Event Processing over Uncertain Data

Avigdor Gal, Segev Wasserkrug, and Opher Etzion

Abstract. Events are the main input of event-based systems. Some events are generated externally and flow across distributed systems, while other events and their content need to be inferred by the event-based system itself. Such inference has a clear trade-off between inferring events with certainty, using full and complete information, and the need to provide a quick notification of newly revealed events. Timely event inference is therefore hampered by the gap between the actual occurrences of events, to which the system must respond, and the ability of event-based systems to accurately infer these events. This gap results in uncertainty and may be attributed to unreliable data sources (*e.g.*, an inaccurate sensor reading), unreliable networks (*e.g.*, packet drop at routers), the use of fuzzy terminology in reports (*e.g.*, normal temperature) or the inability to determine with certainty whether a phenomenon has occurred (*e.g.*, declaring an epidemic). In this chapter we present the state-of-the-art in event processing over uncertain data. We provide a classification of uncertainty in event-based systems, define a model for event processing over uncertain data, and propose algorithmic solutions for handling uncertainty. We also define, for demonstration purposes, a simple pattern language that supports uncertainty and detail open issues and challenges in this research area.

1 Introduction

This chapter covers a topic of increasing interest in the event processing research and practice communities. Event processing typically refers to an approach to software systems that is based on event delivery, and that includes specific logic to filter,

Avigdor Gal
Technion – Israel Institute of Technology, Faculty of Industrial Engineering & Management, Technion City, 32000 Haifa, Israel
e-mail: avigal@ie.technion.ac.il

Segev Wasserkrug
IBM Haifa Research Lab, Haifa, Israel
e-mail: segevw@il.ibm.com

S. Helmer et al.: Reasoning in Event-Based Distributed Systems, SCI 347, pp. 279–304.
springerlink.com

transform, or detect patterns in events as they occur. A first generation of event processing platforms is diversified into products with various approaches towards event processing, including the stream oriented approach, such as: StreamSQL EventFlow (StreamBase),[30] CCL (Sybase),[31] Oracle CEP (Oracle),[24] and Stream Processing Language (IBM) [29]; the rule oriented approach that is implemented in products such as: AutoPilot M6 (Nastel),[4] Reakt (ruleCore),[28] TIBCO BusinessEvents (TIBCO),[33] and Websphere Business Events (IBM) [38]); the imperative approach is implemented in products such as Apama (Progress Software) [3] and Netcool Impact Policy Language (IBM) [23]; and, finally, the publish-subscribe approach that is part of Rendezvous (TIBCO),[32] Websphere Message Queue (IBM),[39] and RTI Data Distribution System (RTI).[27] As a common denominator, all of these approaches assume that all relevant events are consumed by the event processing system, all events reported to the system have occurred, and processing of event processing systems can be done in a deterministic fashion. These assumptions hold in many of the applications first generation platforms support.

Moving to the second generation of event processing platforms, one of the required characteristics is the extension of the range of applications that employ event processing platforms beyond the early adopters. Some of the target applications deviate from the basic assumptions underlying the first generation of event processing platforms in both functional and non-functional aspects. In this chapter we focus on a specific functional aspect, the ability to deal with inexact reasoning [11]. We motivate this requirement with two examples.

Example 1. An increasing number of event processing systems are based on Twitter feeds as raw events, by using either structured Twitter feeds, or using tags, *e.g.,* a bus notiyfing Twitter every time it arrives at a station;[1] automated trading decision applications, based on analysis of Tweets about traded companies,[2] *etc.*. One cannot assume that the collection of events sent to Twitter is complete. Furthermore, one cannot assume it is accurate, as some tweets may be of rumor types, some may contain inaccurate information, and some are even sent by malicious sources. Yet, in the highly competitive world of trading, Twitter events are considered to be a good source of information, given that the processing can take into account all these possible inaccuracies.

Example 2. A service provider is interested in detecting the frustration of a valued customer in order to mitigate the risk of the customer deserting. In some cases there is an explicit event where a customer calls and loudly expresses dissatisfaction, however in most cases this is inferred from detecting some patterns over the event history. Assume that practice (or applying some machine learning techniques) concluded that if a customer approaches the customer service center three times within a single day about the same topic, this customer is frustrated. The fact that the customer is frustrated is the business situation that the service provider wants

[1] http://twitter.com/hursleyminibus

[2] http://www.wallstreetandtech.com/data-management/showArticle.jhtml?articleID=218101018

to identify, while the pattern detected on the customer interaction event is only an approximation of this situation.

In the absence of support for inexact reasoning, applications as those described above may suffer, either directly or indirectly, from incorrect situation detection. Using current technology, there are four different ways to support such situations of uncertainty. First, situations of uncertainty can simply be ignored. Such a solution may be cost-effective if situations of uncertainty are relatively infrequent, and the damage of not handling them is not substantial. For example, in network management systems problem indications such as device time-out may be lost, but such events are not critical since they are issued on a recurring basis.

A second solution for handling uncertainty requires situations to be created only when the event pattern is a necessary and sufficient condition to detect the situation in a deterministic way. This is the case in many current systems. According to a third solution, the system is designed so that some detected situations require reinforcement from multiple indications. For example, in a fraud detection system, often a fraud suspicion requires reinforcement from multiple patterns, and possibly within the context of a customer's history. This is useful when false positives should be minimized, and it comes at the cost of false negatives.

Finally, it is possible to notify the result of each situation detection to a human observer that needs to decide whether an action should be taken. Again, this method is aimed at minimizing false positives.

The applications motivating a second generation of event processing platforms include applications in which false positives and false negatives may be relatively frequent, and the damage inflicted by these cases may be substantial, sometimes critical. For example, a stock value of a company may collapse if automated trading decisions are based on false rumors. While the motivation exists, the handling of inexactness in event processing in general is still a challenge in the current state-of-the-art and in this chapter we explore quantitative methods to manage inexactness.

The rest of the chapter is structured in the following way: Section 2 provides basic terminology to be used in this chapter; Section 3 provides a taxonomy of uncertainty cases and the handling of uncertain events is discussed in Section 4; Section 5 discusses the handling of uncertain situations and Section 6 describes an algorithm for uncertain derivation of events; We conclude with research challenges in Section 7.

2 Preliminaries

In this section we provide some basic concepts in event processing (Section 2.1) and uncertainty handling (Section 2.2). Throughout this chapter we shall use the following two examples for demonstration.

Example 3. A thermometer generates an event whenever the temperature rises above 37.5°C. However, the thermometer is known to be accurate to within ±0.2°C. Therefore, when the temperature measured by the thermometer is 37.6°C, there is some uncertainty regarding whether the event has actually occurred.

Example 4. Consider an e-Trading Web site, where customers can buy and sell stocks, check their portfolio and receive information regarding the current price of any stock. We would like to identify a variety of events, including that of speculative customers (illegal trading events) and customers becoming dissatisfied (CRM – Customer Relationship Management – related events).

2.1 Event Processing

We base our description of basic concepts in event processing on the model proposed by Etzion and Niblett [11].

An *event* is an occurrence within a particular system or domain; it is something that has happened, or is contemplated as having happened in that domain. The word event is also used to mean a programming entity that represents such an occurrence in a computing system. We classify events as either raw or derived events. A *raw event* is an event that is introduced into an event processing system by an external event producer. A *derived event* is an event that is generated as a result of event processing that takes place inside the event processing system. A derived event can be generated either by event transformation or event pattern matching. An event transformation transforms one or more event inputs into one or more event output by translation, enrichment from external data source, aggregation, composition or splitting. Example of event transformation is an aggregation that finds the average of a temperature measurement over a one hour shifting window.

An *event pattern* is a template, specifying one or more combinations of events. Given any collection of events one may be able to find one or more subsets of those events that match a particular pattern. We say that such a subset *satisfies* the pattern. An example of an event pattern is a sequence of events of type "buy stock" followed by an event of type "sell stock." The pattern matching process in this case creates pairs of events of these two types that match this pattern, satisfying all matching conditions (same stock, same customer, same day), defined to be the *pattern matching set*. A pattern matching set can serve as a *composite event*, composed of all member events in the event set, *e.g.*, the two matched event {buy stock, sell stock}. Alternatively, it can yield an event, created as some transformation of the matching set. For example, the event can be a newly created event type, containing some information from the payloads of both events, such as: ⟨customerid, buyamount, sellamount⟩.

A *situation* is an event occurrence that might require a reaction and is consumed by an external event consumer. In current systems, a situation can either be a raw event or a derived event, however, in some cases a derived event is merely an approximation of a situation. An example of a situation may be the detection of a speculative customer, with the condition that a speculative customer is a customer that buys and then sells the same stock on the same day, both transactions exceeding $1M.

The linkage between an event pattern and a situation might suffer from false positive or false negative phenomena. *False negative situation detection* refers to cases in which a situation occurred in reality, but the event representing this situation was not emitted by an event processing system. In the example discussed above,

the amount paid for a stock was recorded to be slightly less than $1M, and yet the customer was indeed a speculative customer. In this case, the event processing system did not detect it. *False positive situation detection* refers to cases in which an event representing a situation was emitted by an event processing system, but the situation did not occur in reality. In our example, the purchase was recorded erroneously, and was later corrected, yet the event system declared the customer to be speculative. Two of the main goals of an uncertainty handling mechanism for events are a) making explicit, and b) quantifying, the knowledge gap between an event pattern and the corresponding situation.

A variety of data can be associated with the occurrence of an event. Two examples of such data are: The point in time at which an event occurred, and the new price of a stock in an event describing a change in the price of a specific stock. Some data are common to all events (*e.g.*, their time of occurrence), while others are specific only to some events (*e.g.*, data describing stock prices are relevant only for stock-related events). The data items associated with an event are termed *attributes*. In what follows, $e.attributeName$ denotes the value of a specific attribute of a specific event e. For example, $e_1.occT$ refers to the occurrence time of event e_1. In addition, the type of event e is denoted by $e \in type$.

2.2 Uncertainty Management Mechanisms

There is a rich literature on mechanisms for uncertainty handling, including, among others, *Lower and Upper Probabilities*, *Dempster-Shafer Belief Functions*, *Possibility Measures* (see [17]), *Fuzzy Sets* and *Fuzzy Logic* [41]. We next present, in more details, three common mechanisms that were applied in the context of event processing, namely probability theory, fuzzy set theory, and possibility theory.

2.2.1 Probability Theory

The most well known and widely used framework for quantitative representation and reasoning about uncertainty is probability theory. An intuitively appealing way to define this probability space involves possible world semantics [13]. Using such a definition, a probability space is a triple $pred = (W, F, \mu)$ such that:

- W is a set of possible worlds, with each possible world corresponding to a specific set of event occurrence that is considered possible. A typical assumption is that the real world is one of the possible worlds.
- $F \subseteq 2^{|W|}$ is a σ-algebra over W. σ-algebra, in general, and in particular F, is a nonempty collection of sets of possible worlds that is closed under complementation and countable unions. These properties of σ-algebra enable the definition of a probability space over F.
- $\mu : F \rightarrow [0,1]$ is a probability measure over F.

We call the above representation of the probability space the *possible world representation*. One problem with possible worlds semantics is performance. Performing operations on possible worlds can lead to an exponential growth of alternatives. In

Section 5.1 we present an alternative representation to the possible worlds semantics and in Section 6 we use this alternative representation to compute efficiently probabilities of complex events.

The most common approach for quantifying probabilities are Bayesian (or belief) networks [25]. However, Bayesian networks are only adequate for representing propositional probabilistic relationships between entities. In addition, standard Bayesian networks cannot explicitly model temporal relationships. To overcome these limitations, several extensions to Bayesian networks have been defined, including Dynamic Belief Networks [19] , Time Nets [18], Modifiable Temporal Belief Networks [8] and Temporal Nodes Bayesian Networks [14]. Although these extensions are more expressive than classical Bayesian networks, they nonetheless lack the expressive power of first-order logic. In addition, some of these extensions allow more expressive power at the expense of efficient calculation.

Another formal approach to reasoning about probabilities involves probabilistic logics (*e.g.*, [5] and [16]). These enable assigning probabilities to statements in first-order logic, as well as inferring new statements based on some axiomatic system. However, they are less suitable as mechanisms for the calculation of probabilities in a given probability space.

A third paradigm for dealing with uncertainty using probabilities is the KBMC (Knowledge Based Model Construction) paradigm [7]. This approach combines the representational strength of probabilistic logics with the computational advantages of Bayesian networks. In this paradigm, separate models exist for probabilistic knowledge specification and probabilistic inference. Probabilistic knowledge is represented in some knowledge model (usually a specific probabilistic logic), and whenever an inference is carried out, an inference model would be constructed based on this knowledge.

2.2.2 Fuzzy Set Theory

The background on fuzzy set theory is based on [41, 12, 22]. A fuzzy set M on a universe set U is a set that specifies for each element $x \in U$ a degree of membership using a membership function

$$\mu_M : U \rightarrow [0,1]$$

For example, considering Example 3, the membership function that assigns a value to the reading of a thermometer can be represented as a bell shape over the range $[37.3°C, 37.7°C]$, with higher membership value in the center ($37.5°C$), slowly decreasing to 0 on both sides. It is worth noting that, unlike probability theory, the area under the curve does not necessarily sum to 1.

2.2.3 Possibility Theory

Possibility theory formalizes users' subjective uncertainty of a given state of the world [10]. Therefore, an event "customer x is frustrated" may be associated with

a confidence measure $\pi_{\text{frustrated}}(x)$. Both fuzzy set and possibility theories use a numerical measure, yet they express different uncertainties. Fuzzy set theory is more suitable to represent vague description of an object (*e.g.*, value of a temperature reading) and possibility measures define the subjective confidence of the state in the world (*e.g.*, the occurrence of an event).

In [22], two measures were defined to describe the matching of a subscription to a publication and can be easily adopted to event processing under uncertainty. The *possibility measure* (Π) expresses the plausability of an event occurrence. The *necessity measure* (N) expresses the necessity of occurrence of an event e or, formulated differently, the impossibility of the complement of e. If it is completely possible to have occurred then possibility is $\Pi(e) = 1$. If it is impossible then the possibility is $\Pi(e) = 0$. Intermediate numbers in $[0,1]$ represent an intermediate belief in event occurrence. A necessity measure is introduced to complement the information available about the state described by the attribute. The relationship between possibility and necessity satisfies:

$$N(e) = 1 - \Pi(\bar{e})$$
$$\forall e, \Pi(e) \geq N(e)$$

where $\bar{e}$ represents the complement of e. It is worth noting that if $\Pi(e)$ is a probability distribution, then $\Pi(e) = N(e)$.

2.2.4 Discussion

The literature carries heated debates about the role of fuzzy sets framework and probabilistic methods. A probabilistic-based approach assumes that one has an incomplete knowledge on the portion of the real world being modeled. However, this knowledge can be encoded as probabilities about events. The fuzzy approach, on the other hand, aims at modeling the intrinsic imprecision of features of the modeled reality. Therefore, the amount of knowledge at the user's disposal is of little concern. In addition to philosophical reasoning, the debate also relates to pragmatics. Probabilistic reasoning typically relies on event independence assumptions, making correlated events harder to assess. Results presented in [9] show a comparative study of the capabilities of probability and fuzzy methods. This study shows that probabilistic analysis is intrinsically more expressive than fuzzy sets. However, fuzzy methods demonstrate higher computational efficiency.

3 Taxonomy of Event Uncertainty

This section provides a taxonomy of event uncertainty. Section 3.1 defines two dimensions to classify uncertainties as relating to events and Section 3.2 describes the causes of event uncertainties for these dimensions.

3.1 Dimensions of Event Uncertainty

We classify the uncertainty according to two orthogonal dimensions: *Element Uncertainty* and *Origin Uncertainty*. The first dimension, *Element Uncertainty*, refers to the fact that event-related uncertainty may involve one of two elements:

Uncertainty regarding event occurrence: Such uncertainty is associated with the fact that although the actual event occurrence is atomic, the system does not know whether or not this event has in fact occurred. One example of such an event is the thermometer reading event from Example 3. Another example is money laundering, where at any point in time, money laundering may have been carried out by some customer. However, a Complex Event Processing (CEP) system can probably never be certain whether money laundering actually took place.

Uncertainty regarding event attributes: Even in cases in which the event is known to have occurred, there may be uncertainty associated with its attributes. For example, while it may be known that an event has occurred, its time of occurrence may not be precisely known. As another example, an event may be associated with a fuzzy domain, stating that a temperature is mild, in which case there is uncertainty regarding the exact temperature.

The second dimension, *Origin Uncertainty*, pertains to the fact that in a CEP system, there may be two types of events, raw events, signalled by event sources, and derived events, which are inferred based on other events. In the following, events which serve as the basis for the inference of other events will be termed *evidence events*, be they raw or derived events. Therefore, there are two possible origins for uncertainty:

Uncertainty originating at the event source: For raw events, there may be uncertainty associated either with the event occurrence itself, or the event's attributes, due to a feature of the event source. Example 3, in which uncertainty regarding an event occurrence is caused by the limited measuring accuracy of a thermometer, illustrates such a case.

Uncertainty resulting from event inference: Derived events are based on other events and uncertainty can propagate to the derived events. This is demonstrated by Example 2, in which uncertainty regarding measures of frustration of a customer propagates to the uncertainty of a customer deserting event.

Two additional examples are given next. In Example 5, the uncertainty of an event originates from the source, but is limited to its attributes, rather than to its occurrence. Example 6 shows uncertainty regarding an event's attributes resulting from event inference.

Example 5. Consider a case in which an event is generated whenever the temperature reading changes. Assume that the thermometer under discussion has the same accuracy as the one defined in Example 3. Furthermore, assume that the new temperature is an attribute of this event. In this case, there is no uncertainty regarding the actual occurrence. There is only uncertainty regarding this new temperature attribute.

Table 1 Uncertainty classification

	Origin: Event Source	**Origin: Event Inference**
Uncertainty Regarding Event Occurrence	Unreliable Source Imprecise Source Problematic Communication Medium Estimates	Propagation of Uncertainty Uncertain Rules
Uncertainty Regarding Event Attributes	Unreliable Source Imprecise Source Problematic Communication Medium Estimates Time Synchronization in Distributed Systems	Propagation of Uncertainty

Example 6. Consider a case in which an event e_3 should be inferred whenever an event e_2 occurs after an event of type e_1. Assume that the inferred event e_3 has an attribute a_1^3 whose value is the sum of the value of the attribute a_1^1 of event e_1 and the value of the attribute a_1^2 of event e_2. Now assume that both e_1 and e_2 are known to have occurred with certainty, but there is uncertainty regarding the value of attribute a_1^1. In this case there is uncertainty only regarding the value of attribute a_1^3, and this uncertainty results from event inference.

The above examples demonstrate that the two dimensions are indeed orthogonal. Therefore, uncertainty associated with events could be mapped into one of four quadrants, as shown in Table 1. In addition, due to the orthogonality of these dimensions, we define four types of event uncertainty: *Uncertainty regarding event occurrence originating at an event source*, *uncertainty regarding event occurrence resulting from inference*, *uncertainty regarding event attributes originating at an event source*, and *uncertainty regarding event attributes resulting from event inference*.

3.2 Causes of Event Uncertainty

This section describes, at a high level, the various causes of event uncertainty, according to the dimensions defined in Section 3.1. Table 1 summarizes the causes of uncertainty.

3.2.1 Causes of Uncertainty Originating at the Source

Uncertainty regarding event *occurrence* originating at an event source is caused by one of the following:

An unreliable source: An event source may malfunction and indicate that an event has occurred even if it has not. Similarly, the event source may fail to signal the occurrence of an event which has, in fact, occurred.

An imprecise event source: An event source may operate correctly, but still fail to signal the occurrence of events due to limited precision (or may signal events that did not occur). This is illustrated by Example 3.

Problematic communication medium: Even if the event source has full precision, and operates correctly 100% of the time, the communication medium between the source and the active system may drop indications of an event's occurrence, or generate indications of events that did not occur.

Uncertainty due to estimates: In some cases, the event itself may be a result of a statistical estimate. For example, it may be beneficial to generate an event whenever a network Denial of Service (DoS) event occurs, where the occurrence of such a DoS event is generated based on some mathematical model. However, the reasoner that deduce the event occurrence may produce a false positive type of error and hence this event also has uncertainty associated with it.

Uncertainty regarding the *attributes* originating at the event source can also be caused by any of the above reasons. An unreliable or imprecise source may be unreliable or imprecise regarding just the attribute values. Similarly, the communication medium may garble just the values of attributes, rather than messing with event occurrence. Finally, estimates or fuzzy values may also result in uncertainty regarding event attributes.

In distributed systems, there exists an additional cause for uncertainty regarding the special attribute capturing the occurrence time of the event. This is due to the fact that in distributed systems, the clocks of various nodes are usually only guaranteed to be synchronized to within some interval of a global system clock [21]. Therefore, there is uncertainty regarding the occurrence time of events as measured according to this global system clock.

It is worth noting that in both of the above cases, uncertainty regarding a specific event may be caused by a combination of factors. For example, it is possible that both the event source itself and the communication medium simultaneously corrupt information sent regarding the same event.

3.2.2 Causes of Inferred Uncertainty

Uncertainty regarding event *occurrence* resulting from inference has the following two possible causes:

1. **Propagation of Uncertainty:** A derived event can be a result of a deterministic pattern. However, when there is uncertainty regarding the events that are used for the derivation, there is also uncertainty regarding the derived event.
2. **Uncertain Patterns:** The pattern itself may be defined in an uncertain manner, whenever an event cannot be inferred with absolute certainty based on other events. An example of this is money laundering, where events denoting suspicious transactions only serve to indicate the possible occurrence of a money laundering event. Usually, a money laundering event cannot be inferred with certainty based on such suspicious transactions.

Note that these two causes may be combined. That is, it may happen that not only is the inference of an event based on an uncertain pattern, but also uncertainty exists regarding the occurrence (or attribute values) of one of the events which serve as evidence for this inference.

Regarding uncertainty of derived event *attributes*, the possible causes depend on how these attributes are calculated from the attributes of the evidence events. The most intuitive way to calculate such derived attributes is using deterministic functions defined over the attributes of the evidence events. In such a case, the only cause of uncertainty in the derived attributes is the propagation of uncertainty from the attributes of the evidence events. This is because the uncertainty regarding the event attributes is defined to be the uncertainty regarding the attribute values given that the event occurred. Therefore, the pattern cannot induce uncertainty regarding the attribute values of the derived events. However, if the inference system makes it possible to define the attribute values of the derived events in a non-deterministic (*e.g.*, fuzzy) manner, uncertain patterns may also be a cause of derived attribute uncertainty.

4 Handling Uncertainty at the Source

In this section we present two models for representing uncertainty at the source.

4.1 Probability Theory-Based Representation

The model presented next provides a probability theory-based representation for event uncertainty, based on [35]. A similar model was presented later by Balazinska *et al.* in [6] for RFID data. Arguing in favor of probability theory includes following reasons:

- Probability theory has widespread acceptance.
- Probability theory is a well-understood and powerful tool.
- Many technical results that facilitate its use have been shown formally.
- Under certain assumptions, probability is the only "rational" way to represent uncertainty (see [17]).
- There are well-known and accepted methodologies for carrying out inferences based on probability theory, involving structures such as Bayesian networks.
- Probability theory can be used together with utility theory (see [26]) for automatic decision making. Such automatic decision making would facilitate the implementation of automatic actions by complex event processing systems.

To apply probability theory to the handling of event processing in the context of uncertainty the notion of an event has to be extended, to allow the specification of uncertainty associated with a specific event. Also, deriving events in the context of uncertainty needs to be defined.

We represent the information a composite event system holds about each event instance with a data structure we term *Event Instance Data* (*EID*). *EID* incorporates

all relevant data about an event, including its type, time of occurrence, *etc*. In event composition systems with no uncertainty, each event can be represented by a single tuple of values $Val = \langle val_1, \ldots, val_n \rangle$, one value for each attribute associated with the event (see Section 2.1 for the introduction of event attributes). In our case, to capture the uncertainty associated with an event instance, the *EID* of each event instance is a Random Variable (*RV*). The possible values of *EID* are taken from the domain $V = \{notOccurred\} \cup V'$, where V' is a **set** of tuples of the form $\langle val_1, \ldots, val_n \rangle$.

The semantics of a value of E (encoded as an *EID*), representing the information the system has about event e are as follows: The probability that the value of E belongs to a subset $S \subseteq V \setminus \{notOccurred\}$ is the probability that event e has occurred, and that the value of its attributes is some tuple of the form $\langle val_1, \ldots, val_n \rangle$, where $\{\langle val_1, \ldots, val_n \rangle\} \subseteq S$. Similarly, the probability associated with the value $\{notOccurred\}$ is the probability that the event did not occur.

Example 7. Consider an event that quotes a price of \$100 for a share of IBM stock at time 10:45. Say that the system considers the following possible: The event did not occur at all; the event occurred at time 10:34 and the price of the stock was \$105; and the event occurred at time 10:45 and the price was \$100. In addition, say that the system considers the probabilities of these possibilities to be 0.3, 0.3 and 0.4, respectively. Then, the event can be represented by an RV E whose possible values are $\{notOccurred\}, \{10{:}34, IBM, 105\}$, and $\{10{:}45, IBM, 100\}$. Also, $\Pr(E = \{notOccurred\})$ and $\Pr\{E = \{10:34, IBM, 105\}\}$ are both 0.3 and $\Pr\{E \in \{\{10:45, IBM, 100\}, \{10:34, IBM, 105\}\}\} = 0.7$.

Example 8. In case of the thermometer related event appearing in Example 3, assume the following: the conditional probability that the temperature is 37.3°C, given that the thermometer reads 37.5°C, is 0.1, the probability that the temperature is 37.4°C is 0.15, the probability that the temperature is 37.5°C is 0.5, the probability that the temperature is 37.6°C is 0.15, and the probability that the temperature is 37.7°C is 0.1. The probability that the event did not occur is 0.3 and the probability that the event did occur is 0.7. Moreover, assume that whenever the event does occur, the temperature is an attribute of this event. Assume that at time 5, the thermometer registers a reading of 37.5°C. This information would be represented by an EID E with the following set of values: $\{notOccurred\}$ - indicating that the event did not occur, and 5 value sets of the form $\{5, 37.X°C\}$ - indicating that the event occurred at time 5 with temperature $37.X$°C, and X stands for $1, 2, 3, 4$ or 5. Examples of probabilities defined over E are $\Pr(E = \{notOccurred\}) = 0.3$ and $\Pr(E = \{5, 37.5°C\}) = 0.5 \cdot 0.7 = 0.35$ (due to the removal of the conditioning).

The set of possible values of the EID RVs contains information regarding both the occurrence of the event and its attributes. Therefore, this representation is sufficient to represent the uncertainty regarding both occurrence and attributes.

An additional concept, relevant in the context of event composition systems, is that of *Event History* (*EH*). An event history $EH_{t_1}^{t_2}$ is the set of all events (of interest to the system), as well as their associated data, whose occurrence time falls between t_1 and t_2. For example, consider the following events: an event e_1, at time 10:30,

quoting the value of an IBM share as \$100; event e_2, at time 10:45, quoting the value of an IBM share as \$105; event e_3 at time 11:00, quoting the value of an IBM share as \$103. Using the notation described above, the events e_1, e_2, e_3 can be described by the tuples $\{10{:}30, IBM, 100\}, \{10{:}45, IBM, 105\}$, and $\{11{:}00, IBM, 103\}$. Examples of event histories defined on these events are the following: $EH_{10:30}^{10:45} = \{e_1, e_2\}$, $EH_{10:45}^{11:00} = \{e_2, e_3\}$, $EH_{10:30}^{10:35} = \{e_1\}$ and $EH_{10:30}^{11:00} = \{e_1, e_2, e_3\}$. It is worth noting that there does not exist an event history that consists of both e_1 and e_3, and that does not include e_2.

The actual event history is not necessarily equivalent to the information regarding the event history possessed by the system. For example, a thermometer reading of $37.5°C$ is not necessarily the "true" one, as illustrated in example 8. We will therefore make the distinction by denoting the event history possessed by the system by *system event history*.

4.2 Fuzzy Set Theory-Based Representation

In [22], a model for representing basic fuzzy events, both in subscriptions and in publications, is given. An event will be of the form "x is $\tilde{A}$" where x is an attribute of an event and $\tilde{A}$ is a fuzzy value, associated with a membership function. For example, following Example 8, an event of the form "*temperature* of sensor s is *normal*" can be defined with the following membership definition of the fuzzy term *normal*:

$$\mu_{normal}(x) = \begin{cases} 0 & \text{if } x < 37.3°C \\ 0.1 & \text{if } x = 37.3°C \\ 0.15 & \text{if } x = 37.4°C \\ 0.5 & \text{if } x = 37.5°C \\ 0.15 & \text{if } x = 37.6°C \\ 0.1 & \text{if } x = 37.7°C \\ 0 & \text{if } x > 37.7°C \end{cases}$$

It is worth highlighting the differences between the two approaches presented in this section. The example above provides an interpretation of a fuzzy term using a distribution of values. In Example 8, we provided an interpretation of possible values, given an exact reading. Given the different settings, there is a room for both representations, and investigating the best method for combining them is a topic for future research.

5 Handling Inference Uncertainty

Inference uncertainty is handled by extending the notion of an event pattern, used in deterministic event processing systems, to represent the uncertainty associated with event derivation. This is described in this section, based on [35]. Although the description uses probability theory as a basis for the extension, this framework can be adapted to the use of fuzzy set theory or possibility theory. In the next section

we discuss methods for precise and efficient probability computation, as well as methods for evaluating complex fuzzy events.

In this chapter, we follow the KBMC approach, as introduced in Section 2.2: Knowledge is represented as probabilistic patterns (see Section 5.1), while probability calculation is carried out by constructing a Bayesian network based inference model (see Section 6).

Contemporary deterministic event processing systems use pattern matching for situation detection. In such systems, a situation is said to have occurred if the stream of incoming events match some pattern. At each point in time t, the existence of relevant patterns can be checked in the event histories that are known at that time. A pattern p is given in the form $\langle sel_p^n, pred_p^n, eventType_p, mappingExpressions_p, prob_p \rangle$ where:

sel_p^n is a deterministic predicate returning a subset of an event history of size less than or equal to n (for some integer n). If the returned subset has strictly less then n elements, no further evaluation of the pattern is carried out. A possible selection expression is "the first two events of type *stockQuote*." Therefore, for the event history e_1, e_2, e_3, if only e_1 is of type *stockQuote* the pattern is not triggered. However, if both e_1 and e_3 are of type *stockQuote*, then the subset $\{e_1, e_3\}$ is selected, and evaluation of the pattern continues.

$pred_p^n$ is a predicate of arity n over event instances (note that this is the same n appearing in sel_p^n). This predicate is applied to the n event instances selected by sel_p^n. An example of $pred_p^n$ is "the events e_1, e_2, e_3 have occurred in the order e_1, e_2, e_3, all three events are events regarding the same stock, and event e_3 occurred no later than 5 minutes after event e_1."

$eventType_p$ is the type of event inferred by this pattern. It can be, for example, an event of type *speculativeCustomer*.

$mappingExpressions_p$ is a set of functions, mapping the attribute values of the events that triggered this pattern to the attribute values of the derived event.

$prob_p \in [0,1]$ is the probability of inferring the derived event given that the pattern has occurred. The exact semantics of this probability are defined in Section 5.1.

By definition, the predicates defined by sel_p^n and $pred_p^n$ are deterministic, as are the functions $mappingExpressions_p$. Therefore, the only uncertainty present in the pattern is represented by the quantity $prob_p$. Indeed, many deterministic composite event languages, *e.g.*, the Situation Manager pattern Language [2] can be viewed as defining a set of patterns P such that each pattern $p \in P$ is of the form $\langle sel_p^n, pred_p^n, eventType_p, mappingExpressions_p \rangle$.

We assume that an event type is either explicit or inferred by a single pattern. This is done for simplicity. If there is more than one source of information for an event type (*e.g.*, two patterns), the probabilities supplied by the separate sources (patterns) must be combined to create a well-defined probability space (see Section 5.1).

We conclude this section with the definition of a specific language, instantiating each part of the rule, including the allowable syntax and semantics of s_r, p_r, *etc.* In this language, we have the following:

- sel_p^n is of the form $\langle selExpression_1, \ldots, selExpression_n \rangle$, where $selExpression_i$ is a selection expression of the form $\varepsilon_i \in eventType_i$, with *eventType* being a

valid event type. Given an event history, $selExpression_i$ will select a single event, ε_i. The event ε_i selected by $selExpression_i$ is the first event in the event history of type $eventType_i$ that was not selected by a selection expression $selExpression_j$ such that $j < i$.

- $pred_p^n$ is a conjunctive predicate defined over the events $\varepsilon_1, \ldots, \varepsilon_n$ selected by sel_p^n, of the form $\wedge_{i=1}^{m} predicate_i$. $predicate_i$ is either a temporal predicate, or an equality relation between attributes. If $predicate_i$ is an equality predicate, it is of the form $\varepsilon_k.attribute_l = \varepsilon_j.attribute_m$ for $k \neq j$. This specifies that the value of $attribute_l$ of event ε_k must be the same as $attribute_m$ of event ε_j. A temporal predicate $predicate_i$ takes one of the following forms:
  - $a \leq \varepsilon_k.occT \leq b$, where a and b are temporal constants denoting time points in the range $[0, \infty]$. This predicate specifies that the event has occurred within the interval $[a, b]$.
  - $\varepsilon_j.occT < \varepsilon_k.occT$ for $k \neq j$. This predicate defines a partial order over subsets of events.
  - $\varepsilon_j.occT \leq \varepsilon_k.occT \leq \varepsilon_j.occT + c$ for $k \neq j$, where c is a temporal constant such that $c > 0$. This predicate specifies that an event has happened within a specified interval relative to another event.

- Regarding $mappingExpression_r$ the occurrence time of the derived event is always determined to be the point in time at which the inference was carried out. As for other attributes, two types of functions are allowed. The first is a function, mapping a specific attribute value of a specific event participating in $pattern_r$ to an attribute of the derived event. The second is the mapping of a constant value to a derived attribute.

5.1 Event Inferencing Using Probability Theory

A valid probability space needs to be defined to quantify the probabilities associated with each event derived using an uncertain pattern. Therefore, pattern reasoning facilities need to be able to compute at any point in time t the probability that an event e, with specific data, occurred at some time $t' \leq t$. In addition, the only evidence that can be taken into account is that which is known to the system at time t. Therefore, a (possibly different) probability space is defined for each t. Extending our discussion in Section 2.2.1, the probability space at time t is a triple $pred_t = (W_t, F_t, \mu_t)$ such that:

- W_t is a set of possible worlds, with each possible world corresponding to a specific event history that is considered possible at time t. An assumption that holds in all practical applications is that the number of events in each event history, as well as the overall number of events, is finite. This is because an actual system cannot consider an infinite number of events in a finite time period. Therefore, each possible world corresponds to an event history that is finite in size. In addition, we assume that the real world is one of the possible worlds.

- $F_t \subseteq 2^{|W_t|}$ is a σ-algebra over W_t.
- $\mu_t : F_t \rightarrow [0,1]$ is a probability measure over F_t.

A less intuitive, yet more computationally useful way to define the probability space is as follows. Let $E_1, E_2, \ldots$ be the set of EIDs representing the information about all events of interest. It is clear that each finite event history can be represented by a finite number of values $e_1, \ldots, e_n$, such that there exists a finite number of EIDs $E_1, \ldots, E_n$ where e_i is a possible value of E_i. Therefore, each possible world $w_t \in W_t$ can be represented by such a finite number of values. In addition, as the overall number of events is finite, there is a finite number of events $E_1, \ldots, E_m$ such that E_i could have occurred in some $w_t \in W_t$. Finally, if $|W_t|$ is finite, each E_i can only have a finite number of associated values (one for each world in W_t) in which it appears. Note that in such a case, each possible w_t can be represented by a finite number of values $Val_1, \ldots, Val_m$, where the value $Val_1, \ldots, Val_n$ for some $n \leq m$ is a set of values, each such set representing the values of one of the n events that occurred in w_t, and $Val_{n+1}, \ldots Val_m$ are all $\{notOccurred\}$. From this it follows that if the probability space $pred_t$ represents the knowledge of the composite event system at time t, this knowledge can be represented by a set of m EIDs - $E_1, \ldots, E_m$.

Therefore, in the case where $|W_t|$ is finite, it is possible to define the probability space $pred_t$ as (Ω_t, F_t, μ'_t) where:

- $\Omega_t = \{Val_1, \ldots, Val_m\}$ such that the tuple $Val_1, \ldots, Val_m$ is the set of values corresponding to an event history, where this event history is a possible world $w_t \in W_t$ as described above. Obviously, $|\Omega_t|$ is finite.
- $F_t = 2^{|\Omega_t|}$
- $\mu'_t(\{Val_1, \ldots, Val_m\}) = \mu_t(w_t)$ such that w_t is the world represented by $\{Val_1, \ldots, Val_m\}$

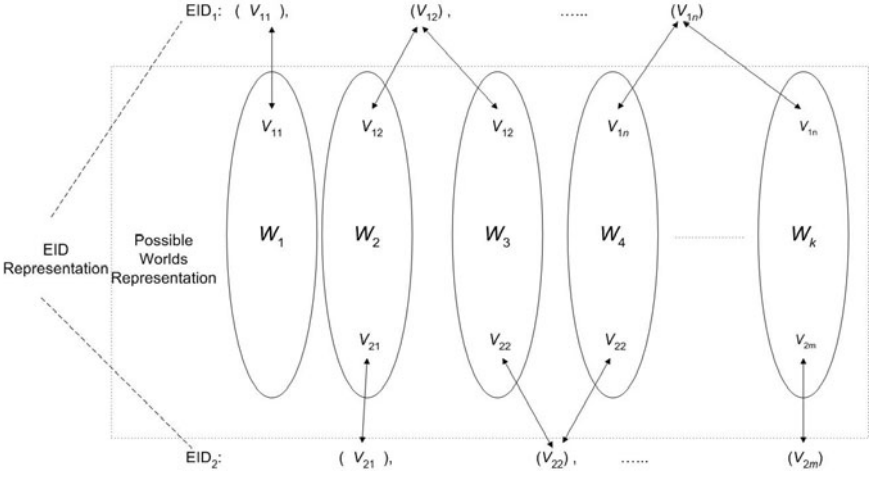

Fig. 1 Alternative probability space representation

This representation is termed the *EID representation*.

Figure 1 provides an illustration of two equivalent representations of the probability space. A possible world is marked as ovals in Figure 1. The figure presents two EIDs, one at the top and the other at the bottom. The participation of an event with a concrete value is marked by an arrow from the specific value to the possible worlds in which it participates. It is worth noting that the bottom event does not participate in world W_1, and therefore its value there is $\{notOccurred\}$.

As a concluding remark, note that each possible set of values of EIDs, $\{Val_1,\ldots,Val_m\}$ corresponds to some event history. Therefore, given an EID representation of $pred_t$, where $|\Omega_t|$ is finite, it is obviously possible to create the corresponding finite-size possible worlds representation by defining a possible world $w_t \in W_t$ for each distinct set of values $\{Val_1,\ldots,Val_m\}$.

We now define the semantics of patterns in the probability space discussed above. Intuitively, in such a probability space the semantics of each pattern p are as follows: Let $EH_{t_1}^{t_2}$ be an event history. If the pattern p is applied at some time $t \geq t_2$, and the set of events selected by sel_p^n from $EH_{t_1}^{t_2}$ is of size n and is such that $pred_p^n$ on this event is true, then the event inferred by pattern p occurred with probability $prob_p$. In addition, in such a case, the value of its corresponding attributes is the value defined by $mappingExpressions_p$. Otherwise, the event cannot be inferred.

Formally, let $sel_p^n(EH_{t_1}^{t_2})$ denote the set of events selected by sel_p^n from $EH_{t_1}^{t_2}$, and let $pred_p^n(sel_p^n(EH_{t_1}^{t_2}))$ denote the value of the predicate $pred_p^n$ on $sel_p^n(EH_{t_1}^{t_2})$ (recall that if $|sel_p^n(EH_{t_1}^{t_2})| < n$ then the pattern is not applied). In addition, let $val_1,\ldots,val_n$ denote the value of the attributes of the inferred event e_p as defined by $mappingExpressions_p$. Then, if the specific event history is known, and denoting by E_p the EID corresponding to e_p, we have the following:

$$\Pr(E_p = \{occurred, val_1,\ldots,val_n\} \mid EH_{t_1}^{t_2}) = prob_p \qquad if\ pred_p^n(SEL_p^n(EH_{t_1}^{t_2})) = true \tag{1}$$

$$\Pr(E_p = \{notOccurred\} \mid EH_{t_1}^{t_2}) = (1 - prob_p) \qquad if\ pred_p^n(SEL_p^n(EH_{t_1}^{t_2})) = true \tag{2}$$

$$\Pr(E_p = \{notOccurred\} \mid EH_{t_1}^{t_2}) = 1 \qquad if\ pred_p^n(SEL_p^n(EH_{t_1}^{t_2})) = false \tag{3}$$

Recall from Section 5.1 that if $|W_t|$ is finite, the probability space can be represented by a finite set of random variables, each with a finite set of values. In addition, note that the pattern semantics defined above specify that the probability of the derived event does not depend on the entire event history, but rather on the events selected by sel_p^n. Therefore, let us denote by $E_1,\ldots,E_m$ the set of EIDs that describe knowledge regarding the event history, and assume, without loss of generality that $\{E_1,\ldots E_l\}$ describe the subset of $\{E_1,\ldots,E_m\}$ that are candidates for selection by sel_p^n (note that $l \geq n$, as sel_p^n must choose the first n events that have **actually occurred**). An EID E is a candidate for selection if there is a possible event history in the

probability space $pred_t$ such that there is a set of n events which will be chosen by sel_p^n from this event history, and the event e corresponding to E is in this set. Then for all sets of values $\{Val_1,\ldots,Val_m\}$ such that $E_i = Val_i$, we have that

$$\Pr(E_p|E_1,\ldots E_l) = \Pr(E_p|E_1,\ldots,E_l,E_{l+1},...E_m) \tag{4}$$

i.e., E_p is conditionally independent of $\{E_{l+1},\ldots,E_m\}$ given $\{E_1,\ldots E_l\}$. Now let $Val_1,\ldots,Val_l$ denote a specific set of values of $E_1,\ldots E_l$. Given such a set of specific values, the subset $\{e'_{j_1},\ldots,e'_{j_n}\}$ selected by sel_p^n is well defined. Therefore, we have from the above equations that:

$$\Pr(E_p = \{occurred, val_1,\ldots,val_n\}|Val_1,\ldots,Val_l) = prob_p \quad if\ pred_p^n(e'_{j_1},\ldots,e'_{j_n}) = true \tag{5}$$

$$\Pr(E_p = \{notOccurred\}|Val_1,\ldots,Val_l) = (1 - prob_p) \quad if\ pred_p^n(e'_{j_1},\ldots,e'_{j_n}) = true \tag{6}$$

$$\Pr(E_p = \{notOccurred\}|\ e_1,\ldots,e_l)\ = 1 \quad if\ pred_p^n(e'_{j_1},\ldots,e'_{j_n}) = false \tag{7}$$

Eqs. 5-7 state that the inferred event and its values are conditionally probabilistically independent of all events prior to the inference, given exact information regarding the events that are candidates for selection by the corresponding selection expression.

As a concluding remark, we note that the mechanism of this section can also be used to predict future events, *i.e.*, at each point in time t, events occurring at any point in time t' could be inferred, based on a set of probabilistic patterns described at the beginning of the chapter. In order to enable such prediction, however, prediction patterns should be defined that, at time t, given a set of events, infer events whose occurrence time t' is such that $t' > t$.

5.2 Event Inferencing Using Fuzzy Set Theory

Recall that in Section 4.2, we have defined events of the form "x is $\tilde{A}$," associated with a membership function $\mu_A(x)$. Let R be a relation, describing how complex events are evaluated. The membership function of a complex event s over a set of m events $(x_1,x_2,...,x_m)$ is defined as follows:

$$M(x_1,x_2,...,x_m) = R(\mu_{A_1}(x_1),\mu_{A_2}(x_2),...,\mu_{A_m}(x_m))$$

R determines the method according to which membership values of different fuzzy sets can be combined. For example, consider the event "*temperature* of sensor s is *normal*" presented above, and assume another fuzzy event "*day t* is *hot*," with the following membership function:

$$\mu_{hot}(x) = \begin{cases} 0 & \text{if } x < 20°C \\ 0.1 & \text{if } 20°C \leq x < 25°C \\ 0.15 & \text{if } 25°C \leq x < 30°C \\ 0.75 & \text{if } 30°C \leq x \end{cases}$$

We can define a complex event "*temperature* of sensor *s* is *normal* and *day t* is *hot*" with the membership function

$$M(x_1, x_2) = \min(\mu_{normal}(x_1), \mu_{hot}(x_2))$$

The min operator is the most well-known representative of a large family of operators called *triangular norms* (t-norms, for short), routinely deployed as interpretations of fuzzy conjunctions (see, for example, the monographs [20, 15]).

A *triangular norm* $T : [0,1] \times [0,1] \to [0,1]$ is a binary operator on the unit interval satisfying the following axioms for all $x, y, z \in [0,1]$:

$$T(x,1) = x \text{ (boundary condition)}$$
$$x \leq y \text{ implies } T(x,z) \leq T(y,z) \text{ (monotonicity)}$$
$$T(x,y) = T(y,x) \text{ (commutativity)}$$
$$T(x,T(y,z)) = T(T(x,y),z) \text{ (associativity)}$$

The following t-norm examples are typically used as interpretations of fuzzy conjunctions:

$$Tm(x,y) = \min(x,y) \text{ (minimum t-norm)}$$
$$Tp(x,y) = x \cdot y \text{ (product t-norm)}$$
$$Tl(x,y) = \max(x+y-1,0) \text{ (Lukasiewicz t-norm)}$$

All t-norms over the unit interval can be represented as a combination of the triplet (Tm, Tp, Tl) (see [15] for a formal presentation of this statement). For example, the Dubois-Prade family of t-norms T^{dp}, also used often in fuzzy set theory and fuzzy logic, is defined using Tm, Tp and Tl as:

$$T^{dp}(x,y) = \begin{cases} \lambda \cdot Tp(\frac{x}{\lambda}, \frac{y}{\lambda}) & (x,y) \in [0,\lambda]^2 \\ Tm(x,y) & \text{otherwise} \end{cases}$$

The *average* operator belongs to another large family of operators termed *fuzzy aggregate operators* [20]. A fuzzy aggregate operator $H : [0,1]^n \to [0,1]$ satisfy the following axioms for every $x_1, \ldots, x_n \in [0,1]$:

$$H(x_1, x_1, \ldots, x_1) = x_1 \text{ (idempotency)} \quad (8)$$
for every $y_1, y_2, \ldots, y_n \in [0,1]$ such that $x_i \leq y_i$,
$$H(x_1, x_2, \ldots, x_n) \leq H(y_1, y_2, \ldots, y_n) \text{ (increasing monotonicity)} \quad (9)$$
$$H \text{ is a continuous function} \quad (10)$$

Let $\bar{x} = (x_1, \ldots, x_n)$ be a vector such that for all $1 \leq i \leq n$, $x_i \in [0,1]$ and let $\bar{\varpi} = (\varpi_1, ..., \varpi_n)$ be a weight vector that sums to unity. Examples of fuzzy aggregate operators include the *average* operator $Ha(\bar{x}) = \frac{1}{n}\sum_1^n x_i$ and the *weighted average* operator $Hwa(\bar{x}, \bar{\varpi}) = \bar{x} \cdot \bar{\varpi}$. Clearly, *average* is a special case of the *weighted average* operator, where $\varpi_1 = \cdots = \varpi_n = \frac{1}{n}$. It is worth noting that Tm (the min t-norm) is also a fuzzy aggregate operator, due to its idempotency (its associative property provides a way of defining it over any number of arguments). However, Tp and Tl are not fuzzy aggregate operators.

T-norms and fuzzy aggregate operators are comparable, using the following inequality:

$$\min(x_1, \ldots, x_n) \leq H(x_1, \ldots, x_n)$$

for all $x_1, \ldots, x_n \in [0,1]$ and function H satisfying axioms 8-10.

Tm is the only idempotent t-norm. That is, $Tm(x,x) = x$.[3] This becomes handy when comparing t-norms with fuzzy aggregate operators. It can be easily proven (see [15]) that

$$Tl(x,y) \leq Tp(x,y) \leq Tm(x,y) \tag{11}$$

for all $x, y \in [0,1]$.

Following this discussion, it becomes clear that the space of possible computation methods for the similarity of complex events is large. Additional research is required to identify the best fuzzy operator for a given complex event.

6 Algorithms for Uncertain Inferencing

This section is devoted to the efficient inferencing in a setting of uncertainty. We provide an example of an algorithm for uncertain inferencing of events, following [36]. In a nutshell, the proposed algorithm works as follows: Given a set of rules and a set of EIDs at time t, a Bayesian network is automatically constructed, correctly representing the probability space at t according to the semantics defined in Section 5.1. Probabilities of new events are then computed using standard Bayesian network methods.

Two important properties that must be maintained in any algorithm for event derivation (both in the deterministic and uncertain setting), are determinism and termination [40]. Determinism ensures that for the same set of explicit EIDs, the algorithm outputs the same set of derived EIDs. Termination ensures that the derivation algorithm terminates.

An algorithm for constructing a Bayesian network in this setting should take into account two main features. First, the Bayesian network is dynamically updated as information about events reaches the system. This is to ensure that the constructed network reflects, at each time point t, the probability space at t. Second, throughout the inference process additional information beyond the Bayesian network is stored. This additional information is used both to allow an efficient dynamic update of the network, and to make the inference process more efficient.

[3] For a binary operator f, idempotency is defined to be $f(x,x) = x$ (similar to [20], pp. 36).

The set of patterns is assumed to have no cycles, and a priority is assigned to each pattern so that determinism is guaranteed. By a cycle we mean a set of complex events that are both determined and participate in the decision making of each other. Priority determines a unique ordering of rule activation and can be set using rule quasi-topological ordering [1].

Recall that by definition, the occurrence time of each derived event is the time in which the pattern was applied. Therefore, the single possible occurrence time of each EID defines a full order on the EIDs (this single point in time for EID E is denoted by $E.occT$). In addition, according to Eq. 4, the uncertainty encoded with each EID is independent of all preceding EIDs, given the EIDs that may be selected by the selection expression. Therefore, a Bayesian network is constructed such that the nodes of the network consist of the set of random variables in the system event history, and an edge exists between EID E_1 and EID E_2 iff $E_1.occT \leq E_2.occT$ and E_2 is an EID corresponding to an event that may be inferred by pattern p, where the event corresponding to E_1 may be selected by sel_p^n. A network constructed by these principles encodes the probabilistic independence required by Eq. 4 (see [25]). This structure is now augmented with values based on Eq. 5-7. It is worth noting that this construction of the Bayesian network guarantees the probabilistic independence of EIDs. Any value dependency (*e.g.*, similar readings of close-by sensors) needs to be captured by pattern definition.

Based on the above principles, a Bayesian network is constructed and dynamically updated as events enter the system. At each point in time, nodes and edges may be added to the Bayesian network. The algorithm below describes this dynamic construction. The information regarding the new event is represented by some EID E, the system event history is represented by EH, and the constructed Bayesian network by BN. The algorithm follows:

1. $EH \leftarrow EH \cup \{E\}$
2. Add a node for E to BN.
3. For each such pattern p in a decreasing priority order:
   a. Denote by $sel_p^n(EH)$ the subset of EIDs in EH that may be selected by sel_p^n (these are all EIDs whose *type* attribute is of one of the types specified by sel_p^n).
   b. If there is a subset of events in $sel_p^n(EH)$ that may be selected by sel_p^n such that $pred_p^n$ is true, add a vertex for the derived event's EID E_p. In addition, add edges from all events in $sel_p^n(EH)$ to the event E_p.
   c. For E_p, fill in the quantities for the conditional probabilities according to Eq. 5-7.
4. Calculate the required occurrence probabilities in the probability space defined by the constructed Bayesian network.

The algorithm describes at a high level the calculation of the required probabilities, omitting the details of several steps. The omitted details include the mechanism for selection of events as indicated by sel_p^n, the evaluation of the predicates defined by $pred_p^n$, and the exact algorithm used to infer the required probabilities

from the Bayesian network. In all of these cases, standard algorithms from the domains of deterministic event composition and Bayesian networks may be used and extended. The specific algorithms used for these tasks will determine the complexity of our algorithm. However, the dominant factor will be the calculation of the required probabilities from the Bayesian network, which is known to be computationally expensive. Therefore, ways to speed up this step, including reduction in network size and approximate computation, are topics that warrants future research (see discussion in Section 7).

6.1 Inference Example

This section illustrates the above algorithm using a specific example. Assume that in the system there exists a pattern p_1 designed to recognize an illegal stock trading operation, and which is defined as follows: $sel^n_{p_1}$ is

$$\langle \varepsilon_1 \in stockSell, \varepsilon_2 \in stockPurchase \rangle$$

$pred^n_{p_1}$ is

$$(\varepsilon_1.occT \leq \varepsilon_2.occT \leq \varepsilon_1.occT + 5) \wedge (\varepsilon_1.stockTicker = \varepsilon_2.stockTicker) \wedge (\varepsilon_1.customerID = \varepsilon_2.customerID)$$

$eventType_{p_1}$ is $illegalStockTrading$, and $mappingExpression_{p_1}$ consists of two functions: The first maps $\varepsilon_1.stockTicker$ to the $stockTicker$ attribute of the inferred event, and the second maps $\varepsilon_1.customerID$ to the $customerID$ attribute of the inferred event. Finally, $prob_p$ is 0.7. The intuition underlying such a definition is that the sale of a stock, followed closely by a purchase of the same stock, is an indication of suspicious activity.

Consider now the following information about the possible occurrence of an event e_1 such that

$$e_1 \in stockSell, e_1.occT = 5, e_1.stockTicker = \text{“}IBM\text{”}, e_1.customerID = \text{“}C1\text{”}$$

In addition, the probability that this event occurred is 0.6. This information is represented in the system by an EID E_1 with two possible states: $\{notOccurred\}$ and $\{occurred, stockSell, 5, IBM, Customer1\}$ (we will abbreviate the second state by $\{occurred\}$. The constructed Bayesian network will consist of a single node E_1 with $\Pr(E_1 = \{notOccurred\}) = 0.4$ and $\Pr(E_1 = \{occurred\}) = 0.6$.

Another information is also received regarding the possible occurrence of an additional event

$$e_2 \in stockSell, e_2.occT = 9, e_2.stockTicker = \text{“}IBM\text{”}, e_2.customerID = \text{“}C1\text{”}$$

This is represented in the system by the EID E_2 with two states as above, which is added to the Bayesian network. At this stage, the Bayesian network consists of two disconnected nodes, E_1 and E_2.

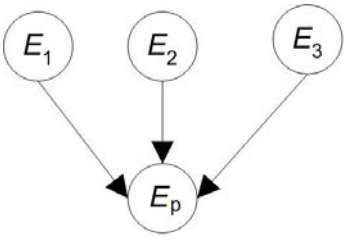

Fig. 2 Inference Example, Part I

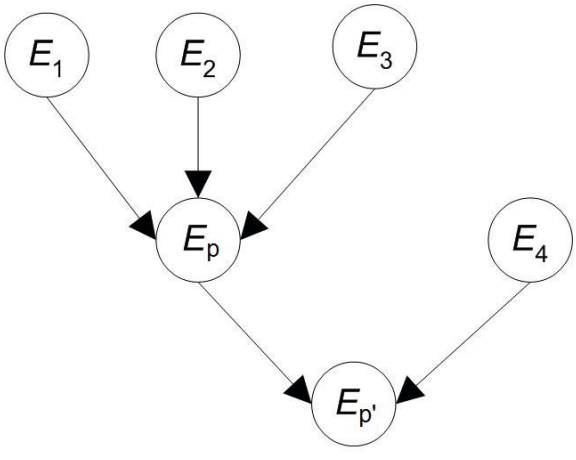

Fig. 3 Inference Example, Part II

Note that although two possible events have occurred, there is no possible world in which two events are selected by $sel^n_{p_1}$, and, therefore, the pattern p_1 is not recognized. Now, assume that information regarding a third event e_3 reaches the system, such that

$$e_3 \in stockPurchase, e_3.occT = 5, e_3.stockTicker = \text{“}IBM\text{”}, e_3.customerID = \text{“}C1\text{”}$$

This is represented in the system by the EID E_3. Now there is one possible world in which there is a non-zero probability that E_p occurs - this is the world in which the event history is e_2, e_3. Therefore, a node E_p is added to the network, and edges will be added from E_1, E_2, E_3 to E_p. This will result in the Bayesian network depicted in Figure 2.

In addition, the event corresponding to the EID E_p occurs only if e_1 did not occur, and e_2 and e_3 both occurred. Therefore, according to Eq. 5-7,

$$\Pr(E_p = \{occurred\} | E_1 = \{notOccurred\}, E_2 = \{occurred\}, E_3 = \{occurred\}) = 0.7$$

and

$$\Pr(E_p = \{occurred\} | E_1, E_2, E_3) = 0$$

for all other value combinations of E_1, E_2 and E_3.

Finally, if we define an additional pattern p' which states that an event has a non-zero occurrence probability whenever e_p and an additional event of type e_4 occurs, and e_4 is signaled, this will result in the network depicted in Figure 3.

7 Conclusions

In this chapter we have provided the basics of uncertain event processing. We have provided a basic model of uncertain events, demonstrated the use of probability theory, fuzzy set theory, and possibility theory in measuring uncertainty of events and the inferencing of such uncertainty in complex events. A specific simple event language is presented, highlighting the role of uncertainty management in event-based systems.

The challenges, associated with the use of uncertainty in event processing, may be classified into three categories, namely model, usability, and implementation issues, as detailed next.

In the modeling area, we have shown that probability-based models are suitable for some cases, but for other cases, there are more suitable models such as possibility theory or fuzzy set theory. The challenge is to construct a flexible generalized model that can match the appropriate model for a specific implementation.

In the usability area, a major difficulty is the practicality of obtaining the required rules and probabilities. As in many cases, it may be difficult even for domain experts to correctly specify the various cases as well as the probabilities associate with them. Machine learning techniques may apply for automatic generation of rules and probabilities, but the state-of-the-art in this area supports mining only simple patterns; furthermore, in some cases, the history is not a good predictor of the future. Initial work regarding automatic derivation of such rules appears in [34].

In the implementation area, event processing systems may be required to comply with scalability and performance requirements. Therefore, there is a need to develop algorithmic performance improvements to the current models such as Bayesian networks, which are known to be computationally intensive. See [37] for some initial steps in this direction. Possible additional directions for future work include performance improvements of existing derivation algorithms, either by general algorithmic improvements, or by developing domain and application specific efficient algorithms.

References

1. Adi, A.: A Language and an Execution Model for the Detection of Reactive Situations. PhD thesis, Technion – Israel Institute of Technology (2003)
2. Adi, A., Etzion, O.: Amit - the situation manager. The International Journal on Very Large Data Bases 13(2), 177–203 (2004)

3. The Apama home page, http://web.progress.com/en/apama/index.html
4. The AutoPilot home page, http://www.nastel.com/autopilot-m693.80.html
5. Bacchus, F.: Representing and Reasoning with Probabilistic Knowledge. MIT Press, Cambridge (1990)
6. Balazinska, M., Khoussainova, N., Suciu, D.: PEEX: Extracting probabilistic events from rfid data. In: Proceedings of the IEEE CS International Conference on Data Engineering, Cancun, Mexico (2008)
7. Breese, J.S., Goldman, R.P., Wellman, M.P.: Introduction to the special section on knowledge-based construction of probabilistic and decision models. IEEE Transactions on Systems, Man and Cybernatics 24(11), 1577 (1994)
8. Constantin, A., Gregory, C.: A structurally and temporally extended bayesian belief network model: Definitions, properties, and modelling techniques. In: Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI 1996), pp. 28–39. Morgan Kaufmann, San Francisco (1996)
9. Drakopoulos, J.: Probabilities, possibilities and fuzzy sets. International Journal of Fuzzy Sets and Systems 75(1), 1–15 (1995)
10. Dubois, D., Prade, H.: Possibility Theory: An Approach to Computerized Processing of Uncertainty. Plenum Press, New York (1988)
11. Etzion, O., Niblett, P.: Event Processing in Action. Manning publications (2010)
12. Gal, A., Anaby-Tavor, A., Trombetta, A., Montesi, D.: A framework for modeling and evaluating automatic semantic reconciliation. VLDB Journal 14(1), 50–67 (2005)
13. Green, T., Tannen, V.: Models for incomplete and probabilistic information. IEEE Data Eng. Bull. 29(1), 17–24 (2006)
14. Gustavo, A.-F., Luis, S.: A temporal bayesian network for diagnosis and prediction. In: Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI 1999), pp. 13–20. Morgan Kaufmann, San Francisco (1999)
15. Hajek, P.: The Metamathematics of Fuzzy Logic. Kluwer Acad. Publ., Dordrecht (1998)
16. Halpern, J.Y.: An analysis of first-order logics of probability. Artificial Intelligence 46(3), 311–350 (1990)
17. Halpern, J.Y.: Reasoning About Uncertainty. MIT Press, Cambridge (2003)
18. Kanazawa, K.: A logic and time nets for probabilistic inference. In: AAAI, pp. 360–365 (1991)
19. Kjaerul, U.: A computational scheme for reasoning in dynamic probabilistic networks. In: Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence, pp. 121–129 (1992)
20. Klir, G.J., Yuan, B. (eds.): Fuzzy Sets and Fuzzy Logic. Prentice-Hall, Englewood Cliffs (1995)
21. Liebig, C., Cilia, M., Buchman, A.: Event composition in time-dependant distributed systems. In: Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems, pp. 70–78. IEEE Computer Society Press, Los Alamitos (1999)
22. Liu, H., Jacobsen, H.-A.: Modeling uncertainties in publish/subscribe systems. In: Proceedings of the IEEE CS International Conference on Data Engineering, pp. 510–522 (2004)
23. The Netcool impact policy language home page, http://www-01.ibm.com/software/tivoli/products/netcool-impact/
24. The Oracle cep home page, http://www.oracle.com/technologies/soa/complex-eventprocessing.html

25. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)
26. Raiffa, H.: Decision Analysis: Introductory Lectures on Choices under Uncertainty. Addison-Wesley, Reading (1968)
27. The RTI data distribution service, http://www.rti.com/products/dds/
28. The RuleCore home page, http://www.rulecore.com/
29. Soulé, R., Hirzel, M., Grimm, R., Gedik, B., Andrade, H., Kumar, V., Wu, K.-L.: A universal calculus for stream processing languages. In: Proceedings of the 19th European Symposium on Programming, pp. 507–528 (2010)
30. The StreamBase home page, http://www.streambase.com/
31. The Sybase cep home page, http://www.sybase.com/products/financialservicessolutions/sybasecep
32. The Tibco cep home page, http://www.tibco.com/products/soa/messaging/rendezvous/default.jsp
33. The Tibco rendezvous home page, http://www.tibco.com/software/complex-eventprocessing/businessevents/default.jsp
34. Turchin, Y., Wasserkrug, S., Gal, A.: Rule parameter tuning using the prediction-correction paradigm. In: Proceedings of the third International Conference on Distributed Event-Based Systems (DEBS 2009), Nashville, TN, USA (July 2009)
35. Wasserkrug, S., Gal, A., Etzion, O.: A model for reasoning with uncertain rules in event composition systems. In: Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence (UAI 2005), Edinburgh, Scotland, pp. 599–608 (July 2005)
36. Wasserkrug, S., Gal, A., Etzion, O., Turchin, Y.: Complex event processing over uncertain data. In: Proceedings of the 2nd International Conference on Distributed Event-Based Systems (DEBS 2008), Rome, Italy (2008)
37. Wasserkrug, S., Gal, A., Etzion, O., Turchin, Y.: Efficient processing of uncertain events in rule-based systems. IEEE Transactions on Knowledge and Data Engineering, TKDE (2010) (accepted for publication)
38. The WebSphere business events home page, http://www-01.ibm.com/software/integration/wbe/
39. The WebSphere MQ home page, http://www-01.ibm.com/software/integration/wmq/
40. Widom, J., Ceri, S. (eds.): Active Database Systems: Triggers and Rules for Advanced Database Processing. Morgan Kaufmann, San Francisco (1996)
41. Zadeh, L.A.: Fuzzy sets. Information and Control 8, 338–353 (1965)