminates}(\mathtt{e}, \mathtt{stateowned(c)})$$

In [KS85], Kowalski and Sergot also define general rules that apply to all domains. These rules make it possible to determine, in relation to an event, whether a state holds.

Definition 7.2.2. *Let e be an event label and s be a state. The general rule "State s holds after event e if e initiates s" is:*

$$\forall e \in \mathit{events},\ s \in \mathit{states},\ \mathsf{initiates}(e, s) \rightarrow \mathsf{holds}(\mathsf{after}(e, s))$$

The general rule "State s holds before event e if e terminates s" is:

$$\forall e \in \mathit{events},\ s \in \mathit{states},\ \mathsf{terminates}(e, s) \rightarrow \mathsf{holds}(\mathsf{before}(e, s))$$

Example 7.2.3. *That a company is bankrupt after the event of announcing its bankruptcy would be rendered:*

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c} \in \mathtt{companies}\ \mathsf{initiates}(\mathtt{e}, \mathtt{bankrupt(c)}) \rightarrow \mathsf{holds}(\mathsf{after}(\mathtt{e}, \mathtt{bankrupt(c)}))$$

That a company is state owned before the event of its privatisation would be rendered:

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c} \in \mathtt{companies}\ \mathsf{terminates}(\mathtt{e}, \mathtt{stateowned(c)}) \rightarrow \mathsf{holds}(\mathsf{before}(\mathtt{e}, \mathtt{stateowned(c)}))$$

It is of course possible to define the rules that allow us to derive holds(after(e, s)) and holds(before(e, s)) directly from the definitions of the events:

Example 7.2.4. *A direct definition of the rule with the head* holds(before($e, stateowned(c)$)) *would be*

$$\forall \mathtt{c} \in \mathtt{companies},\ \mathtt{e} \in \mathtt{events}\ \mathtt{act(e, privatised)} \wedge \mathtt{object(e, c)} \rightarrow \mathsf{holds}(\mathsf{before}(\mathtt{e}, \mathtt{stateowned(c)}))$$

However, the indirect definition provides a set of general rules that are compact and are applicable to any domain. For these reasons, Kowalski and Sergot separate the domain specific *initiates* and *terminates* rules from the general *holds before* and *holds after* rules.

Kowalski and Sergot define *start* and *end* predicates that determine the events that stare and end each period, where a period is a continuous state.

Definition 7.2.3. *Let s be a state and e an event. That the period* after(e, s) *is begun by event e is formally defined as:*

$$\forall e \in \mathit{events}\ \mathsf{start}(\mathsf{after}(e, s), e)$$

That the period after(e, s) *is ended by e' is formally defined as:*

$$\forall e \in \mathit{events}\ \mathsf{after}(e, s) = \mathsf{before}(e', s) \rightarrow \mathsf{end}(\mathsf{after}(e, s), e')$$

That the period before(e', s) *is begun by event e is formally defined as:*

$$\forall e \in \mathit{events}\ \mathsf{before}(e', s) = \mathsf{after}(e, s) \rightarrow \mathsf{start}(\mathsf{before}(e', s), e)$$

That the period before(e', s) *is ended by the event e' is defined as:*

$$\forall e \in \mathit{events}\ \mathsf{end}(\mathsf{before}(e', s), e')$$

in [KS85], Kowalski and Sergot also define a set of general, domain independent rules that determine which states hold at a given timepoint. In order to do so, the time periods after(e, s) and before(e', s) define a time period during which state s holds and which is bounded by events e and e'. These rules determine whether two events begin and end the same time period.

It is first necessary to determine whether there is some event that interrupts the state between those two events. An event interrupts a state, s if it begins another state s' that is exclusive of s. s is exclusive of s' if s is incompatible with s' or if s is equal to s'. s is incompatible with s' if s and s' cannot hold simultaneously. Incompatibility is defined by rules for specific domains.

Example 7.2.5. $\forall \mathtt{x}, \mathtt{y}, \mathtt{z} \in \mathtt{companies}\ \mathtt{y} \neq \mathtt{z} \rightarrow \mathsf{incompatible}(\mathtt{owns(y, x)}, \mathtt{owns(z, x)})$

Definition 7.2.4. *For any states, s and s', those states are exclusive if they are identical or incompatible:*

$$\mathsf{exclusive}(s, s')$$
$$\mathsf{incompatible}(s, s') \rightarrow \mathsf{exclusive}(s, s')$$

A state s is broken by an event that begins a state s' that is exclusive to s:

Definition 7.2.5. *For all events e, e', e*, and all states s and s*. Let time(e) be the timepoint at which event e occurred.*

$\mathsf{broken}(e, s, e')$ *if:*
$\mathsf{holds}(\mathsf{after}(e^*, s^*))$ *and*
$\mathsf{exclusive}(s, s^*)$ *and*
$time(e) < time(e^*) < time(e')$

$\mathsf{broken}(e, s, e')$ *if:*
$\mathsf{holds}(\mathsf{before}(e^*, s^*))$ *and*
$\mathsf{exclusive}(s, s^*)$ *and*
$time(e) < time(e^*) < time(e')$

The state s, begun by event e is the same state s, ended by event e' if e happened before e' and the period between e and e' is unbroken:

Definition 7.2.6. *Let s be a state and let e be an event. The state* $\mathsf{after}(e, s) = \mathsf{before}(e', s)$ *iff*

$\mathsf{holds}(\mathsf{after}(e, s))$ *and*
$\mathsf{holds}(\mathsf{before}(e', s))$ *and*
$time(e) < time(e')$ *and*
not $\mathsf{broken}(e, s, e')$

Kowalski and Sergot define a set of meta predicates that enables us to define events that begin and end states. These can then be used to construct abstract narratives. These allow us know what sequences of events and states are possible.

7.2.2 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. This model is based on the initiates and terminates rules as defined in Definition 7.2.1.

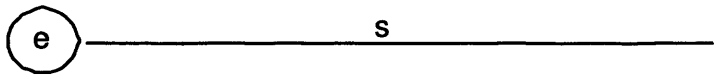


Figure 7.1: The graphical representation of the rule $e \rightarrow \mathsf{initiates}(e, s)$.

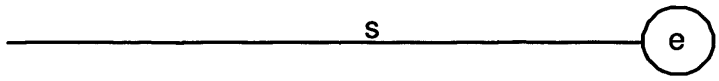


Figure 7.2: The graphical representation of the rule $e \rightarrow \mathsf{terminates}(e, s)$.

Definition 7.2.7. *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. 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 is assumed to exhaustively define all possible orderings of events and, by extension, states.

In [KS85], Kowalski and Sergot graphically represent states and events. Arcs represent states and nodes represent events. Only those events that have been reported and the states that can be inferred from those events are represented. However, it is possible to take the initiates and terminates rules in definition 7.2.1 and use them to graphically represent hypothetical sequences of events and states:

Definition 7.2.8. *For all initiates rules such that e is an event, s is a state and* $e \rightarrow \mathsf{initiates}(e, s)$*, the graphical representation of this rule is given in Figure 7.1. For all terminates rules such that e is an event, s is a state and* $e \rightarrow \mathsf{terminates}(e, s)$*, the graphical representation of this rule is given in Figure 7.2.*

Definition 7.2.9. *Let e, e' be events and s be a state. If* $\mathsf{after}(e, s) = \mathsf{before}(e', s)$ *then the nodes e and e' are nodes that are joined by the arc s. The graphical representation of this rule is given in Figure 7.3.*

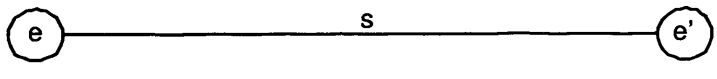


Figure 7.3: The graphical representation of the rule $e \rightarrow \mathsf{initiates}(e, s)$ and $e' \rightarrow \mathsf{terminates}(e', s)$ and $\mathsf{after}(e, s) = \mathsf{before}(e', s)$.

Whereas Kowalski and Sergot apply this graphical representation to reported events and the derived states only, the following example demonstrates how such graphical representations can be derived from the initiates and terminates rules to represent all possible sequences of events and states.

Example 7.2.6. *Let the following be a set of simplified initiates and terminates rules for the mergers and acquisitions domain. N.B: the subscripts on the event identifier variables are included for the purposes of clarity in the graphical representation. These variables should not be mistaken for constant symbols.*

$$\forall e_1 \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e_1, \texttt{bidFor}) \wedge \texttt{buyer}(e_1, c_1) \wedge \texttt{target}(e_1, c_2) \rightarrow \texttt{initiates}(e_1, \texttt{bidFor}(c_1, c_2))$$

$$\forall e_2 \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e_2, \texttt{approvesBid}) \wedge \texttt{buyer}(e_2, c_1) \wedge \texttt{target}(e_2, c_2) \wedge \texttt{body}(e_2, \texttt{board}) \rightarrow \texttt{terminates}(e_2, \texttt{bidFor}(c_1, c_2))$$

$$\forall e_2 \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e_2, \texttt{approvesBid}) \wedge \texttt{buyer}(e_2, c_1) \wedge \texttt{target}(e_2, c_2) \wedge \texttt{body}(e_2, \texttt{board}) \rightarrow \texttt{initiates}(e_2, \texttt{takeover}(c_1, c_2))$$

$$\forall e_3 \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e_3, \texttt{rejectsBid}) \wedge \texttt{buyer}(e_3, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e_3, \texttt{board}) \rightarrow \texttt{terminates}(e_3, \texttt{bidfor}(c_1, c_2))$$

$$\forall e_4 \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{rejectsBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{regulator}) \rightarrow \texttt{terminates}(e, \texttt{takeover}(c_1, c_2))$$

$$\forall e_5 \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{regulator}) \rightarrow \texttt{terminates}(e, \texttt{takeover}(c_1, c_2))$$

$$\forall e_5 \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{regulator}) \rightarrow \texttt{initiates}(e, \texttt{owns}(c_1, c_2))$$

Figure 7.4 represents the state model that can be derived from these rules.

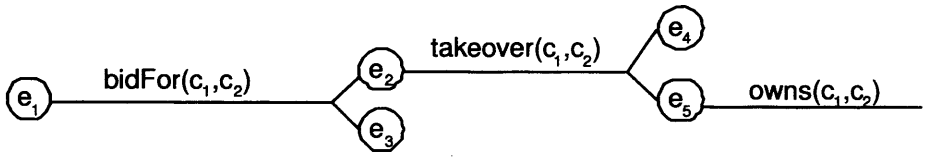


Figure 7.4: A simplified example of a state model for the mergers and acquisitions domain based on the rules in Example 7.2.6

7.3 Events and the event model

An event model is a set of object-level and meta-level formulae that is the union of the set of state rules and a set of event records. Event records are a repository of information obtained from news reports. As each news report is applied to the sets of access rules and event rules, the set of event records is increased. State rules are then used to create an event model that relates reported events to states. Each event model is self-contained. This means each event model can be regarded as a separate module.

Based on this modularity, the system may have more than one event model based on different state rules. Furthermore, the underlying version of event calculus can differ.

An event record is a set of facts that share a unique event identifier and, together, describe a reported event. An event rule is a rule that constructs an event record from the facts extracted from a news report by an access rule.

Definition 7.3.1. *Let α and β be conjunctions of predicates. Let $\bar{x}$ be a tuple of the variables in α. Let e be a variable that is a place holder for a unique event identifier and let p be a predicate symbol. An event rule is a rule such that if $p(x_1)$ is a conjunct in α then $p(e, x_1)$ is a conjunct in β. $\forall \bar{x} \exists e\ \alpha \rightarrow \beta$. is an event rule*

Example 7.3.1. *The following is a event rule:*

$$\forall \mathtt{b}, \mathtt{t}, \mathtt{v}\ \exists \mathtt{e}\ \mathtt{buyer(b)} \wedge \mathtt{target(t)} \wedge \mathtt{act(launchBid)} \wedge \mathtt{value(v)} \rightarrow \mathtt{buyer(e,b)} \wedge \mathtt{target(e,t)} \wedge \mathtt{act(e,launchBid)} \wedge \mathtt{value(e,v)}$$

The rule sates that for any event in which a buyer, b launches a bid of value v for target t there is an event identifier, e, which is used as a unique identifier in an event record.

Event rules are then used to generate event records, which are sets of ground predicates, united by a unique identifier, that describe an event:

Definition 7.3.2. *Let repSet be a representative set and Σ be a set of event rules. An event record*

for repSet with respect to Σ, *denoted* $\mathsf{event}(repSet, \Sigma)$, *is as follows:*

$$\begin{aligned}\mathsf{event}(repSet, \Sigma) = \{&\mathsf{ground}(\alpha(e, x_1, .., x_k), \Phi) \mid \textit{there exists a grounding set } \Phi \textit{ s.t}\\ &\mathsf{ground}(\alpha(x_1, .., x_k), \Phi) \in repSet\\ &\textit{and } \Sigma \vdash \forall x_1, .., x_n \; \exists e;\; \alpha(x_1, .., x_n) \rightarrow \alpha(e, x_1, .., x_n)\\ &\textit{and } e \textit{ is a unique event identifier}\}\end{aligned}$$

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

Example 7.3.2. *Let*

$$\begin{aligned}\forall \mathtt{b}, \mathtt{t}, \mathtt{v}\; \exists \mathtt{e}\; &\mathtt{buyer(b)} \wedge \mathtt{target(t)} \wedge \mathtt{act(launchBid)} \wedge \mathtt{value(v)} \rightarrow\\ &\mathtt{buyer(e, b)} \wedge \mathtt{target(e, t)} \wedge \mathtt{act(e, launchBid)} \wedge \mathtt{value(e, v)}\end{aligned}$$

in Σ *and* $\mathtt{repSet} = \{\mathtt{buyer(ryanair)}, \mathtt{target(buzz)}, \mathtt{act(launchBid)}, \mathtt{value(gbp15m)}\}$ *and* $\Phi = [\mathtt{b} = \mathtt{ryanair}, \mathtt{t} = \mathtt{buzz}, \mathtt{v} = \mathtt{gbp15m}]$. *Let* $\mathtt{e}_1$ *be an event identifier that is not used in any other event record.* $\mathsf{event}(\mathtt{repSet}, \Sigma)$=

$$\begin{aligned}\{&\mathtt{buyer}(\mathtt{e}_1, \mathtt{ryanair}),\\ &\mathtt{target}(\mathtt{e}_1, \mathtt{buzz})\\ &\mathtt{act}(\mathtt{e}_1, \mathtt{launchBid})\\ &\mathtt{value}(\mathtt{e}_1, \mathtt{gbp15m})\}\end{aligned}$$

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.

7.3.1 Combining events records with the state model

The event records extracted from representative sets by means of the set of event rules and the Event operator can then be applied to the *initiates* and *terminates* rules, as defined in Definition 7.2.1. It is then possible to determine the states that are begun or ended by that event.

Example 7.3.3. *Given the initiates rule*

$$\forall \mathtt{c}, \mathtt{e}\; \mathtt{act(e, bankruptcy)} \wedge \mathtt{object(e, c)} \rightarrow \mathsf{initiates}(\mathtt{e}, \mathtt{bankrupt(c)})$$

and the representative set

$$\{\mathtt{act(bankruptcy)}, \mathtt{object(USAirlines)}$$

and the event rule

$$\forall \mathtt{c}\; \exists \mathtt{e}\; \mathtt{act(bankruptcy)} \wedge \mathtt{object(c)} \rightarrow \mathtt{act(e, bankruptcy)} \wedge \mathtt{object(e, c)}$$

results in the set

$$\mathsf{event}(\mathtt{repSet}, \Sigma) = \{\mathtt{act}(\mathtt{e}_1, \mathtt{bankruptcy}), \mathtt{object}(\mathtt{e}_1, \mathtt{USAirlines})\}$$

This event record, together with the initiates rule leads to the conclusion that

$$\mathsf{holds}(\mathsf{after}(\mathtt{e}_1, \mathtt{bankrupt}(\mathtt{USAirlines})))$$

7.4 Using an event model

In [KS85] Kowalski and Sergot define the *holdsAt* meta predicate that can be used to determine which states hold at which time points.

Definition 7.4.1. *Let t be a timepoint of any duration and p be a period of duration greater or equal to t.*

$$\begin{aligned} &t \textit{ in } p \textit{ iff:} \\ &\qquad \mathsf{start}(p, e_1) \textit{ and } \mathsf{end}(p, e_2) \textit{ and} \\ &\qquad \textit{time}(e_1, t_1) \textit{ and } \textit{time}(e_2, t_2) \textit{ and} \\ &\qquad t_1 < t < t_2 \end{aligned}$$

Knowing the periods during which states hold allows us to determine whether a time point falls within that period and therefore whether a given state holds at a given timepoint.

Definition 7.4.2. *Let s be a state, e be an event and t be a timepoint. That state s holds at timepoint t (denoted* $\mathsf{holdsAt}(s, t)$*) is determined by the following rules:*

$$\mathsf{holds}(\mathsf{after}(e, s)) \wedge t \textit{ in } \mathsf{after}(e, s) \rightarrow \mathsf{holdsAt}(s, t)$$

and

$$\mathsf{holds}(\mathsf{before}(e, s)) \wedge t \textit{ in } \mathsf{before}(e, s) \rightarrow \mathsf{holdsAt}(s, t)$$

Example 7.4.1. *Let*

$$\{\mathtt{act}(\mathtt{e}_1, \mathtt{bidFor}), \mathtt{buyer}(\mathtt{e}_1, \mathtt{bmw}), \mathtt{target}(\mathtt{e}_1, \mathtt{rover})\}$$

$$\{\mathtt{act}(\mathtt{e}_2, \mathtt{approvesBid}), \mathtt{buyer}(\mathtt{e}_2, \mathtt{bmw}), \mathtt{target}(\mathtt{e}_2, \mathtt{rover}), \mathtt{body}(\mathtt{e}_2, \mathtt{shareholders})\}$$

be event records.

Let

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c_1}, \mathtt{c_2} \in \mathtt{companies}\ \mathtt{act(e, bidFor)} \wedge \mathtt{buyer(e, c_1)} \wedge \mathtt{target(e, c_2)} \rightarrow \mathsf{initiates}(\mathtt{e}, \mathtt{bidFor(c_1, c_2)})$$

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c_1}, \mathtt{c_2} \in \mathtt{companies}\ \mathtt{act(e, approvesBid)} \wedge \mathtt{buyer(e, c_1)} \wedge \mathtt{target(e, c_2)} \wedge \mathtt{body(e, board)} \rightarrow \mathsf{terminates}(\mathtt{e}, \mathtt{bidFor(c_1, c_2)})$$

be initiates and terminates rules.

Let the time of $\mathtt{e_1}$ *be 1st December 1993 and the time of* $\mathtt{e_2}$ *be 13th February 1994. Let* $\mathtt{t}$ *be 12th January 1994. By definition 7.4.1,* $\mathtt{t}$ *is in the period begun by* $\mathtt{e_1}$ *and ended by* $\mathtt{e_2}$*. The initiates and terminates rules, together with the general* holds *rule, determine that*

$$\mathsf{holds}(\mathsf{after}(\mathtt{e_1}, \mathtt{bidFor(bmw, rover)}) \wedge \mathsf{holds}(\mathsf{before}(\mathtt{e_2}, \mathtt{bidFor(bmw, rover)}))$$

therefore, by definition 7.4.2, at time $\mathtt{t}$, $\mathsf{holdsAt}(\mathtt{t}, \mathtt{bidFor(bmw, rover)})$

The definition of representatives(ρ, Γ, Δ) can now be modified in order to explicitly include facts from the event model that were preciously held to be implicit in Δ:

Definition 7.4.3. *Let* ρ *be a report,* Γ *be a set of access rules and* Σ *be an event model. The state that holds at the time of the event reported in* ρ *are:*

$$\mathsf{states}(\rho, \Gamma, \Sigma) = \{s | \Sigma \vdash \mathsf{holdsAt}(t, s)\ \textit{and}\ \textit{time}(t) \in \textit{access}(\rho, \Gamma)\}$$

Let Δ *be a set of background facts. A representative set for* ρ, Γ, Σ *and* Δ *is:*

$$\mathsf{representatives}(\rho, \Gamma, \Delta) = \mathsf{access}(\rho, \Gamma) \cup \mathsf{match}(\rho, \Gamma, \Delta) \cup \mathsf{states}(\rho, \Gamma, \Sigma)$$

7.5 State interpolation

The above definitions are sufficient when information about events is complete, which is the assumption that Kowalski and Sergot made when devising these predicates in [KS85]. In the domain of news reports however there will be times when there is no recorded initiating or terminating event for a state. This leads to difficulties as the following example will demonstrate:

Example 7.5.1. *Let*

$$\{\mathtt{act(e_1, bidFor)}, \mathtt{buyer(e_1, ryanair)}, \mathtt{target(e_1, buzz)}\}$$

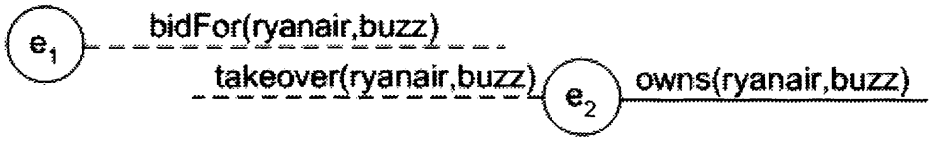


Figure 7.5: The states and events as described in example 7.5.1

and

$$\{\texttt{act}(e_2, \texttt{approvedBid}), \texttt{buyer}(e_2, \texttt{ryanair}), \texttt{target}(e_2, \texttt{buzz}), \texttt{body}(e_2, \texttt{shareholders})\}$$

be event records, and the set of initiates and terminates rules be:

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{bidFor}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \rightarrow \texttt{initiates}(e, \texttt{bidFor}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{board}) \rightarrow \texttt{terminates}(e, \texttt{bidFor}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{board}) \rightarrow \texttt{initiates}(e, \texttt{takeover}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{rejectsBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{board}) \rightarrow \texttt{terminates}(e, \texttt{bidfor}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{rejectsBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{regulator}) \rightarrow \texttt{terminates}(e, \texttt{takeover}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{regulator}) \rightarrow \texttt{terminates}(e, \texttt{takeover}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{regulator}) \rightarrow \texttt{initiates}(e, \texttt{owns}(c_1, c_2))$$

Therefore the sequence of states initiated and terminated by e_1 and e_2 is as pictured in figure 7.5.

The problem with this example is as follows: in definition 7.4.1, in order to determine the period

in which a time point t falls, that period must be explicitly begun and ended by a pair of events. However, in Example 7.5.1 two states are begun, one state is ended but no state is both begun and ended. Therefore it is not possible to conclude that a time point t falls in any particular period.

This difficulty can be partially overcome by identifying those states that, while not explicitly terminated, can be deduced to be terminated by the existence of later states.

In Example 7.5.1, there is no event that explicitly terminates the state `bidFor(ryanair, buzz)`. However, there is an event that terminates the state `takeover(ryanair, buzz)`. We can see from the state model that the state `takeover(ryanair, buzz)` could only have been initiated by an event that simultaneously terminates `bidFor(ryanair, buzz)`. Therefore between $\mathtt{e}_1$ and $\mathtt{e}_2$ there must be at least one event $\mathtt{e}'$ that initiates `takeover(ryanair, buzz)` and simultaneously terminates `bidFor(ryanair, buzz)`.

We have a state model that we know to be a complete description of the possible sequences of states and events for a domain. Therefore we can exploit this to identify intervening events that are unreported. This means that we can alter the definition of *start* and *end* given in Definition 7.2.3, in order to take into account the existence of unreported states.

Firstly we require two new definitions:*precedes* and *follows*.

Definition 7.5.1. *Let e be an event and s be a state.*

$$\begin{aligned}&\mathsf{precedes}(e, s) \textit{ iff.}\\ &\qquad \textit{there exists an initiates rule such that } \mathsf{initiates}(e, s) \textit{ or}\\ &\qquad \textit{there exists a state } s' \textit{ such that } \mathsf{initiates}(e, s') \textit{ and}\\ &\qquad\qquad \mathsf{terminates}(e', s') \textit{ and } \mathsf{precedes}(e', s)\end{aligned}$$

The terminates and precedes rules embody the canonical sequences of events as derived from state models such as those in Figure 7.6. By querying the rules that describe state models we know what hypothetical states may exist between to events that do not begin and end the same state. State models provide us with exhaustive lists of the potential events and states for a given narrative and may be generated manually by a domain expert. Alternatively it may be possible that state models be derived from news reports by machine learning, but if some states characteristically go unreported, this may not be a viable solution.

Example 7.5.2. *Let*

$$\{\mathtt{act}(\mathtt{e}_1, \mathtt{bidFor}), \mathtt{buyer}(\mathtt{e}_1, \mathtt{ryanair}), \mathtt{target}(\mathtt{e}_1, \mathtt{buzz})\}$$

and

$$\{\mathtt{act}(\mathtt{e}_2, \mathtt{approvedBid}), \mathtt{buyer}(\mathtt{e}_2, \mathtt{ryanair}), \mathtt{target}(\mathtt{e}_2, \mathtt{buzz}), \mathtt{body}(\mathtt{e}_2, \mathtt{shareholders})\}$$

be event records and the set of initiates and terminates rules be

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{bidFor}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \rightarrow \text{initiates}(e, \texttt{bidFor}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{board}) \rightarrow \text{terminates}(e, \texttt{bidFor}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{board}) \rightarrow \text{initiates}(e, \texttt{takeover}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{rejectsBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{board}) \rightarrow \text{terminates}(e, \texttt{bidfor}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{rejectsBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{regulator}) \rightarrow \text{terminates}(e, \texttt{takeover}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{regulator}) \rightarrow \text{terminates}(e, \texttt{takeover}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{regulator}) \rightarrow \text{initiates}(e, \texttt{owns}(c_1, c_2))$$

To determine $\text{precedes}(e_1, \texttt{owns}(\texttt{ryanair}, \texttt{buzz}))$*:* e_1 *does not initiate* $\texttt{owns}(\texttt{ryanair}, \texttt{buzz})$*. There is however a state* s' *such that an event* e *could initiate:* $\texttt{bidFor}(\texttt{ryanair}, \texttt{buzz})$*. Let* e' *terminate* $\texttt{bidFor}(\texttt{ryanair}, \texttt{buzz})$*. If this were so,* e' *would have to be*

$$\{\texttt{act}(e', \texttt{rejectsBid}), \texttt{buyer}(e', \texttt{ryanair}), \texttt{target}(e, \texttt{buzz}), \texttt{body}(e', \texttt{shareholders})\}$$

or

$$\{\texttt{act}(e', \texttt{acceptsBid}), \texttt{buyer}(e', \texttt{ryanair}), \texttt{target}(e, \texttt{buzz}), \texttt{body}(e', \texttt{shareholders})\}$$

If $e' = \{\texttt{act}(e', \texttt{acceptsBid}), \texttt{buyer}(e', \texttt{ryanair}), \texttt{target}(e, \texttt{buzz}), \texttt{body}(e', \texttt{board})\}$*, does it follow that* $\text{precedes}(e', \texttt{owns}(\texttt{ryanair}, \texttt{buzz}))$*?*

e′ does not initiate owns(ryanair, buzz). *However* e′ *may initiate* takeover(ryanair, buzz). *Let* e″ *terminate* takeover(ryanair, buzz). *If this were so,* e″ *would have to be*

$$\{\texttt{act}(e'', \texttt{rejectsBid}), \texttt{buyer}(e'', \texttt{ryanair}), \texttt{target}(e, \texttt{buzz}), \texttt{body}(e'', \texttt{board})\}$$

or

$$\{\texttt{act}(e'', \texttt{acceptsBid}), \texttt{buyer}(e'', \texttt{ryanair}), \texttt{target}(e, \texttt{buzz}), \texttt{body}(e'', \texttt{board})\}$$

If $e'' = \{\texttt{act}(e'', \texttt{acceptsBid}), \texttt{buyer}(e'', \texttt{ryanair}), \texttt{target}(e, \texttt{buzz}), \texttt{body}(e'', \texttt{board})\}$, *does* precedes(e″, owns(ryanair, buzz)) *hold?*

e″ *initiates* owns(ryanair, buzz), *therefore* precedes(*e″*, owns(ryanair, buzz)), precedes(e′, owns(ryanair, buzz)) *and* precedes(e_1, owns(ryanair, buzz))

Definition 7.5.2. *Let e be an event and let s be a state.*

follows(e, s) *iff.*
there exists an terminates rule such that terminates(e, s) *or*
there exists a state s' such that terminates(e, s') *and*
initiates(e', s') *and* follows(e', s)

Example 7.5.3. *Let*

$$\{\texttt{act}(e_1, \texttt{bidFor}), \texttt{buyer}(e_1, \texttt{ryanair}), \texttt{target}(e_1, \texttt{buzz})\}$$

and

$$\{\texttt{act}(e_2, \texttt{approvedBid}), \texttt{buyer}(e_2, \texttt{ryanair}), \texttt{target}(e_2, \texttt{buzz}), \texttt{body}(e_2, \texttt{shareholders})\}$$

be event records and the set of initiates and terminates rules be

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{bidFor}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \rightarrow \text{initiates}(e, \texttt{bidFor}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{board}) \rightarrow \text{terminates}(e, \texttt{bidFor}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{board}) \rightarrow \text{initiates}(e, \texttt{takeover}(c_1, c_2))$$

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c_1}, \mathtt{c_2} \in \mathtt{companies}\ \mathtt{act(e, rejectsBid)} \wedge \mathtt{buyer(e, c_1)} \wedge \mathtt{target(e, c_2)} \wedge \mathtt{body(e, board)} \rightarrow \mathsf{terminates}(\mathtt{e}, \mathtt{bidfor}(\mathtt{c_1}, \mathtt{c_2}))$$

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c_1}, \mathtt{c_2} \in \mathtt{companies}\ \mathtt{act(e, rejectsBid)} \wedge \mathtt{buyer(e, c_1)} \wedge \mathtt{target(e, c_2)} \wedge \mathtt{body(e, regulator)} \rightarrow \mathsf{terminates}(\mathtt{e}, \mathtt{takeover}(\mathtt{c_1}, \mathtt{c_2}))$$

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c_1}, \mathtt{c_2} \in \mathtt{companies}\ \mathtt{act(e, approvesBid)} \wedge \mathtt{buyer(e, c_1)} \wedge \mathtt{target(e, c_2)} \wedge \mathtt{body(e, regulator)} \rightarrow \mathsf{terminates}(\mathtt{e}, \mathtt{takeover}(\mathtt{c_1}, \mathtt{c_2}))$$

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c_1}, \mathtt{c_2} \in \mathtt{companies}\ \mathtt{act(e, approvesBid)} \wedge \mathtt{buyer(e, c_1)} \wedge \mathtt{target(e, c_2)} \wedge \mathtt{body(e, regulator)} \rightarrow \mathsf{initiates}(\mathtt{e}, \mathtt{owns}(\mathtt{c_1}, \mathtt{c_2}))$$

To determine $\mathsf{follows}(\mathtt{e_3}, \mathtt{bidFor(ryanair, buzz)})$*, first determine that* e_3 *does not terminate* $\mathtt{bidFor(ryanair, buzz)}$*. There is however a state* $\mathtt{s'}$ *such that an event* $\mathtt{e}$ *could terminate:* $\mathtt{takeover(ryanair, buzz)}$*. Let* $\mathtt{e'}$ *initiate* $\mathtt{takeover(ryanair, buzz)}$*. If this were so,* $\mathtt{e'}$ *would have to be*

$$\{\mathtt{act(e', acceptsBid)}, \mathtt{buyer(e', ryanair)}, \mathtt{target(e, buzz)}, \mathtt{body(e', shareholders)}\}$$

If $\mathtt{e'} = \{\mathtt{act(e', acceptsBid)}, \mathtt{buyer(e', ryanair)}, \mathtt{target(e, buzz)}, \mathtt{body(e', board)}\}$ *, is it the case that* $\mathsf{follows}(\mathtt{e'}, \mathtt{bidFor(ryanair, buzz)})$*?*

e' *terminates* $\mathtt{bidFor(ryanair, buzz)}$*, therefore ,* $\mathsf{follows}(\mathtt{e'}, \mathtt{bidFor(ryanair, buzz)})$ *and* $\mathsf{follows}(\mathtt{e_1}, \mathtt{bidFor(ryanair, buzz)})$

Definition 7.5.3. *Let* e *be an event and* s *be a state. That the period* $\mathsf{after}(e, s)$ *is begun by event* e *is formally defined as:*

$$\forall e \in \mathit{events}\ \mathsf{start}(\mathsf{after}(e, s), e)$$

That the period $\mathsf{after}(e, s)$ *is ended by* e' *is formally defined as:*

$$\forall e \in \mathit{events}\ \mathsf{follows}(e', s) \wedge \neg\exists\ e^* \textit{such that } \mathsf{follows}(e^*, s) \wedge \mathit{time}(e^*) < \mathit{time}(e') \rightarrow \mathsf{end}(\mathsf{after}(e, s), e')$$

That the period $\mathsf{before}(e', s)$ *is begun by event* e *is formally defined as:*

$$\forall e \in \mathit{events}\ \mathsf{precedes}(e, s) \wedge \neg\exists\ e^* \textit{such that } \mathsf{precedes}(e^*, s) \wedge \mathit{time}(e) < \mathit{time}(e^*) \rightarrow \mathsf{start}(\mathsf{before}(e', s), e)$$

That the period $\mathsf{before}(e', s)$ *is ended by the event* e' *is defined as:*

$$\forall e \in \mathit{events}\ \mathsf{end}(\mathsf{before}(e', s), e')$$

These new definitions for $\mathsf{begins}(s, e)$ and $\mathsf{ends}(s, e)$ now means that the Definition 7.4.2 now returns two states for the example given in Example 7.5.1. This is an improvement in the amount of information returned. The original definition provided no information about the states that may hold at a given timepoint. The new definition increases the information we have about the state at a given timepoint by restricting that information to a subset of the possible states.

Example 7.5.4. *Let*

$$\{\mathtt{act}(\mathtt{e}_1, \mathtt{bidFor}), \mathtt{buyer}(\mathtt{e}_1, \mathtt{ryanair}), \mathtt{target}(\mathtt{e}_1, \mathtt{buzz})\}$$

and

$$\{\mathtt{act}(\mathtt{e}_2, \mathtt{approvedBid}), \mathtt{buyer}(\mathtt{e}_2, \mathtt{ryanair}), \mathtt{target}(\mathtt{e}_2, \mathtt{buzz}), \mathtt{body}(\mathtt{e}_2, \mathtt{shareholders})\}$$

be event records and the set of event records be

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c}_1, \mathtt{c}_2 \in \mathtt{companies}\ \mathtt{act}(\mathtt{e}, \mathtt{bidFor}) \wedge \mathtt{buyer}(\mathtt{e}, \mathtt{c}_1) \wedge \mathtt{target}(\mathtt{e}, \mathtt{c}_2) \rightarrow \mathsf{initiates}(\mathtt{e}, \mathtt{bidFor}(\mathtt{c}_1, \mathtt{c}_2))$$

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c}_1, \mathtt{c}_2 \in \mathtt{companies}\ \mathtt{act}(\mathtt{e}, \mathtt{approvesBid}) \wedge \mathtt{buyer}(\mathtt{e}, \mathtt{c}_1) \wedge \mathtt{target}(\mathtt{e}, \mathtt{c}_2) \wedge \mathtt{body}(\mathtt{e}, \mathtt{board}) \rightarrow \mathsf{terminates}(\mathtt{e}, \mathtt{bidFor}(\mathtt{c}_1, \mathtt{c}_2))$$

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c}_1, \mathtt{c}_2 \in \mathtt{companies}\ \mathtt{act}(\mathtt{e}, \mathtt{approvesBid}) \wedge \mathtt{buyer}(\mathtt{e}, \mathtt{c}_1) \wedge \mathtt{target}(\mathtt{e}, \mathtt{c}_2) \wedge \mathtt{body}(\mathtt{e}, \mathtt{board}) \rightarrow \mathsf{initiates}(\mathtt{e}, \mathtt{takeover}(\mathtt{c}_1, \mathtt{c}_2))$$

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c}_1, \mathtt{c}_2 \in \mathtt{companies}\ \mathtt{act}(\mathtt{e}, \mathtt{rejectsBid}) \wedge \mathtt{buyer}(\mathtt{e}, \mathtt{c}_1) \wedge \mathtt{target}(\mathtt{e}, \mathtt{c}_2) \wedge \mathtt{body}(\mathtt{e}, \mathtt{board}) \rightarrow \mathsf{terminates}(\mathtt{e}, \mathtt{bidfor}(\mathtt{c}_1, \mathtt{c}_2))$$

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c}_1, \mathtt{c}_2 \in \mathtt{companies}\ \mathtt{act}(\mathtt{e}, \mathtt{rejectsBid}) \wedge \mathtt{buyer}(\mathtt{e}, \mathtt{c}_1) \wedge \mathtt{target}(\mathtt{e}, \mathtt{c}_2) \wedge \mathtt{body}(\mathtt{e}, \mathtt{regulator}) \rightarrow \mathsf{terminates}(\mathtt{e}, \mathtt{takeover}(\mathtt{c}_1, \mathtt{c}_2))$$

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c}_1, \mathtt{c}_2 \in \mathtt{companies}\ \mathtt{act}(\mathtt{e}, \mathtt{approvesBid}) \wedge \mathtt{buyer}(\mathtt{e}, \mathtt{c}_1) \wedge \mathtt{target}(\mathtt{e}, \mathtt{c}_2) \wedge \mathtt{body}(\mathtt{e}, \mathtt{regulator}) \rightarrow \mathsf{terminates}(\mathtt{e}, \mathtt{takeover}(\mathtt{c}_1, \mathtt{c}_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{regulator}) \rightarrow \texttt{initiates}(e, \texttt{owns}(c_1, c_2))$$

According to definition 7.5.3, e_2 now ends $\texttt{bidFor}(\texttt{ryanair}, \texttt{buzz})$ *and e_1 now initiates* $\texttt{takeover}(\texttt{ryanair}, \texttt{buzz})$. *Therefore, according to Definition 7.4.1, if $time(e_1) < t < time(e_2)$ then t is in both the periods where state* $\texttt{bidFor}(\texttt{ryanair}, \texttt{buzz})$ *holds and where stat e* $\texttt{takeover}(\texttt{ryanair}, \texttt{buzz})$ *holds.*

Therefore, using the new definition, data is derivable about two possible states that may hold at time t. This is an improvement, for our purposes on the original definition, which provided no information at all about the possible states that may hold at time t as we can now draw tentative conclusions about those states which may hold, as well as being able to rule out those states that definitely do not hold. However, this definition as it stands returns only the first and the last of an unknown sequence of events that may hold at time t. We will now examine a definition that allows us to enumerate all the possible states that may have fallen between those first and last states.

Definition 7.5.4. *Let e, e^* be events, s and s' be states. The minimum set of all possible states that may hold between two events:*

$$\begin{aligned}\mathsf{interveningStates}(e, s) = \{s' | &\mathsf{precedes}(e, s)\ \textit{and} \\ &\neg\exists\ e^*\ \textit{s.t.}\ \mathsf{precedes}(e^*, s)\ \textit{and}\ time(e^*) > time(e)\ \textit{and} \\ &\mathsf{precedes}(e', s)\ \textit{and}\ \mathsf{initiates}(e, s^*)\ \textit{and}\ \mathsf{follows}(e', s^*) \textit{and} \\ &\mathsf{initiates}(e', s')\end{aligned}$$

Example 7.5.5. *Let*

$$\{\texttt{act}(e_1, \texttt{bidFor}), \texttt{buyer}(e_1, \texttt{bmw}), \texttt{target}(e_1, \texttt{rover})\}$$

$$\{\texttt{act}(e_2, \texttt{sells}, \texttt{bmw}, \texttt{rover})\}$$

be event records and the following be a set of initiates and terminates rules:

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{bidFor}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \rightarrow \texttt{initiates}(e, \texttt{bidFor}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{board}) \rightarrow \texttt{terminates}(e, \texttt{bidFor}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{board}) \rightarrow \text{initiates}(e, \texttt{takeover}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{rejectsBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{board}) \rightarrow \text{terminates}(e, \texttt{bidfor}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{rejectsBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{regulator}) \rightarrow \text{terminates}(e, \texttt{takeover}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{regulator}) \rightarrow \text{terminates}(e, \texttt{takeover}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{approvesBid}) \wedge \texttt{buyer}(e, c_1) \wedge \texttt{target}(e, c_2) \wedge \texttt{body}(e, \texttt{regulator}) \rightarrow \text{initiates}(e, \texttt{owns}(c_1, c_2))$$

$$\forall e \in \texttt{events},\ c_1, c_2 \in \texttt{companies}\ \texttt{act}(e, \texttt{sells}, c_1, c_2) \rightarrow \text{terminates}(e, \texttt{owns}(c_1, c_2))$$

It is the case that $\text{precedes}(e_1, \texttt{owns}(\texttt{bmw}, \texttt{rover}))$ *and that there is no recorded event that precedes* $\texttt{owns}(\texttt{bmw}, \texttt{rover})$ *that is later than event* e_1. *It is also the case that there are hypothetical states begun by events that both precede* $\texttt{owns}(\texttt{bmw}, \texttt{rover})$ *and follow* $bidFor(bwm, rover)$ *therefore*

$$\begin{aligned}\text{interveningStates}(e_1, \texttt{owns}(\texttt{bmw}, \texttt{rover})) = \{&\texttt{bidFor}(\texttt{bmw}, \texttt{rover}),\\ &\texttt{takeover}(\texttt{bmw}, \texttt{rover}),\\ &\texttt{owns}(\texttt{bmw}, \texttt{rover})\}\end{aligned}$$

The new definitions go some way to increasing the information we can derive about states from the event records. There remain two shortcomings however: deriving that a timepoint falls in a state with no recorded preceding events and deriving that a timepoint falls in a state with no recorded following events.

Definition 7.5.5. *An event* e *is an initial event, that is,* $\text{initial}(e)$ *iff:*

$$\begin{aligned}&\exists\ s \in \textit{states such that}\ \text{terminates}(e, s)\ \textit{and}\\ &\neg\exists e' \in \textit{events s.t.}\ \text{precedes}(e', s)\end{aligned}$$

Example 7.5.6. *Let*

$$\forall \mathtt{c} \in \mathtt{companies},\ \mathtt{e} \in \mathtt{events}\ \mathtt{act(e, privatised)} \wedge \mathtt{object(e, c)} \rightarrow \mathsf{terminates}(\mathtt{e}, \mathtt{stateowned(c)})$$

be a terminates rule and

$$\{\mathtt{act}(\mathtt{e}_1, \mathtt{privatised}) \wedge \mathtt{object}(\mathtt{e}, \mathtt{britishGas})\}$$

be an event record. Let there be no e_n *such that* $\mathsf{initiates}(\mathtt{e_n}, \mathtt{stateowned}(\mathtt{britishGas}))$.

By Definition 7.5.5, $\mathtt{e}_1$ *is an initial event.*

Definition 7.5.6. *An event* e *is an terminal event, that is* $\mathsf{terminal}(e)$ *iff:*

$$\exists\ s \in \textit{states such that}\ \mathsf{initiates}(e, s)\ \textit{and}$$
$$\neg\exists e' \in \textit{events s.t.}\ \mathsf{follows}(e', s)$$

Example 7.5.7. *Let*

$$\forall \mathtt{c}_1, \mathtt{c}_2 \in \mathtt{companies},\ \mathtt{e} \in \mathtt{events}\ \mathtt{act(e, bidAccepted)} \wedge \mathtt{buyer(e, c_1)} \wedge \mathtt{target(e_c 2)} \wedge \mathtt{body(e, regulator)} \rightarrow \mathsf{initiates}(\mathtt{e}, \mathtt{owns}(\mathtt{c}_1, \mathtt{c}_2))$$

be an initiates rule and

$$\{\mathtt{act}(\mathtt{e}_1, \mathtt{bidAccepted}) \wedge \mathtt{buyer}(\mathtt{e}, \mathtt{morrisons}), \mathtt{target}(\mathtt{e}_1, \mathtt{safeway}), \mathtt{body}(\mathtt{e}_1, \mathtt{regulator})\}$$

be an event record. Let there be no e_n *such that* $\mathsf{terminates}(\mathtt{e_n}, \mathtt{owns}(\mathtt{morrisons}, \mathtt{safeway}))$.

By Definition 7.5.6, $\mathtt{e}_1$ *is an terminal event.*

If an event is an initial event, then the state which that event terminates is assumed to hold at all points up to that event. If an event is a terminal event, then the state which that event initiates is assumed to hold at all points since that event.

Definition 7.5.7. *Let* s *be a state,* e *be an event and* t *be a timepoint. The rules determining whether* $\mathsf{holdsAt}(s, t)$ *must be changed to incorporate knowledge of terminal and initial events.*

$$(\mathsf{holds}(\mathsf{after}(e, s)) \wedge t\ \textit{in}\ \mathsf{after}(e, s)) \vee (\mathsf{holds}(\mathsf{after}(e, s)) \wedge t > \mathit{time}(e) \wedge \mathsf{terminal}(e)) \rightarrow \mathsf{holdsAt}(s, t)$$

and

$$(\mathsf{holds}(\mathsf{before}(e, s)) \wedge t\ \textit{in}\ \mathsf{before}(e, s)) \vee (\mathsf{holds}(\mathsf{before}(e, s)) \wedge t < \mathit{time}(e) \wedge \mathsf{initial}(e)) \rightarrow \mathsf{holdsAt}(s, t)$$

Example 7.5.8. *Let*

$$\{\mathtt{act}(\mathtt{e}_1, \mathtt{approvesBid}), \mathtt{buyer}(\mathtt{e}_1, \mathtt{ryanair}), \mathtt{target}(\mathtt{e}_1, \mathtt{buzz}), \mathtt{body}(\mathtt{e}_1, \mathtt{regulator})\}$$

be an event record and the following be an initiates rule:

$$\forall \mathtt{e} \in \mathtt{events},\ \mathtt{c}_1, \mathtt{c}_2 \in \mathtt{companies}\ \mathtt{act}(\mathtt{e}, \mathtt{approvesBid}) \wedge \mathtt{buyer}(\mathtt{e}, \mathtt{c}_1) \wedge \mathtt{target}(\mathtt{e}, \mathtt{c}_2) \wedge \mathtt{body}(\mathtt{e}, \mathtt{regulator}) \rightarrow \mathsf{initiates}(\mathtt{e}, \mathtt{owns}(\mathtt{c}_1, \mathtt{c}_2))$$

There is no event $\mathtt{e}_2$ *such that* $\mathtt{time}(\mathtt{e}_2) > \mathtt{time}(\mathtt{e}_1)$*, therefore* $\mathsf{terminal}(\mathtt{e}_1)$*. Let the time of* $\mathtt{e}_1$ *be September 2003 and* $\mathtt{t}$ *be the 1st January 2005. By Definition 7.5.7,* $\mathsf{holdsAt}(\mathtt{owns}(\mathtt{ryanair}, \mathtt{buzz}), \mathtt{t})$*, as the state that Ryanair owns Buzz was initiated in September 2003 and has not been terminated since.*

7.6 An example event model for the business domain

This section is an extended example of a state model and an event model for the business domain.

Firstly let the following be the initiates and terminates rules for the domain:

$$\forall e_1 \in events,\ c_1, c_2 \in companies\ act(e_1, bidFor) \wedge buyer(e_1, c_1) \wedge target(e_1, c_2) \rightarrow \mathsf{initiates}(e_1, bidFor(c_1, c_2))$$

$$\forall e_2 \in events,\ c_1, c_2 \in companies\ act(e_2, approvesBid) \wedge buyer(e_2, c_1) \wedge target(e_2, c_2) \wedge body(e_2, board) \rightarrow \mathsf{terminates}(e_2, bidFor(c_1, c_2))$$

$$\forall e_2 \in events,\ c_1, c_2 \in companies\ act(e_2, approvesBid) \wedge buyer(e_2, c_1) \wedge target(e_2, c_2) \wedge body(e_2, board) \rightarrow \mathsf{initiates}(e_2, takeover(c_1, c_2))$$

$$\forall e_3 \in events,\ c_1, c_2 \in companies\ act(e_3, rejectsBid) \wedge buyer(e_3, c_1) \wedge target(e, c_2) \wedge body(e_3, board) \rightarrow \mathsf{terminates}(e_3, bidfor(c_1, c_2))$$

$$\forall e_4 \in events,\ c_1, c_2 \in companies\ act(e, rejectsBid) \wedge buyer(e, c_1) \wedge target(e, c_2) \wedge body(e, regulator) \rightarrow \mathsf{terminates}(e, takeover(c_1, c_2))$$

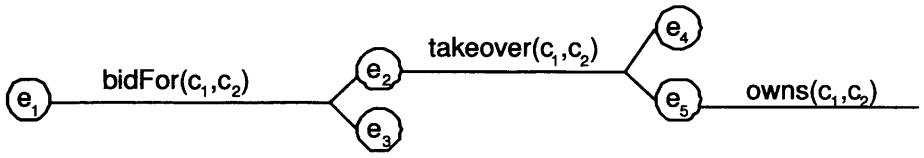


Figure 7.6: A simplified example of a state model for the mergers and acquisitions domain based on the rules in Example 7.5.7

$$\forall e_5 \in events,\ c_1, c_2 \in companies\ act(e, approvesBid) \wedge buyer(e, c_1) \wedge target(e, c_2) \wedge body(e, regulator) \rightarrow \mathsf{terminates}(e, takeover(c_1, c_2))$$

$$\forall e_5 \in events,\ c_1, c_2 \in companies\ act(e, approvesBid) \wedge buyer(e, c_1) \wedge target(e, c_2) \wedge body(e, regulator) \rightarrow \mathsf{initiates}(e, owns(c_1, c_2))$$

Figure 7.6 represents the state model that can be derived from these rules.

Recall from 7.2.2 that the general rule "State s holds after event e if e initiates s" is:

$$\forall e \in events,\ s \in states,\ \mathsf{initiates}(e, s) \rightarrow \mathsf{holds}(\mathsf{after}(e, s))$$

and the general rule "State s holds before event e if e terminates s" is:

$$\forall e \in events,\ s \in states,\ \mathsf{terminates}(e, s) \rightarrow \mathsf{holds}(\mathsf{before}(e, s))$$

Let r_1, r_2, r_3 and r_4 be representative sets:

r_1: {act(bidFor), buyer(ryanair), target(buzz)}
r_2: {act(approvesBid), buyer(ryanair), target(buzz), body(regulator)}
r_3: {act(approvesBid), buyer(morrisons), target(safeway), body(shareholders)}
r_4: {act(approvesBid), buyer(morrisons), target(safeway), body(regulator)}

Let Σ, the set of event rules, be

$$\{\forall \mathtt{b}, \mathtt{t}\ \exists \mathtt{e}\ \mathtt{buyer(b)} \wedge \mathtt{target(t)} \wedge \mathtt{act(bidFor)} \rightarrow \mathtt{buyer(e, b)} \wedge \mathtt{target(e, t)} \wedge \mathtt{act(e, bidFor)},$$
$$\{\forall \mathtt{b}, \mathtt{t}, \mathtt{d}\ \exists \mathtt{e}\ \mathtt{buyer(b)} \wedge \mathtt{target(t)} \wedge \mathtt{act(bidFor)} \wedge \mathtt{body(d)} \rightarrow \mathtt{buyer(e, b)} \wedge \mathtt{target(e, t)} \wedge \mathtt{act(e, bidFor)}, \wedge \mathtt{body(e, d)}$$

Therefore

$$\begin{aligned}
&\mathsf{Events}(r_1, \Sigma) = \{\mathtt{act}(e_1, \mathtt{bidFor}), \mathtt{buyer}(e_1, \mathtt{ryanair}), \mathtt{target}(e_1, \mathtt{buzz})\} \\
&\mathsf{Events}(r_2, \Sigma) = \{\mathtt{act}(e_2, \mathtt{approvesBid}), \mathtt{buyer}(e_2, \mathtt{ryanair}), \mathtt{target}(e_2, \mathtt{buzz}), \\
&\qquad \mathtt{body}(e_2, \mathtt{regulator})\} \\
&\mathsf{Events}(r_3, \Sigma) = \{\mathtt{act}(e_3, \mathtt{approvesBid}), \mathtt{buyer}(e_3, \mathtt{morrisons}), \mathtt{target}(e_3, \mathtt{safeway}), \\
&\qquad \mathtt{body}(e_3, \mathtt{shareholders})\} \\
&\mathsf{Events}(r_4, \Sigma) = \{\mathtt{act}(e_4, \mathtt{approvesBid}), \mathtt{buyer}(e_4, \mathtt{morrisons}), \mathtt{target}(e_4, \mathtt{safeway}), \\
&\qquad \mathtt{body}(e_4, \mathtt{regulator})\}
\end{aligned}$$

The above event records lead us to the conclusions that:

$$\mathtt{act}(e_1, \mathtt{bidFor}) \wedge \mathtt{buyer}(e_1, \mathtt{ryanair}) \wedge \mathtt{target}(e_1, \mathtt{buzz}) \rightarrow \mathsf{initiates}(e_1, \mathtt{bidFor}(\mathtt{ryanair}, \mathtt{buzz}))$$

$$\mathtt{act}(e_2, \mathtt{approvesBid}) \wedge \mathtt{buyer}(e_2, \mathtt{ryanair}) \wedge \mathtt{target}(e_2, \mathtt{buzz}) \wedge \mathtt{body}(e_2, \mathtt{shareholders}) \rightarrow \mathsf{terminates}(e_2, \mathtt{takeover}(\mathtt{ryanair}, \mathtt{buzz}))$$

$$\mathtt{act}(e_2, \mathtt{approvesBid}) \wedge \mathtt{buyer}(e_2, \mathtt{ryanair}) \wedge \mathtt{target}(e_2, \mathtt{buzz}) \wedge \mathtt{body}(e_2, \mathtt{regulators}) \rightarrow \mathsf{initiates}(e_2, \mathtt{owns}(\mathtt{ryanair}, \mathtt{buzz}))$$

$$\mathtt{act}(e_3, \mathtt{approvesBid}) \wedge \mathtt{buyer}(e_3, \mathtt{morrisons}) \wedge \mathtt{target}(e_3, \mathtt{safeway}) \wedge \mathtt{body}(e_3, \mathtt{shareholders}) \rightarrow \mathsf{initiates}(e_3, \mathtt{takeover}(\mathtt{ryanair}, \mathtt{buzz}))$$

$$\mathtt{act}(e_3, \mathtt{approvesBid}) \wedge \mathtt{buyer}(e_3, \mathtt{morrisons}) \wedge \mathtt{target}(e_3, \mathtt{safeway}) \wedge \mathtt{body}(e_3, \mathtt{shareholders}) \rightarrow \mathsf{terminates}(e_3, \mathtt{bidFor}(\mathtt{ryanair}, \mathtt{buzz}))$$

$$\mathtt{act}(e_4, \mathtt{approvesBid}) \wedge \mathtt{buyer}(e_4, \mathtt{morrisons}) \wedge \mathtt{target}(e_4, \mathtt{safeway}) \wedge \mathtt{body}(e_4, \mathtt{regulator}) \rightarrow \mathsf{terminates}(e_3, \mathtt{takeover}(\mathtt{ryanair}, \mathtt{buzz}))$$

$$\mathtt{act}(e_4, \mathtt{approvesBid}) \wedge \mathtt{buyer}(e_4, \mathtt{morrisons}) \wedge \mathtt{target}(e_4, \mathtt{safeway}) \wedge \mathtt{body}(e_4, \mathtt{regulator}) \rightarrow \mathsf{initiates}(e_4, \mathtt{owns}(\mathtt{ryanair}, \mathtt{buzz}))$$

This event model can then be used to determine the states that hold for entities at given times, using Definition 7.5.7: Let

$$\epsilon_1 = \forall \mathtt{x}, \mathtt{y}\ \mathtt{bidFor}(\mathtt{x}, \mathtt{y}) \rightarrow \mathtt{profitable}(\mathtt{x})$$

and

$$\epsilon_2 = \forall \mathtt{x}, \mathtt{y}\ \mathtt{takeover}(\mathtt{x}, \mathtt{y}) \rightarrow \mathtt{profitable}(\mathtt{x})$$

be expectations and let

$$\{\mathtt{timePoint}(\mathtt{t}_1), \neg\mathtt{profitable}(\mathtt{ryanair})\}$$

be a set of facts extracted from a news report, ρ_1

Let $time(e_1) < t_1 < time(e_2)$. From the above event records, state model and the definition of holdsAt(s, t) in Definition 7.5.7, it is possible to derive holdsAt$(\mathtt{bidFor}(\mathtt{ryanair}, \mathtt{buzz}), \mathtt{t}_1)$ and
holdsAt$(\mathtt{takeover}(\mathtt{ryanair}, \mathtt{buzz}), \mathtt{t}_1)$. The facts $\mathtt{bidFor}(\mathtt{ryanair}, \mathtt{buzz})$ and $\mathtt{takeover}(\mathtt{ryanair}, \mathtt{buzz})$ can then be added to representatives(ρ_1, Γ, Δ). Therefore representatives(ρ_1, Γ, Δ) violates ϵ_1 and ϵ_2.

Likewise, let

$$\epsilon_2 = \forall \mathtt{x}, \mathtt{y}, \mathtt{z}\ \mathtt{owns}(\mathtt{x}, \mathtt{y}) \rightarrow \neg\mathtt{bidFor}(\mathtt{y}, \mathtt{z})$$

be an expectation and let

$$\{\mathtt{timepoint}(\mathtt{t}_2), \mathtt{profitable}(\mathtt{morrisons})\}$$

be facts extracted from report ρ_2.

Let $e_4 < t_2$. Therefore, by Definition 7.5.7, holdsAt$(\mathtt{owns}(\mathtt{morrisons}, \mathtt{safeway}), \mathtt{t}_2)$. The fact $\mathtt{owns}(\mathtt{morrisons}, \mathtt{safeway})$ is in states(ρ_2, Γ, Σ). Therefore representatives(ρ_2, Γ, Δ) fires, but does not violate, ϵ_2.

7.7 Discussion of Chapter 7

In order to reason about expectations that hold only for given states it is necessary to be able to determine which states hold at which timepoints. One way of achieving this is with a logical calculus of states and events. This chapter has demonstrated that it is possible to reason about the narratives in news reports by using the event calculus as defined in [KS85] with some novel

extensions that permit the interpolation of missing state information. The event calculus was chosen as the basis for the event models illustrated in this chapter because it has clear semantics, can cope with simultaneous sequences of events and is easily implemented in Prolog, thus making it ideal for an implemented EVA system. The extensions suggested in this chapter preserve these advantages.

In this chapter, the event calculus is used to generate event models. However, there is no reason why another formalism may not be used, as long as it is able to produce a set of facts analogous to $\mathsf{states}(\rho, \Gamma, \Sigma)$ for any given report ρ, set of access rules, Γ and event model Σ. in order to use the event model in an EVA system, it is necessary only to know $\mathsf{states}(\rho, \Gamma, \Sigma)$, the method by which the set is produced is not relevant to the evaluation of reports with respect to expectations. As such, event models can be viewed as being modular.

An EVA system may in practice have more than one event model in order to deal with different types of narratives. In the mergers and acquisitions domain for example, it is necessary to keep track of the state of a company's key performance indicators such as profit, earnings per share, and so on. An event model may exist to record and reason with events pertaining to company figures that is totally separate to the event model for merger activities.

The modifications to the event calculus presented in Definitions 7.5.3 and 7.5.7 result in more information than the original definitions in [KS85]. In the original formulation, the function $\mathsf{holdsAt}(t, s)$ only holds for states with explicit start and end points. The result of this is that, in the case of missing events in the narrative, no information is provided about the states between two timepoints. In Definition 7.5.7, the function $\mathsf{holdsAt}(t, s)$ is modified so that it holds for all possible states on the path between two events, assuming that there are no repetitions of states. As a result it is possible to tentatively apply state dependent expectations that are relevant to those hypothetical states. This may result in false positive firings, confirmations and violations of expectations, but at least allows some conclusions to be drawn when there are missing states.

The original formulation of the event calculus did not allow us to derive information about unterminated states. Due to the up-to-date nature of the reports that are assessed by an EVA system, it is likely that many of the states relevant to events in the reports are, as yet, unterminated. The second advantage of the modifications to the event calculus presented here is that terminal states (those that are not yet terminated, either directly or indirectly) can be derived by using the extended formulation in Definition 7.5.7.

The event models for an EVA system are, then, specialised sets of background knowledge, with their own axioms and functions. There may be other types of knowledge for certain domains that also benefit from specialised definitions. It is envisaged that these too could be defined in a similar way to event models, and queried by a function analogous to $\mathsf{states}(\rho, \Gamma, \Sigma)$ that will return a set of facts relevant to a given report.

Chapter 8

Existing Research Findings and their Relevance to EVA

This chapter will discuss the Expectation Violation Analysis (EVA) framework with respect to the existing state of the art. A summary of these conclusions is presented below whilst detailed information will be given in sections 8.1 and 8.2.

The EVA framework is not the first attempt to solve the problem of information overload. It is however the first approach that addresses the problem of identifying news that is interesting because of its unexpectedness. There is a wealth of literature that covers other approaches to identifying information and there is a danger that the EVA framework, due to its applicability to filtering large volumes of information may be seen as "just another information retrieval tool".

I would argue this is not the case, as the current focus of the development of Information Retrieval and Filtering (IR/IF) tools is almost entirely relevance-oriented and does not address unexpectedness. The TREC (Text Retrieval Conferences) provide a standard corpus of texts and developers of IR/IF systems demonstrate the effectiveness of their systems with respect to this corpus, by measuring recall and precision rates for queries addressed to it.

In contrast to IR/IF, the EVA framework has no measure of relevance and so would not be suitable for searching for documents that are relevant to queries. An EVA system *would* identify reports containing *unexpected* information. Current IR/IF systems do not aim to identify unexpected information. The EVA framework, whilst it is an information management approach, is a distinct but complementary approach to those of IR and IF.

Another way of managing information is collaborative filtering. In collaborative filtering it is users rather than documents that are classified, specifically into groups with similar tastes. Users who have expressed similar tastes are grouped by some method. The recommendation of an item

by one group member is considered to be a reliable recommendation for other members of that group. The advantage of such an approach is that it exploits the user's ability to make qualitative judgments. The drawback is that the first person to view an item must do so without a recommendation, thus punishing early or frequent searchers. This is known as the *cold start problem.* One of the qualitative judgments that can be made by a user is whether a particular item (for example, a news report) is unexpected. However, the collaborative filtering method is particularly inappropriate for the purposes of news filtering due to the cold start problem.

The EVA framework is also not the first attempt to identify unexpected or novel information. Approaches arising from the data mining, machine learning and machine creativity communities also seek to identify unexpected information. Topic Detection and Tracking (TDT) is a method that arose from the field of information filtering and is a way of identifying the occurrence of topic threads in streams of documents or of topic clusters in static sets of documents. However, TDT is only concerned with the *novelty* of the topic and not the *unexpectedness* of the information concerning that topic.

Belief driven approaches do attempt to identify unexpectedness. These approaches attempt to capture the user's *a priori* expectations and use these expectations to identify rules identified by data mining that are extrinsically unexpected (that is, unexpected with respect to that particular user's reported worldview). Unlike the EVA framework, belief driven approaches rely on the presence of an expert user in order to generate a set of beliefs. An EVA system has no need of expert input as it has the ability to learn a set of expectations from news reports and background knowledge.

The identification of deviations in numerical values is one way of identifying intrinsic unexpectedness (that is, unexpectedness with respect to the usual run of events). Numerical deviations are only applicable to quantitative information whereas the EVA framework is concerned with qualitative information.

The comparisons between the EVA framework and other approaches to information management and to the identification of unexpected information will now be considered in detail.

8.1 The EVA Framework contrasted with information management

Clearly, the EVA framework is not the first attempt to make large volumes of information more manageable to their users. Information retrieval, information filtering and collaborative filtering are three popular and well researched methods of information management. However, I will argue that the EVA framework is more than 'just another information management method': there are clear and significant differences between all of these methods and the EVA framework that

represent a step change away from traditional relevance or cluster based information management systems.

8.1.1 Information retrieval

Information retrieval (IR) has flourished as a means of dealing with large numbers of documents. Whilst IR predates the world wide web and began as a discipline of library science, the aim remains the same: to present a user with the information that matches their needs. Over recent years this approach has been adapted to the specific needs of retrieving electronically-stored information. A user presents the IR system with a query, which is then matched against a relatively static set of documents, from which the user is presented with those documents which exactly or closely match their query.

The Boolean retrieval model is an exact match approach. Queries consist of keywords joined by the Boolean connectives representing conjunction, disjunction and negation. Documents are treated as sets of words. The Boolean retrieval model is an "exact match" model, which partitions the document space into two sets: documents which match the query and documents which do not [War92]. Documents are not ranked further beyond identifying whether they are members of the set of relevant documents. Boolean retrieval is one of the oldest methods of information retrieval, but its reliance on an exact match between the query term and the document makes it limited in its effectiveness [Fuh01].

Term Frequency/Inverse Document Frequency (TF/IDF) is a ranking technique that has been employed in both information retrieval and information extraction [Sek03]. Such weightings can help to improve the usefulness of the results from a query matching system by adding ranking information. Inverse Document Frequency (IDF) is $-log\ n/N$ where N is the number of available documents and n is the number of documents in this set which contain the term. IDF identifies terms which are good differentiators of documents (those with a low n/N ratio) from those which are poor differentiators. Term Frequency (TF) is a count of the number of occurrences of a term in a document and documents are weighted by the frequency of highly discriminating terms from the query [PC98]. For example, in the query "big AND cats AND leopards" the term "leopard" is likely to be a better discriminator (4,160,000 Google hits as at 12/01/05) than "big" (241,000,000 Google hits) and "cat" (127,000,000 Google hits). Therefore occurrences of the term leopard will be more heavily weighted than occurrences of the terms "big" or "cat".

Exact match methods rely on matching words in documents to those in queries. Stemming increases the number of hits by looking for terms which are related but are different parts of speech by truncating the term. For example the query '`stemming`' would also return documents containing the terms '`stem`', '`stemmed`', '`stemmer`' and so on. Indexing on stems, rather than full terms, also reduces the size of the index file. Compression factors of over 50% can be achieved by indexing on stems [Fra92]. However, stemming can be misleading. For example the query '`mute`'

may find also documents which match 'mutable', 'mutate', 'mutation' because of inappropriate stemming, when we really may prefer to match 'silent' or 'speechless'. In short, lexigraphic similarity is insufficient to capture (and often unrelated to) semantic similarity.

The vector space model (VSM) [Voo99] is a best match approach to IR that treats each word in queries and in documents as a dimension in a high-dimensional space. The vectors of queries and documents are compared by using a measure of spatial similarity: the cosine of the angle between the vectors. The closer the angle between the document and the query the greater the likelihood that the document is relevant to the user. This approach is more flexible than Boolean matching as it allows for similarities between documents and query terms to be identified and does not demand an exact match. The VSM approach is more likely to identify words with similar meanings than the stemming approach as, as mentioned above, lexigraphic similarity is often a poor indicator of semantic similarity. However, because of the number of terms required in order to generate a useful lexicon, and the relatively low number of documents each term will appear in, the vector space model tends to result in a sparsely populated space of very high dimension. Such a space leads to difficulties in making useful generalisations about term associations.

Latent Semantic Indexing (LSI) addresses the problems that arise from the use of natural language documents by organising textual information into a *semantic* structure. The method compares co-occurrence frequencies of terms and derives associations between them. These associations usually denote related concepts and a query is automatically augmented with terms which occur in the associations of the terms in the query. For more details see [RLDL97]. Different users may use different words to mean the same concept, such as "car" and "automobile", or they may use the same word to mean different concepts, for example "train" (noun: mode of transport) and "train" (verb: to teach or learn a skill). Latent semantic analysis reduces the number of dimensions in the original vector space by identifying concepts: groups of co-occurring terms which describe the same idea. Using LSI rather than raw textual data both reduces the size of the space to be searched and also allows for more sophisticated addressing of users' queries; a user may submit the query "car" and have returned documents which do not contain the word car but do contain the word automobile for example. For a review of approaches to LSI see [DDL+90]. Papadimitriou *et al* demonstrate in [PTRV98] that LSI can be used to improve the retrieval performance of information retrieval systems by capturing the underlying meaning of terms in the document.

Probabilistic retrieval (See [Fuh92, CCH92] for reviews) is based on the principle that documents should be ranked based on the probability that they are relevant to the query. A document will be retrieved if the cost of retrieving it given the probability it is relevant is lower than the cost of not retrieving it [Fuh01]. The probability of a document d being relevant to a query is given as:

$$Pr(rel|d)$$

and the probability of it not being relevant is given as

$$Pr(\overline{rel}|d)$$

The cost of retrieving a document may include factors such as the business of the server and the network and the complexity of the query [LLH05], as well as the monetary cost (subscription price, bandwidth charges) of obtaining the information. This must then be offset against the benefit of obtaining that information if it is relevant, or added to the cost of examining that information if it is not relevant. Therefore the overall cost, the total costs minus the total benefits, of retrieving a document are:

$C_{ret,rel}$ when the document is relevant and $C_{ret,\overline{rel}}$ when not relevant

and the cost of not receiving a document is

$C_{\overline{ret},rel}$ when the document is relevant and $C_{\overline{ret},\overline{rel}}$ when not relevant

The economic cost of retrieving a document d is:

$$EC_{ret} = Pr(rel|d)C_{ret,rel} + Pr(\overline{rel}|d)C_{ret,\overline{rel}}$$

and the economic cost of not retrieving d is:

$$EC_{\overline{ret}} = Pr(rel|d)C_{\overline{ret},rel} + Pr(\overline{rel}|d)C_{\overline{ret},\overline{rel}}$$

If $EC_{ret} < EC_{\overline{ret}}$ then the document should be retrieved. Documents can be ranked by the probability of their relevance to the user. These probabilities are usually derived from a Bayesian model of the probability of the document being relevant. However, alterative models of uncertainty, such as Dempster-Shafer theory of evidence, have been used to handle uncertainty in structured documents for example in [Lal97, LV04, FO04]. The Dempster-Shafer method is particularly suited to the retrieval of documents in structured format as the combination rule provides a way of accumulating evidence for the document's relevance from different sections of that document.

Further advances in the field of information retrieval include the provision of cross-language information retrieval (see [McC99, BHH+00] for background material) and the use of machine learning to improve the performance of information retrieval systems (see [Bal97, Che95, BVSL02, OY01, SPK00] for highlights of these advances). Logic has been applied to the task of information retrieval in several ways. A number of inference mechanisms that determine the 'aboutness' of a document with respect to a query have been proposed (for a review see [LB98]).

However, these advances and refinements in information retrieval have not addressed the question

of how to identify unexpected information. The EVA framework is a distinct but complementary approach to information retrieval. IR systems, concentrating as they do on relevance, are very good at responding to information requirements that can be expressed as queries such as "Find all terrorist activities that have taken place in South America", for example. The EVA framework has no concept of a query and no measure of relevance and so would not be suitable for addressing such an information need. However, no IR system can respond to the information need "Find all unexpected information about terrorists" as none of these techniques has any measure of unexpectedness.

Information Retrieval	The EVA Framework
Identifies documents that contain information that is *relevant* to a query.	Identifies news reports that are *unexpected* with respect to background knowledge and some expectation.

Information retrieval may be brought together with the EVA approach in order to develop a more fully rounded approach to recommending news. The two approaches could be used in tandem, with information retrieval being used to pre-filter news that is relevant to a given keyword, and the EVA system being used to identify news items that are unexpected.

8.1.2 Information content filtering

Information filtering (IF) is another method of dealing with large numbers of documents. IF filters a dynamic stream of documents according to static user profiles in contrast to IR, where the set of documents is largely static but queries are dynamic. Belkin and Croft present an overview of the similarities and differences in [BC92]. For the purposes of this section, discussion will be restricted to information *content* filtering[1]. Once again, it is necessary to understand how the information filtering approach to information management differs from the EVA approach.

The same methods used to select and rank documents in information retrieval can be used to select and rank documents in information filtering. Because of the relatively static nature of user profiles, compared to the dynamic nature of the queries used in information retrieval, these profiles can be refined over time using measures such as relevance feedback, learning and belief revision [BCCN94, CCH92, Cal98, Cha99, PB97, ZS01] as well as identifying new topics in the flow of filtered documents [ZCM02]. The developments in Topic detection and Tracking will be addressed fully in Section 8.2.1.

Given the similarities of information filtering to information retrieval, it is unsurprising to realise that the contrast between IF and the EVA framework is similar to the contrast between IR and the EVA framework: IF is concerned mainly with matters of *relevance* rather than unexpectedness.

[1]This is in contrast to collaborative filtering, which will be addressed in Section 8.1.3

Information Filtering	The EVA Framework
Identifies documents that contain information that is *relevant* to a profile.	Identifies news reports that are *unexpected* with respect to background knowledge and some expectation.

As with information retrieval, information filtering could be used alongside the EVA approach in order to identify news that is both unexpected and relevant to a user's profile. Pre-filtering, by an information filtering system, could be applied before news is presented to an EVA system.

8.1.3 Collaborative filtering

Collaborative filtering addresses the lack of qualitative information from content based filtering in that it uses ratings given by human users to recommend items. The advantage of using other humans as filters is that they find it easy to deal with natural language and can also factor in elements such as the credibility of the source of the document, the style of the document and the "quality" of the document, based on the timeliness or accuracy of the information for example [Rie94].

Because of the static nature of user profiles, it is possible to identify clusters of users with similar long-term information needs. A variety of techniques have been developed to identify clusters, including probabilistic approaches, statistical measures and machine learning techniques. Terveen and Hill provide an overview of collaborative filtering systems in [TH01]. All collaborative filtering approaches use some method to group users into clusters with shared tastes, identified by similarities in their ratings of items. Once such clusters have been identified, user relevance feedback from some cluster members can be used as recommendation measures for other cluster members.

There is a drawback to collaborative filtering, known as the "cold start problem". Where there are items in the data set which no-one has rated it is impossible to recommend those items based on collaborative means. To circumvent this difficulty, Schein et al. [SPU02] propose a hybrid method, which takes content information for the unranked item and compares it with content information for ranked items. Similarities between items are identified and recommendations are made based on these similarities.

Because of the difficulties of the cold start problem, collaborative filtering is most applicable in domains where user tastes and available items are both relatively static. Good examples of this are e-commerce applications where items are recommended to users of e-commerce sites based on other shoppers purchases [SKR01], and in systems which recommend films such as Movielens [RIS+94]. In [UF98], Ungar and Foster present and evaluate a number of possible clustering methods and in [SKKR00], Sarwar et al analyse the effectiveness of a variety of collaborative filtering based recommender systems.

The overriding strength of the collaborative filtering approach is that it exploits the user's ability to make qualitative judgments. These judgments then form the basis of recommendations to other users and may capture qualities such as the unexpectedness of certain information. However, as already stated, the cold start problem makes the collaborative filtering approach inappropriate for filtering news where the timeliness of the recommendation of a news report is essential. Nevertheless, collaborative filtering does have the great strength of being able to harness the judgments of users. Therefore it may be possible for the EVA framework to form the basis of a hybrid collaborative filtering system where the judgments of users could be used to improve the recommendations of an EVA system, similar to the approach presented in [SPU02].

Collaborative Filtering	The EVA Framework
Identifies *groups of users* with similar tastes and uses these to propagate *users' qualitative judgments.*	Identifies *unexpected* news reports that should be unexpected for *all users.*

It is possible to solicit feedback from users of an EVA system to determine which unexpected news items they find to be most interesting. This data could then be used to filter the news presented to the user, both on an individual basis and also by clusters. Current clustering techniques could be applied to the space of expectations in order to determine what is most surprising to certain clusters of users. Alternatively, clusters could be created of users that rate news as interesting depending on certain features of the representative set, such as the presence of certain predicate or constant symbols.

8.2 The EVA framework versus methods of identifying novelty and interestingness

The EVA framework is not the first approach to identifying interesting items. Topic detection and tracking (Section 8.2.1) grew out of the information filtering community and enables the identification of reports of events that were previously unreported. The general impressions (Section 8.2.2) approach is a method of identifying interesting rules discovered in data mining based on the prior beliefs of domain experts. The deviations approach (Section 8.2.3) uses statistical methods to identify unexpected numerical values. A review of several methods of measuring interestingness can be found in [TKS02]. These methods have been applied to *inter alia*, bank loan data [IG01], web navigation logs [Coo00] and the rules of cricket [RR01]

8.2.1 Topic Detection and Tracking

Topic detection and tracking (TDT) is a method of identifying the first and subsequent appearances of a topic in a stream of documents. The approach began with a pilot study, supported by the US Defense Advanced Research Projects Agency (DARPA), the findings of which were reported in [ACD+98]. The findings of this study were that TDT could be regarded as a special case of information retrieval, but within a more restricted domain. As a result, a number of domain specific strategies could be brought to bear on the problem. A number of tracking methods such as K-nearest neighbour, decision trees and probabilistic queries were applied with similar success to the problem of retrospective TDT, that is, identifying topic threads in sets of documents that have already been received. Such retrospective TDT has been applied to problems of clustering web pages by topic [HDZS01]. Yang *et al* found that online clustering, clustering documents as they emerge, is more difficult than retrospective clustering but is possible [YCB+99]. A review can be found in [PA00].

While TDT is able to detect novel *topics*, that is, thematic areas that have been recently introduced to a stream, they do not identify whether the information concerning that topic is in any way *unexpected*. However, the approach could complement the EVA framework by providing a way in which news articles could be pre- or post-filtered by topic.

Topic Detection and Tracking	**The EVA Framework**
Identifies *topic threads* in a news feed or *clusters of topics* in static collections of documents..	Identifies *unexpected* news reports without regard for topic.

TDT could be used to determine whether there are multiple stories in a news report, and to identify individual stories within that report. These could then be sent to an EVA system as separate news stories, thus improving the performance of the system. Techniques used to identify topic threads could also be used to draw together related news stories, perhaps leading to the identification of cohort violations. This is an area that merits further study.

8.2.2 General Impressions

The general impressions approach to identifying interesting rules obtained by data mining was introduced by Liu *et al* [LHC97]. In this approach, domain experts were asked to enumerate their impressions for a given domain. Rules discovered by data mining were then compared with the beliefs that the general impressions of the experts revealed. The *syntactic distance* between a rule and a belief is based on the structure of the rules: if the antecedent or consequent of a rule is in accordance with a belief then that antecedent or consequent is "similar" to the belief. Conversely, if the antecedent or consequent of a rule is *not* in accordance with a belief then that antecedent

or consequent is "far from" to the belief. If the antecedent of rule is similar to a belief whilst the consequent is far from that same belief (or *vice versa*) then that rule is considered unexpected and therefore of interest. Padmanabhan and Tuzhilin extend the idea of a belief driven means of identifying interesting rules in [PT98] by restricting the definition of interestingness to that of *logical contradiction* with an impression. Sahar [Sah99] improves the efficiency of the approach by asking the user to rank a few rules, after they have been discovered. The user gives their impression of a few generated rules that, if deemed uninteresting, would automatically imply that many other discovered rules are uninteresting. This approach is similar to the EVA framework in that inconsistencies between expectations and findings are the hallmarks of interesting information.

Whilst these belief-driven approaches attempt to identifying information that is unexpected from the point of view of a user they have two drawbacks when considered as methods for identifying interesting news. Firstly, the approach relies on the specification of a limited domain. For anything other than a limited domain, the number of beliefs it would be necessary to elicit becomes unmanageably large. Secondly, the beliefs must come from an expert user. For a general news filtering system of unrestricted domain these restrictions are not practical. As a general point, it is also extremely difficult to ensure that users exhaustively formulate their beliefs *a priori* (see [OP87] for further details).

The EVA framework, in contrast, generates its own expectations with respect to evidence in the form of news reports and background knowledge. These rules do not require the intervention of an expert in order to formulate them. Whilst the EVA framework does not preclude the inclusion of expert-generated expectations in the knowledgebase its strength rests mainly on the fact that it does not rely on expert-generated expectations.

General impressions	**The EVA Framework**
Attempts to *extract* the user's *a priori* expectations in order to identify *interesting rules.*	*Generates expectations* from news and background information that can be used to identify *unexpected information.*

The expectations generated by an EVA system are based on the analysis of a stream of news reports and so represent the usual state of the world, as reported in those news stories. However, as discussed in Chapter 1, the user's view of what is surprising may not be the same as that predicted by the frequency of reported events. It may be possible to extract some of the user's expectations via the general impressions approach. However, as mentioned above, the general impressions approach is not well suited to wide domains.

8.2.3 Deviations

In [PSM94], Piatetsky-Shapiro and Matheus introduce the Key Findings Reporter (KEFIR) system. KEFIR is a system that was originally applied to healthcare data mining to identify value

measures and the impact of deviations in those measures. KEFIR's strength is in identifying interesting trends and anomalous values, as these can then be used to control costs or improve patient outcomes, for example. Deviations are considered interesting proportional to the estimated possible benefit that would arise from taking some action based on that deviant value.

The KEFIR system is a method of identifying data that is unexpected and of using that unexpectedness as an analogue of interest. The KEFIR system succeeds as a method for identifying actionable quantitative measures, but its focus on numerical values means that it is unable to cope with qualitative data.

The Deviations approach	The EVA Framework
Forms a range of *expected values* based on the statistical analysis of *qualitative* data.	Generates a set of *expectations* based on the analysis of *logical formulae*.

The EVA framework can cope well with qualitative data but does have a shortcoming with regard to quantitative data: a range of expected values must be expressed as the consequent of an expectation in order to identify unexpected numbers in the EVA framework. Therefore the deviations approach exploited by a system such as KEFIR may also be of benefit as a complement to an EVA system.

8.3 Techniques investigated for use in the EVA framework

There is also a body of literature that has been used in creating the EVA framework. Some of this, such as the Event Calculus, has been used directly. Details of how these have been used appear in the preceding chapters. However there are other techniques that have been investigated but not employed such as inconsistency measurement, information theory and machine learning.

8.3.1 Wider results concerning the measure of inconsistency

The EVA framework is unusual in that, rather than trying to eliminate or measure inconsistency, the aim is to identify inconsistency between facts and rules and, where this inconsistency arises to measure unexpectedness by the strength of the rule that is violated. However approaches to measuring inconsistency were examined in the hope that they may provide some insight into measuring the unexpectedness of information.

Several approaches for measuring or ranking inconsistency exist: in the diagnostic systems literature 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 reject 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. Likewise they are concerned mainly with the elimination of inconsistency, which is not the aim of the EVA framework.

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], egalitarist 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 as again the approaches aim to eliminate, rather than to identify and exploit inconsistency.

A general characterization of inconsistency has been based on quasi-classical logic, which is a form of paraconsistent logic with a more expressive semantics than Belnap's four-valued logic. Inconsistent knowledge is analysed by considering the conflicts arising in the minimal quasi-classical models for that knowledge. These models are the basis of a measure of coherence for each knowledgebase, and of 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 help in evaluating inconsistencies arising through violations of expectations. In the EVA approach, we evaluate the inconsistency on the basis of the expectation rather than all the formulae involved in the inconsistency.

8.3.2 Information theory

The notion of measuring interesting information may be thought as of falling within the purview of the wider problem of measuring information content. Indeed, [HH01] demonstrates that measures of information content are among the best ways of identifying interesting discoveries in data mining. The concept of measuring information was introduced by Claude Shannon in 1948 (the paper is reproduced in [Sha93]). His formula for the information content, $I(E)$, of a message states that the information value of a message decreases as the probability of that message

occurring, $P(E)$, increases. This is captured by the following formula:

$$I(E) = -logP(E)$$

Thus each message represents a point in the space of all possible messages. The more improbable a message is, the more possibilities it eliminates, making it more informative. Therefore, unexpected news is considered to have more information content than expected news.

The work on information theory that has the most resonance with the EVA framework is presented by Lozinskii in [Loz94a]. Lozinskii presents a measure of the information content of *consistent formulae*, based on the supposition that information increases with the addition of formulae which do not render the set inconsistent and which are not logical consequences of the existing set.

Applying Shannon's measure of information, Lozinskii proposes that the information in a set of propositional formulae Γ, composed of n different atom symbols, is the logarithm of the number of models (2^n) divided by the number of models for the maximally consistent subsets of Γ [Loz94a]. This information theoretic measure increases with additions of consistent information and decreases with additions of inconsistent information.

Definition 8.3.1. *For a set S with the equivalence classes of the models of S, $Mod(S)$, and the number of variables in S, nThe quantity of information, $I(S)$ is:*

$$I(S) = n - log|Mod(S)|$$

In [Loz94b], Lozinskii goes on to describe how this measure can be applied to an inconsistent set S. For an inconsistent set of formulae the quantity of information in the set does not increase monotonically with the introduction of new formulae, as, in addition to adding new models, the new formula may 'fracture' the set into a greater number of maximally consistent subsets. The end effect is that each maximally consistent subset imparts information as to the possible values for variables (the equivalence classes of models), but as the number if maximally consistent subsets increases, the less certain we can be that we can pick a model from any particular one of these sets. In order to model this, Lozinskii devises a formula which links the number of maximally consistent subsets of S with the models of each of these maximally consistent subsets called *quasi-models*.

Wong and Besnard [WB01] point out that the Lozinskii's measure is syntax sensitive and it is affected by 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 $\{\alpha, \neg\alpha, \beta\}$.

Lozinskii's work is interesting for the insight it gives us into how inconsistency may affect in-

formation content. However, within the EVA framework, if an inconsistency arises between a representative set and an expectation it is not necessary to search for maximally consistent subsets of the knowledge base. We allow the expectation to be inconsistent with the representative set and instead use the strength of that expectation as a measure of the unexpectedness of the facts in the representative set.

8.4 Working expectations generator versus machine learning

Machine learning can be thought of as consisting of three main approaches: supervised learning, unsupervised learning and reinforcement learning. Supervised learning approaches, such as FOIL [Qui90] rely on a training set that contains examples of the concepts that are to be learnt. If we wish to use such a method to learn to differentiate between expected and unexpected news, we must present the learner with classified examples of interesting and uninteresting news reports. However, there is the difficulty that the features the learner uses to determine whether a report is interesting or not may be highly context dependent and specific to the articles in the training set. Also, as "what is expected" changes over time, the trained system may find itself providing predictions of unexpectedness that are out of step with reality. The process of retraining a unsupervised learning system is labour intensive. In contrast, an EVA system automatically updates the expectations it generates in the light of new information from reports and background knowledge. Therefore the EVA framework supports a system that is more responsive to change than a system that relies on supervised learning.

Unsupervised learning is learning in which there is no feedback from a trainer. Unsupervised learning is most often used in situations where labelling the training data is too expensive to be practical. The working expectation generator presented in Chapter 5 is a specialised form of unsupervised learning. There are several pre-existing techniques of unsupervised learning that address *clustering*. These include the k-means approach, which handles numerical data, in which each point assigned to nearest cluster iteratively; the fuzzy-c means clustering, which is similar to the k-means approach but clusters numeric points in a "fuzzy" fashion, rather than dividing the data points into crisp sets; and the hierarchical clustering approach which classifies data points using some distance measure. The hierarchical clustering approach could be applied to Dalal distances but, given that the distances between formulae are not of interest to the EVA framework, this approach too is not applicable to the EVA framework.

Inductive logic programming (ILP) is an unsupervised method of generating logical formulae that can generate rules that classify a set of examples. ILP was examined as a method for generating expectations for the EVA framework but found to be unsuitable in its present form. Whilst ILP is ideal for generating rules that divide data sets into crisp clusters, the EVA framework relies on a method that generates rules based on high probabilities. The ILP approach could be modified to generate fuzzy sets, rather than crisp sets, resulting in rules that capture a set of those examples

that is mainly, but not necessarily exclusively positive. The modification of ILP in order to make it suitable as a method for generating expectations rather than hard rules is a possible area for further exploration.

A third type of learning, reinforcement learning is used where preexisting labels are not required. The output from the learner is given a *post-hoc* quality rating by the user, but the learner does not have to be provided with a correct answer. Therefore a reinforcement learning method would not require the identification of interesting rules *a priori*. It is possible that the working expectation generator, which is at present an unsupervised learning approach, could be developed as a reinforcement method expectation by asking the user to rate the reports presented to them. Incorporating reinforcement into the working expectation generator may be addressed in future work.

Given the highly novel nature of the EVA framework there is little literature that is directly relevant to the approach presented in this thesis. However the literature presented here helps to better delineate the scope of the EVA framework as a method of (i) generating expectations that are representative of highly probably behaviour in the real world in a fashion that is distinct from other methods of machine learning and, (ii) identifying information that is inconsistent with those expectations in a way that is distinct from other methods of identifying and measuring inconsistency or unexpectedness and (iii) recommending news reports to users in a way that is unlike the mainly relevance-focussed methods of information retrieval and filtering.

Chapter 9

Conclusions and Discussions

9.1 Discussion

The identification and evaluation of interesting news is a potentially valuable goal. Much work has already been done to address information overload in non-news information. However, news presents two particular challenges: firstly, the identification and evaluation of news must be done in a timely fashion, as the usefulness of news declines rapidly over time. Secondly, the interest that arises from news is due at least in part to the unexpectedness of the information. Current approaches do not address unexpectedness and instead rely on measures of relevance to a query (information retrieval and filtering) or on similarity of tastes (collaborative filtering).

The Expectation Violation Analysis (EVA) framework presented in this thesis addresses the challenges presented by news by looking for unexpected information with respect to a pre-generated set of expectations. The framework accepts as input news reports in the form of first order logic facts, background knowledge as first order logic facts and rules and expectations as first order logic rules. Identifying unexpected information is then a matter of identifying inconsistencies between a consistent set of facts from a news report and relevant background knowledge and one or more expectations.

In addition to identifying interesting news, expectations also indicate the degree of unexpectedness of an item of news. Each expectation also has an *accuracy* and a *validity* value that indicate how representative that expectation is of events in the real world. Each expectation also has a *coverage* value that indicates the strength of the evidence for that expectation's accuracy and validity. These measures are based on evidence from news reports and background knowledge and, as such, have a meaningful derivation and clear semantics.

There is a partial ordering over antecedents and consequents that provides much useful infor-

mation regarding the relative strengths of the fired values of antecedents and the attacked and supported values of consequents. The higher an antecedent is in the antecedent order, the easier it is to fire that antecedent. Likewise, the higher a consequent is in the consequent order, the easier it is to support and the harder it is to attack that consequent.

It has been proved that logically equivalent expectations may have different coverage, accuracy and validity values, depending on the order of antecedents and consequents. Some decision must then be made to decide which of each set of equivalent expectations should be used in order that comparisons between expectations should be equitable. Therefore the expectations that are used to evaluate news are all of the form of a single literal consequent and an antecedent that is a conjunction of one or more literals.

The set of all expectations in this form is large for even quite limited languages. It is therefore not viable to search the entire set on receipt of each news report. A heuristic is required in order that reports can be evaluated in a timely fashion. The approach presented in this thesis is to identify a set of working expectations. This is a set of expectations that is small enough to be searched on receipt of a news report but that contains all those expectations that, when violated, indicate the occurrence of an interesting event. Therefore all members of the set of working expectations must have high coverage and high accuracy and/or validity.

Chapter 5 presents a method for generating sets of working expectations by identifying antecedents with high coverage values and then generating a set of highly accurate/valid expectations from those antecedents. Chapter 6 demonstrates the viability of this approach to generating a set of working expectations. The simulations show that it is possible to reliably create a set of working expectations with a small number of representative sets and in a universe with a large number of expectations. Chapter 6 also examines the possibility of biasing the working expectation generator towards generating expectations with longer antecedents. This is possible, but at the cost of decreased computational efficiency.

Chapters 2 through Chapter 6 present a framework that makes it possible to develop a system that can identify and evaluate unexpected information in a timely manner. Thus the EVA framework meets the challenge of identifying and evaluating interesting news in order to recommend that news to a reader.

Chapter 7 presents a means by which the background knowledge in an EVA system can also incorporate knowledge of sequences of events and states. Event and state data is a highly useful addition to the background knowledge of an EVA system, as it is then possible to develop expectations that apply only to entities that are in given states. The approach presented in this thesis is a modified version of the Event Calculus, first developed by Kowalski and Sergot in [KS85]. I have proposed a modification to the formalisation to include the ability to interpolate missing information about events. However, the event model is modular, and any approach that permits the representation of states and events in first order logic could be used in its place.

Chapter 8 critically examines work that is related to that presented in this thesis and discusses the strengths and weaknesses of the EVA framework with respect to these other approaches. The EVA framework is unique in that it addresses the issue of unexpectedness in news and also in that it makes it possible to present the user with a justification of the decision to rate news as interesting. Furthermore, a system based on the EVA framework is able to do so in a timely, computationally viable way. The EVA framework does not attempt to address the issue of relevance, but it is envisaged that an EVA system could work alongside a traditional information retrieval or filtering method which would address relevance as a pre- or post-filtering operation.

Machine learning techniques could potentially be used to search for good expectations, but the working expectation generator presented in this thesis, that exploits the properties of the set of expectations, works well as a heuristic for reducing the set of expectations to be searched. Information theory and the measurement of inconsistencies were examined for use as measures of unexpectedness, but measuring unexpectedness by means of the strength of expectations has a clear semantics and is easily derivable from news reports and background knowledge. Therefore the work presented in this thesis is a novel and worthwhile contribution to the state of the art in interest detection and evaluation.

9.2 Areas for further research

Overall the EVA framework provides a novel approach to identifying interesting news stories. It addresses the specific challenges posed by news: the timeliness of recommendations and the identification of unexpected information. As such it is a viable basis from which to develop a more sophisticated solution. The areas for refinement described below suggest ways in which such development may take place.

There are several areas of work that merit further exploration. These areas are those where a richer, more expressive formalisation may be possible, where a more computationally viable approach may be taken or both.

Cohort violations were presented in Section 3.3.1. However, for the sake of clarity, the rest of this thesis has concentrated solely on singular violations of expectations. Future work to determine the effect of cohort size on expectation strength would be valuable: as more members of a cohort violate an expectation, that expectation becomes weaker. So, although it is reasonable to expect that the emergence of a cohort would increase the degree of interest in a set of events, under the current definition the estimate of interest decreases. One way of addressing this issue would be through a set of simulations that examines whether delaying the updating of expectation values in order to give a cohort time to emerge would be useful.

The full ramifications of the state interpolation operators are yet to be examined. As the event model is a useful but not a necessary element of the EVA framework and the event interpolation

operators are a useful but not a necessary element of the event calculus, it was decided not to pursue the formalisation of these operators any further. However, further work would address the validity and computability of the state interpolation operators and the applicability of these operators within the EVA framework.

The difficulty of biasing the working expectation generator towards expectations with long antecedents is yet to be resolved. The method suggested in Chapter 6 must evaluate all of the expectations in E that have conjunctive antecedents and single literal consequents. Even for relatively small languages, this results in a very large space to explore. Likewise there remain difficulties with the efficiency of the working expectation generator algorithm. At present the algorithm uses deeply nested iterations (up to four levels deep). Such deep nesting results in an algorithm that is computationally expensive. However, there may be other, more efficient, ways of searching these expectations arising from the machine earning literature. Further work could lead to the identification of other potential algorithms from the machine learning field. A comparative study of these methods would look at the computational viability of these algorithms and their ability to generate the most useful expectations.

The question of identifying expectations that are rarely fired but highly useful also remains open: where there are the relatively infrequent occurrence of some combinations of predicate symbols there will be expectations that are relatively rarely fired. Nevertheless, some of these expectations may be particularly useful in identifying interesting news. Further work would identify an alternative learning strategy that would identify these expectations based on very little evidence. More sophisticated statistical techniques, or the development of some heuristics may be of use here.

Throughout the thesis expectations have been defined as unground (or deground) formulae, containing no constants. However, the approach to inductive generalisation taken by Plotkin [Plo70, Plo71], in which constants are substituted for variables wherever there is a difference in the nomenclature of those constants, may present us with a more flexible approach. To illustrate, consider the following representative sets:

$$\rho_1 = \texttt{companyType(ryanair,airline)}, \texttt{profitable(ryanair)}$$

$$\rho_2 = \texttt{companyType(ba,airline)}, \texttt{profitable(ba)}$$

the approach taken in this thesis has been to completely deground ρ_1 and ρ_1 in order to relate them to some expectation:

$$\epsilon = \forall \texttt{x companyType(x,y)}, \texttt{profitable(x)}$$

However, this results in the loss of some important information: the fact that both of these profitable companies are airlines. Plotkin's approach is to substitute variables for only those constant

symbols that differ. This would result in the following expectation:

$$\epsilon = \forall \mathtt{x}\ \mathtt{companyType(x, airline), profitable(x)}$$

Taking such an approach would vastly increase the size of the set of potential expectations. However it would lead to a more expressive set of expectations. Further work would demonstrate whether this inductive generalisation approach could be exploited by the expectation generator, thus resulting in a richer set of expectations.

Currently all representative sets are given equal weight when determining the accuracy of an expectation. If later representative sets are given higher weighting than older ones the set of working expectations will be more representative of current than long term trends. Further work would determine whether and how best to weight representative sets in order that more recent information has a greater influence on expectation values that older information. Simulations may help to determine how these weights should decline over time (linearly, exponentially...) and examination of real world data should suggest the frequency with which trends tend to alter and thus what rates of decay should be applied to those weights.

In this thesis, the possibility of "typing" constant symbols has not been addressed. It may be that typing the elements of the language used by an EVA framework would enable the identification of expectations that generalise by type. This would result in greater expressive power.

Example 9.2.1. *Let* `britishairways` *be a predicate symbol of type* `airline`. *Rather than generalising the statement* `profitable(britishairlines)` *to* $\forall$`x profitable(x)`, *it would be possible to generalise by type:* $\forall \mathtt{x} \in$ `airline profitable(x)`.

Currently, some typing information is incorporated by means of predicates such as `airline(britishairways)` in the set of background knowledge. Whether introducing explicit types would increase expressive power without an excessive increase in complexity is a question that deserves further study.

9.3 Contributions

To recap the contributions presented in Figure 1.2 on page 15, the major contributions of this thesis are:

A **framework** that is of use in identifying and evaluating interesting news in a timely fashion, presented in Chapters 2 and 3. Such a framework is presented here, with the identification of inconsistency between news, background knowledge and expectations as the indicator of interesting information. The framework incorporates news reports as a set of first order

logic facts, background knowledge as a set of first order logic facts and rules and expectations as a set of first order rules. Also presented is a directed method for identifying such inconsistencies, by means of viaducts.

A theoretical analysis of the set of all possible expectations for a language. **Properties of the set of expectations** (accuracy, validity and coverage) are defined and their relation to the relative order of expectations is proved in Chapter 4.

A **method of generating a working subset of the expectation set**, by exploiting the properties of the set of expectations is defined in Chapter 5.

The **Empirical work** presented in Chapter 6 demonstrates that this approach generates a useful subset of expectations.

Minor contributions are:

Modifications to the Event Calculus that support **reasoning about missing events** in narratives (Chapter 7). Such a modification is necessary when dealing with streams of news reports as it is not possible to assume that all events will be reported.

Justification of the EVA framework *per se*, and with respect to existing findings in the literature can be found in Chapter 8.

Appendix A

Key definitions

Definition	Reference	Page
Grounding set	2.2.5	18
Grounding	2.2.6	18
Access rule	2.3.4	21
access(ρ, Γ)	2.3.5	21
match(ρ, Γ, Δ)	2.4.4	27
representatives(ρ, Γ, Δ)	2.4.5	28
Expectation	2.5.1	29
Antecedent	3.1.1	33
Consequent	3.1.2	34
Fired	3.1.3	34
Attacked	3.1.4	35
Supported	3.1.5	36
Violated	3.1.6	37
Confirmed	3.1.7	37
fset$(\epsilon, \Pi, \Gamma, \Delta)$	3.2.1	39
aset$(\epsilon, \Pi, \Gamma, \Delta)$	3.2.2	39
sset$(\epsilon, \Pi, \Gamma, \Delta)$	3.2.3	39
vset$(\epsilon, \Pi, \Gamma, \Delta)$	3.2.4	40
cset$(\epsilon, \Pi, \Gamma, \Delta)$	3.2.5	40
accuracy$(\epsilon, \Pi, \Gamma, \Delta)$	3.2.6	40
validity$(\epsilon, \Pi, \Gamma, \Delta)$	3.2.7	41
Order over values	3.2.8	41
Marker formulae	4.1.1	57
		(cont...)

Definition	Reference	Page
Implication order ($\preceq$)	4.1.2	57
The set of expectations (E)	4.1.3	58
Non-firing expectation	4.2.1	70
Automatically violated expectation	4.2.2	70
Automatically firing expectation	4.2.3	71
Non-violating expectation	4.2.4	71
Self-defeating expectation	4.2.5	71
Self-reinforcing expectation	4.2.6	71
Literal layer	4.3.1	72
coveredAntecedents($M, \Pi, \Gamma, \Delta, minCoverage$)	5.1.1	81
goodExpectations($A^n, \Pi, \Gamma, \Delta, minCoverage$)	5.1.2	82
workingExpectations($A^n, \Pi, \Gamma, \Delta, minCoverage, minAccuracy$)	5.1.3	83
makeGoodExpectations($literals, repSets, minCoverage$)	5.2.11	97
Full convergence	6.1.2	116
Partial convergence	6.1.3	117
Canonical set	6.1.4	117
initiates and terminates rules	7.2.1	147
holds rules	7.2.2	147
start and end rules	7.2.3	148
exclusive and incompatible rules	7.2.4	149
broken rules	7.2.1	149
State model	7.2.7	150
Event rule	7.3.1	152
precedes rule	7.5.1	157
follows rule	7.5.2	159
State interpolating start and end rules	7.5.3	160
Intervening states	7.5.4	162

Appendix B

List of Sets and Symbols

	Individual	Set
Predicate Symbols	$p_1, ..., p_2$	P
Function Symbols	$f_1, ..., f_n$	F
Constant Symbols	$a, b, c_1, ..., c_n$	C
Variable Symbols	$v_1, ..., v_n, x, y, z$	V
Terms	$t_1, ..., t_n$	T
Literals	α, β, γ	
Groundings	$x = a, y = b...$	Φ
Report Atoms	$\rho_1, ..., \rho_n$	Π
Access Rules		Γ
Background Knowledge		Δ
Unground Formulae	$\forall \bar{x}\ \alpha, \forall \bar{y}\ \beta...$	M
Expectations	$\epsilon_1, ..., \epsilon_n$	E
Antecedents	$\mathsf{antecedent}(\epsilon_1), ..., \mathsf{antecedent}(\epsilon_n)$	
Consequents	$\mathsf{consequent}(\epsilon_1), ..., \mathsf{consequent}(\epsilon_n)$	

Bibliography

[AA93] Paolo Atzeni and Valeria De Antonellis. *Relational database theory*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1993.

[ACD+98] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218, 1998.

[Bal97] Marko Balabanovic. An adaptive web page recommendation service. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 378–385, New York, 5–8, 1997. ACM Press.

[BBJ02] G Boolos, J Burgess, and R Jeffrey. *Computability and Logic*. Cambridge University Press, 2002.

[BC92] Nicholas J. Belkin and Bruce W. Croft. Information filtering and retrieval: Two sides of the same coin? *Artificial Intelligence*, 35(12):29–38, December 1992.

[BCCN94] John Broglio, James P. Callan, W. Bruce Croft, and Daniel W. Nachbar. Document retrieval and routing using the INQUERY system. In *Text REtrieval Conference (TREC-3)*, pages 22–29, 1994.

[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.

[BH04] E. Byrne and A. Hunter. Man bites dog: Looking for interesting inconsistencies in structured news reports. *Data and Knowledge Engineering*, 48(3):265–285, 2004.

[BH05] E. Byrne and A. Hunter. Evaluating violations of expectations to find exceptional information. *Data and Knowledge Engineering*, In Press, 2005.

[BHH+00] M. Braschler, D. Harman, M. Hess, M. Kluck, C. Peters, and P. Schuble. Evaluation of systems for cross-language information retrieval. In *Second International Conference on Language Resources and Evaluation (LREC). Athens, Greece*, 2000.

[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.

[BVSL02] Maria J. Martin Bautista, Mara-Amparo Vila, Daniel Sànchez, and Henrik L. Larsen. Intelligent filtering with genetic algorithms and fuzzy logic. In B. Bouchon-Meunier et al, editor, *Technologies for constructing intelligent systems*, pages 352–377. Physica-Verlag GmbH, 2002.

[Byr05] E. Byrne. Because men don't bite dogs: a logical framework for identifying and explaining unexpected news. In *Workshop on Explanation Aware Computing (EXACT 2005)*. AAAI fall symposium, 3-6 November 2005. To appear.

[Cal98] Jamie Callan. Learning while filtering documents. In *Proceedings of the ACM SIGIR Conference*, pages 224–231, 1998.

[CBLJ04] D.P. Corney, B.F. Buxton, W.B. Langdon, and D.T. Jones. Biorat: extracting biological information from full-length papers. *Bioinformatics*, 20(17):3206–13, November 2004.

[CCH92] James P. Callan, W. Bruce Croft, and Stephen M. Harding. The INQUERY retrieval system. In *Proceedings of DEXA-92, 3rd International Conference on Database and Expert Systems Applications*, pages 78–83, 1992.

[Cha99] Philip K. Chan. A non-invasive learning approach to building web user profiles. In *KDD99 Workshop on Web Usage Analysis and User Profiling*, 1999.

[Che95] H. Chen. Machine learning for information retrieval: Neural networks, symbolic learning, and genetic algorithms. *Journal of the American Society for Information Science*, 46(3):194–216, April 1995.

[Cir01] F. Ciravegna. Challenges in information extraction from text for knowledge management. *Intelligent Systems and Their Applications*, 16(6):88–90, 2001.

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

[Coo00] R. Cooley. Web usage mining: Discovery and application of interesting patterns from web data, 2000.

[CWL99] Sally J. Cunningham, Ian H. Witten, and J. Littin. Applications of machine learning in information retrieval. *Annual Review of Information Science*, 34:341–384, 1999.

[Dal88] Mukesh Dalal. Investigations into a theory of knowledge base revision: Preliminary report. In Paul Rosenbloom and Peter Szolovits, editors, *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 475–479, Menlo Park, California, 1988. AAAI Press.

[DDL+90] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[EG95] T Eiter and G Gottlob. The complexity of logic-based abduction. *Journal of the ACM*, 42:3–42, 1995.

[FMST01] Mary F. Fernandez, Atsuyuki Morishima, Dan Suciu, and Wang Chiew Tan. Publishing relational data in XML: the silkroute approach. *IEEE Data Engineering Bulletin*, 24(2):12–19, 2001.

[FO04] Bahadorreza Ofoghi Farhad Oroumchian, Ehsan Darrudi. XML information retrieval by means of plausible inferences. In *5th International Conference on Recent Advances in Soft Computing*, pages 542–547, Nottingham UK, 2004. RASC 2004.

[Foc01] S. Focardi. Clustering economic and financial time series: exploring the existence of stable correlation conditions. Technical report, The Intertek Group, 2001.

[Fra92] W. B. Frakes. Stemming algorithms. In W.B. Frakes and R. Beza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, 1992.

[FS00] Wenfei Fan and Jerome Simeon. Integrity constraints for XML. In *Symposium on Principles of Database Systems*, pages 23–34, 2000.

[Fuh92] Norbert Fuhr. Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255, 1992.

[Fuh01] Norbert Fuhr. Models in information retrieval. In F Crestani M Agosti and G Pasi, editors, *Lecture Notes in Computer Science 2001*. Springer-Verlag, Berlin, 2001.

[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.

[GJ79] M Garey and D Johnson. *Computers and Intractability*. W H Freeman, 1979.

[HDZS01] Xiaofeng He, Chris H. Q. Ding, Hongyuan Zha, and Horst D. Simon. Automatic topic identification using webpage clustering. In *IEEE International Conference on Data Mining*, pages 195–202, 2001.

[HH01] Robert J. Hilderman and Howard J. Hamilton. Evaluation of interestingness measures for ranking discovered knowledge. *Lecture Notes in Computer Science*, 2035:247–259, 2001.

[HM86] Steve Hanks and Drew McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In *Proceedings of the Fifth National (U.S.) Conference on Artifcial Intelligence*, pages 328–333. Morgan Kaufman, 1986.

[HS04] A Hunter and R Summerton. A knowledgebased approach to merging information. *(Submitted)*, 2004.

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

[Hun00b] A Hunter. Merging potentially inconsistent items of structured text. *Data and Knowledge Engineering*, 3:305–332, 2000.

[Hun00c] A Hunter. Ramification analysis using causal mapping. *Data and Knowledge Engineering*, 32:1–27, 2000.

[Hun00d] 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.

[IG01] N. Ikizler and H. A. Gvenir. Mining interesting rules in bank loans data. In A. Acan et al, editor, *Proceedings of the Tenth Turkish Symposium on Artificial Intelligence and Neural Networks*, pages 238–246, Gazimagusa, Cyprus, 2001. TAINN.

[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] Robert A. Kowalski and Marek J. Sergot. A logic-based calculus of events. In *Foundations of Knowledge Base Management (Xania)*, pages 23–55, 1985.

[KS97] Robert Kowalski and Fariba Sadri. Reconciling the situation calculus and event calculus. *Journal of Logic Programming*, 31:39–58, 1997.

[KW87] J De Kleer and B Williams. Diagnosing mulitple faults. *Artificial Intelligence*, 32:97–130, 1987.

[Lal97] M. Lalmas. Dempster-shafer's theory of evidence applied to structured documents: modelling uncertainty. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 110–118, 1997.

[LB98] M. Lalmas and P.D. Bruza. The use of logic in information retrieval modelling. *Knowledge Engineering Review*, 13:1–33, 1998.

[LHC97] Bing Liu, Wynne Hsu, and Shu Chen. Using general impressions to analyze discovered classification rules. In *Knowledge Discovery and Data Mining*, pages 31–36, 1997.

[LLH05] W. Liu, Z. Liao, and J. Hong. Query cost estimation through remote system contention state analysis over the internet. *Web Intelligence and Agent Systems: an International Journal*, To appear, 2005.

[LM98] J. Lin and A.O. Mendelzon. Merging databases under constraints. *International Journal of Cooperative Information Systems*, 7(1):55–76, 1998.

[Loz94a] E. Lozinskii. Information and evidence in logic systems. *Journal of Experimental and Theoretical A.I.*, 6:163–193, 1994.

[Loz94b] E. Lozinskii. Resolving contradictions: A plausible semantics for inconsistent systems. *Journal of Automated Reasoning*, 12:1–31, 1994.

[LV04] M. Lalmas and P. Vannoorenberghe. Indexation et recherche de documents XML par les fonctions de croyance (modelling XML retrieval with belief functions). In *Premiere Conference en Recherche d'Information et Applications (CORIA'O4)*, 2004.

[McC99] J. McCarley. Should we translate the documents or the queries in cross-language information retrieval? In *37th Annual Meeting of the Association for Computational Linguistics*, pages 208–214, 1999.

[MH69] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[Mor90] Leora Morgenstern. Knowledge and the frame problem. *International Journal of Expert Systems*, 3(4):309–343, 1990.

[MS94] Rob Miller and Murray Shanahan. Narratives in the situation calculus. *Journal of Logic and Computation*, 4(5):513–530, 1994.

[MS96] Rob Miller and Murray Shanahan. Reasoning about discontinuities in the event calculus. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR96)*, pages 63–74. Morgan Kaufmann, 1996.

[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.

[OP87] A. Ortony and D. Partridge. Surprisingness and expectation failure, what's the difference? In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 106–108, 1987.

[OY01] Masayuki Okabe and Seiji Yamada. Interactive document retrieval with relational learning. In *Proceedings of the 2001 ACM symposium on Applied computing*, pages 27–31, 2001.

[PA00] R. Papka and J. Allan. Topic detection and tracking: Event clustering as a basis for first story detection. In W. B. Croft, editor, *Advances in Information Retrieval*, pages 97–126. Kluwer Academic Publishers, 2000.

[PB97] Michel Pazani and Daniel Bilsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331, 1997.

[PC98] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Research and Development in Information Retrieval*, pages 275–281, 1998.

[Plo70] G D Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.

[Plo71] G. D. Plotkin. A further note on inductive generalization. *Machine Intelligence*, 6:101–124, 1971.

[Plo93] S. Plous. *The psychology of judgment and decision making*. McGraw-Hill, London - New York, 1993.

[POT69] Chaim Perelman and L. Olbrechts-Tyteca. *The New Rhetoric: A Treatise on Argumentation*. University of Notre Dame, 1969.

[PR93] J. Pinto and R. Reiter. Temporal reasoning in logic programming: A case for the situation calculus. In *Proceedings of the International Conference on Logic Programming*, pages 203–221, 1993.

[Pro96] Alessandro Provetti. Hypothetical reasoning: From situation calculus to event calculus. *Computational Intelligence Journal*, 12(3):478–498, 1996.

[PSM94] G. Piatetsky-Shapiro and C. Matheus. The interestingness of deviations. In *KDD-94*, Seattle, WA, 1994.

[PT98] Balaji Padmanabhan and Alexander Tuzhilin. A belief-driven method for discovering unexpected patterns. In *Knowledge Discovery and Data Mining*, pages 94–100, 1998.

[PTRV98] Christos H. Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS98)*, pages 159–168, 1998.

[Qui90] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[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.

[Rie94] P. Resnick N. Iacovou M. Suchak P. Bergstorm J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[RIS+94] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[RLDL97] B. Rehder, M.L. Littman, S. Dumais, and T.K. Landauer. Automatic 3-language cross translation information retrieval with latent semantic indexing. In *Proceedings of TREC6*, pages 233–239. NIST, 1997.

[RR01] J. F. Roddick and S. Rice. What's interesting about cricket? - on thresholds and anticipation in discovered rules. *SIGKDD Explorations*, 3(1):1–5, 2001.

[Sah99] Sigal Sahar. Interestingness via what is not interesting. In *Knowledge Discovery and Data Mining*, pages 332–336, 1999.

[SC95] M Schafer and M Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74:249–310, 1995.

[Sch79] R. Schank. Interestingness: Controlling inferences. *Artificial Intelligence*, 12(3):273–297, 1979.

[Sch90] L.K. Schubert. Monotonic solution of the frame problem in the situation calculus: An efficient method for worlds with fully specified actions. In R. Loui H. Kyburg and G. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer, 1990.

[Sek03] Yohei Seki. Sentence extraction by tf/idf and position weighting from newspaper articles. In *Proceedings of the Third NTCIR Workshop on research in information Retrieval, Automatic Text Summarization and Question Answering*, 2003.

[Sen98] Jean Senellart. Locating noun phrases with finite state transducers. In *COLING-ACL*, pages 1212–1219, 1998.

[SGD+02] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble, and H. Levy. An analysis of internet content delivery systems. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation (OSDI).*, 2002.

[Sha93] C. E. Shannon. A mathematical theory of communication. In N. J. A. Sloane and A. D. Wyner, editors, *Claude Elwood Shannon: Collected Papers*, pages 5–83. IEEE Press, New York, 1993.

[Sha97] Murray Shanahan. Event calculus planning revisited. In Sam Steel and Rachid Alami, editors, *Recent Advances in AI Planning, 4th European Conference on Planning, ECP'97, Toulouse, France,*, Lecture Notes in Computer Science, pages 390–395. Springer, 1997.

[SHWA03] M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. Using predicate-argument structures for information extraction. In *Proceedings of the ACL, Sapporo, Japan*, pages 8–15. ACL, 2003.

[SKKR00] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce*, pages 158–167, 2000.

[SKR01] J. Ben Schafer, Joseph A. Konstan, and John Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1–2):115–153, 2001.

[SPK00] Ingo Schwab, Wolfgang Pohl, and Ivan Koychev. Learning to recommend from positive evidence. In *Intelligent User Interfaces*, pages 241–247, 2000.

[SPU02] Andrew I. Schein, Alexandrin Popescul, and Lyle H. Ungar. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

[SS94] L Sterling and E Shapiro. *The Art of Prolog*. MIT Press, 1994.

[ST95] Abraham Silberschatz and Alexander Tuzhilin. On subjective measures of interestingness in knowledge discovery. In *Knowledge Discovery and Data Mining*, pages 275–281, 1995.

[ST96] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Trans. On Knowledge And Data Engineering*, 8:970–974, 1996.

[TH01] L. Terveen and W. Hill. Beyond recommender systems: Helping people help each other. In J. Carroll, editor, *HCI in the New Millennium*, pages 487–509. Addison Wesley, 2001.

[TKS02] P. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 32–41, 2002.

[UF98] L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *Proceedings of the Workshop on Recommendation Systems. AAAI Press, Menlo Park California*, 1998.

[Voo99] Ellen M. Voorhees. Natural language processing and information retrieval. In *SCIE*, pages 32–48, 1999.

[War92] S. Wartik. Boolean operations. In W.B. Frakes and R. Beza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 264–292. Prentice Hall, 1992.

[WB01] P. Wong and P. Besnard. Paraconsistent reasoning as an analytic tool. *L.J. of the IGPL*, 9(2):217–229, 2001.

[YCB+99] Yiming Yang, Jaime Carbonell, Ralf Brown, Tom Pierce, Brian T. Archibald, and Xin Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, 14(4):32–43, 1999.

[ZCM02] Y. Zhang, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the ACM SIGIR*, pages 81–88, 2002.

[ZS01] Byoung-Tak Zhang and Young-Woo Seo. Personalized web-document filtering using reinforcement learning. *Applied Artificial Intelligence*, 15(7):665–685, 2001.

Index