Event Processing Under Uncertainty

Alexander Artikis[1], Opher Etzion[2], Zohar Feldman[2], Fabiana Fournier[2]
[1]Institute of Informatics & Telecommunications,
National Centre for Scientific Research (NCSR) "Demokritos", Athens 15310, Greece
[2]IBM Research – Haifa, Haifa University Campus, Haifa 31905, Israel
a.artikis@iit.demokritos.gr, {opher, zoharf, fabiana}@il.ibm.com

ABSTRACT

Big data is recognized as one of the three technology trends at the leading edge a CEO cannot afford to overlook in 2012. Big data is characterized by volume, velocity, variety and veracity ("data in doubt"). As big data applications, many of the emerging event processing applications must process events that arrive from sources such as sensors and social media, which have inherent uncertainties associated with them. Consider, for example, the possibility of incomplete data streams and streams including inaccurate data. In this tutorial we classify the different types of uncertainty found in event processing applications and discuss the implications on event representation and reasoning. An area of research in which uncertainty has been studied is Artificial Intelligence. We discuss, therefore, the main Artificial Intelligence-based event processing systems that support probabilistic reasoning. The presented approaches are illustrated using an example concerning crime detection.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics—*Probabilistic algorithms*; I.2.3 [**Computing Methodologies**]: Artificial Intelligence—*Deduction and Theorem Proving, Uncertainty and probabilistic reasoning*

General Terms

Design

Keywords

Event Processing, Event Recognition, Pattern Matching, Uncertainty, Artificial Intelligence

1. INTRODUCTION AND MOTIVATION

Big data is recognized by Gartner as one of the three technology trends at the leading edge a CEO cannot afford to overlook in 2012 [34]. Big data is characterized by the four V's[1,2]: Volume or "data at rest", which is the amount of data to be processed; Velocity or "data in motion", which is the speed at which the data must be processed; Variety or "data in many forms", which represents the poly-structure in the data model; and Veracity or "data in doubt", which is the uncertainty in the data. There are a number of trends driving and enabling the exploitation of big data:

- The world is becoming instrumented through initiatives like the Internet of Things (IoT)[3] making data and events available pretty much everywhere (for example, 5 billion phones in use in 2010 [29]).
- The growing availability of cheap storage (for instance, $600 to buy a disk drive that can store the entire world's music [29]).
- The pervasiveness of sensor technology. For example, in 2010, 60% of the world's population was using mobile phones, and about 12% of those people had smartphones. The penetration of smartphones is growing at more than 20% a year [29].
- The spreading of broadband connectivity. For example, more than 30 million networked sensor nodes are now present in the transportation, automotive, industrial, utilities, and retail sectors. And the number of these sensors is increasing at a rate of more than 30% a year [29].
- The uptake of social networks by organizations as part of their business strategy [25].

It is not surprising that there is a pressing need for real-time recognition of events in the multitude of data that is being recorded and processed. This requirement may be addressed by employing recognition systems that detect situations or events of special significance within an organization, given streams of 'low-level' information that are very difficult to be utilized by humans.

The vast majority of today's event processing systems focus on the efficiency of reasoning algorithms. However, these don't take into account the various types of uncertainty that exist in most applications [11]. As big data applications, many of the emerging event processing systems are required

[1]`http://blogs.forrester.com/brian_hopkins/11-08-29-big_data_brewer_and_a_couple_of_webinars`
[2]`http://ibmresearchalmaden.blogspot.com/2011/09/ibm-research-almaden-centennial.html`
[3]For example, `http://www.theinternetofthings.eu/`

DEBS '12, July 16–20, 2012, Berlin, Germany.

to process events that arrive from sources such as sensors and social media, which have inherent uncertainties associated with them. In these cases, the streams of events may be incomplete or inaccurate, for example, regarding the time and location of events.

Many state-of-the-art systems assume that data is precise and certain, or that it has been cleansed before processing. As a result, processing this data is deterministic in nature. However, in many real-time event-driven applications, cleansing the data is not feasible due to time constraints. Consequently, analysis is performed on off-line data, while on-line data is not leveraged for immediate operational decisions.

In this paper we present the common types of uncertainty in event processing. Furthermore, we discuss ways of dealing with these types of uncertainty. One area of research in which uncertainty has been studied is Artificial Intelligence (AI). We discuss, therefore, the main AI-based event processing systems that support probabilistic reasoning.

The remainder of the paper is organized as follows. In Section 2 we present the relevant event processing terminology, while in Section 3 we present a crime detection scenario that we use throughout the paper to illustrate uncertainty in event processing. Section 4 outlines the common types of uncertainty that an event processing system has to address. Then, Section 5 presents how existing event processing approaches may be extended to deal with these types of uncertainty. Section 6 discusses AI-based systems. Finally, in Section 7 we discuss directions for further research.

2. BACKGROUND

2.1 Event Processing Constructs

The terminology we use in this paper is based on the event processing language presented in [15]. In this section we briefly introduce the language constructs fundamental to understanding the essence of concepts presented in this paper.

Event Type — a specification for a set of event objects that have the same semantic intent and the same structure; every event object is considered to be an instance of an event type. An event type can represent either *raw events* deriving from a producer or *derived events* produced by an *event processing agent*. An event can be either *simple* or *composite*. A composite event type is a special type of event, which is actually made up of a collection of other event types.

***Event Processing Agent* (EPA)** — a component that, given a set of input events, applies some logic for generating a set of output (derived) events.

Context — a named specification of conditions that groups event instances so they can be processed in a related way. While there exist several context dimensions, in this work we employ the two most commonly used dimensions: *temporal* and *segmentation-oriented*. A *temporal context* consists of one or more time intervals, possibly overlapping. Each time interval corresponds to a context partition, which contains events that occur during that interval. A *segmentation-oriented context* is used to group event instances into context partitions based on the value of an attribute or collection of attributes in the instances themselves. As a simple example, consider an EPA that takes a single stream of input events, in which each event contains a customer identifier attribute. The value of this attribute can be used to group events so there is a separate context partition for each customer. Each context partition contains only events related to that customer, so the behaviour of each customer can be tracked independently of the other customers.

***Event Processing Network* (EPN)** — a conceptual model, describing the event processing flow execution. An EPN comprises a collection of EPAs, event producers, and event consumers linked by channels. The event processing network was first introduced in the field of modelling by Luckman [28]. The conceptual model of EPN based on this idea was further elaborated by Sharon and Etzion [39]. A schematic presentation of EPN, including producers, consumers, EPAs, and channels between them is demonstrated in Figure 1.

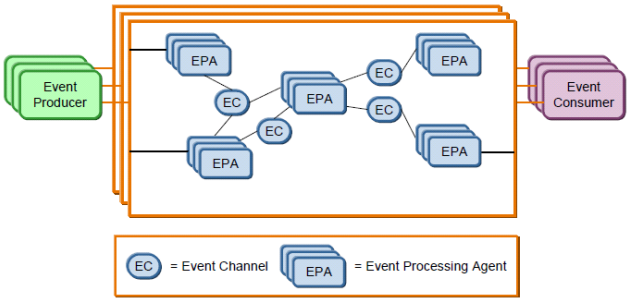

Figure 1: Schematic representation of EPN.

EPAs in a EPN communicate in asynchronous fashion by receiving and sending events. Although a channel can route several event types, in this paper we assume that a channel is manifested as an edge in the graph, connecting a single source with a single sink carrying a single event type.

2.2 Event Processing Agents

Although there are many different types of EPA, we present here a subset sufficient to illuminate the example in Section 3. A *Filter* agent is an EPA that excludes unwanted event instances, so it does not transform the input event. A *Split* EPA takes as an input a single event and creates a collection of events. Each of them can be a clone of the original event, or a projection of that event containing a subset of its attributes.

Pattern matching, that is, event recognition, is a type of task supported by EPAs that allows us to go beyond individual events and look for specific collections of events and the relationships between them [15]. Given any collection of events, it is possible to find one or more subsets of those events that match a particular pattern. We say that such a subset satisfies the pattern. Examples of pattern matching operations are: *All*, *Sequence*, *Threshold*, *Absence*, *Any*, *Count*, *Top-K*. The *Top-K* pattern, for instance, emits the events which have the K highest values of a specific attribute over all the participant events, where K is supplied as a

pattern parameter. *Pattern policies* are used to fine-tune and disambiguate pattern matching semantics. The *repeated* type policy determines what happens if the matching step encounters multiple events of the same type — possible values of this policy are *first*, *last*, *override*, *every*. *override*, for example, means that the participant event set keeps at most a specified number of instances of an event type. If a new event instance is encountered and the participant set already contains the required number of instances of that type, the new instance replaces the oldest previous instance of that type.

Pattern filter is a (usually stateless) condition that each individual event must satisfy to be considered for the pattern matching set. *Pattern assertion* is a (usually stateful) condition that the matching set is required to meet for the pattern to be satisfied. While pattern filter can be defined as part of pattern assertion, thus generating a logically equivalent definition, it is considered good practice to separate stateless and stateful conditions into pattern filter and pattern assertion, respectively.

Table 1: Event types.

Event Type	Attribute Name
Observation	Location Observed-id Picture Criminal-id Crime indication
Suspicious observation	Location Observed-id Picture Criminal-id Crime indication
Crime report	Location Crime type
Matched crime	(composite event) <Suspicious observation, Crime report>
Mega-suspect	Observed-id Picture Criminal record
Known criminal	Observed-id Picture Criminal record
Discovered criminal	Observed-id Picture
Suspicious location	Observed-id Location Num-observations
Top suspicious location list	Location list Array: dimension 1, Element type: location

3. ILLUSTRATIVE EXAMPLE

We illustrate our ideas through a simple scenario in the field of crime detection. A visual surveillance system must be able to detect and track people under a wide variety of environmental and imaging conditions. The system must then analyse their actions and interactions with one another, and with objects in their environment, to determine when real-time alerts should be posted to human security officers. Let us consider the case of a video surveillance system located at a town. The goal of the system is to detect and alert in real-time possible occurrences of crimes (for instance, drug deals). A crime is detected whenever the same person is observed participating in a potential crime scene more than five times in a certain day. The individual may or may not be known to the police. The system also monitors and reports the ongoings at suspicious locations, where many crimes are detected. Specifically, the system identifies locations in which more than ten suspicious detections were observed in three consecutive days and reports the three most suspicious locations in the last month. In our scenario, our system also produces alerts based on reports from citizens and suspicious observations detected by the application. For the sake of simplicity, we assume that video analysis is done elsewhere and the results are input as events to our system. Table 1 presents the events' meta-data or type definitions. Table 2 articulates the EPN of this scenario that is illustrated in Figure 2. Note that the context in rows 1, 3, and 6 of Table 2 is temporal, in rows 2 and 5 it is temporal and segmentation-oriented, while in row 4 it is only segmentation-oriented. The last column of this table indicates the references to the data stores used by the EPAs. The reference in this example is depicted as a dashed line in the EPN displayed in Figure 2. For the sake of simplicity, in Figure 2 we only label channels connecting EPAs to consumers.

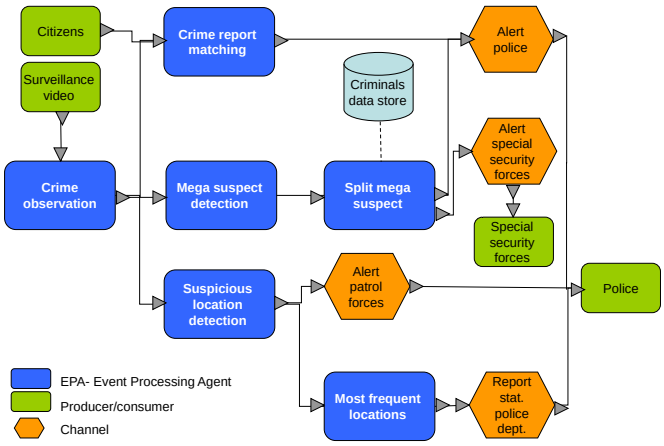

Figure 2: Crime event processing network.

Table 2: Event processing agent types.

EPA name	EPA type	Input event types	Output event types	Context	References
Crime observation	Filter Assertion: Crime indication=true	Observation	Suspicious observation	Always	
Mega suspect detection	Threshold assertion: Count (Suspicious observation>5)	Suspicious observation	Mega suspect	Daily per suspect	
Split mega suspect	Split	Mega Suspect	Known criminal when criminal record is not null — discovered criminal otherwise	Always	Criminals data store
Crime report matching	Sequence (Suspicious observation[override], Crime report)	Suspicious observation, Crime report	Matched crime	Location, Crime type	
Suspicious location detection	Threshold assertion: Count (Suspicious observation>10)	Suspicious observation	Suspicious location	Three consecutive days per location	
Most frequent locations	Top-K, Parameter K=3	Suspicious location	Top suspicious location list	Month	

4. UNCERTAINTY TYPES

Event processing systems often have to deal with various types of uncertainty, such as the following [4, 5]:

- Incomplete event streams. Consider, for example, the case in which the image processing system fails to detect a human activity at some point in time (say, due to occlusion).
- Insufficient event dictionary. The recognition of a composite event may require the detection of some other events that cannot be detected by the event producers or EPAs available at an application domain. In our example, the image processing system may not be able to detect all types of activity necessary for the recognition of a crime. This issue frequently arises in applications involving multimedia processing. For instance, in [4] the image processing system could not detect abrupt motion and, therefore, the overall event processing system could not always accurately recognize when two people were fighting.
- Erroneous event recognition. For example, the time of a civilian crime report may be merely approximated, and the interpretation of a situation may be mistaken.
- Inconsistent event annotation. Pattern recognition, that is, the process of constructing the pattern (definition) of a composite event may be performed manually, by interacting with the experts of an application domain (police officers, in our example), or by means of machine learning techniques using a training dataset. In most cases, the annotation of composite events in the training dataset is inconsistent.
- Imprecise event patterns. In many application domains, we only have an imprecise account of the pattern of a composite event. For instance, it may not be possible to precisely identify all conditions in which a crime of a particular type is said to take place.

Given the aforementioned uncertainty types, we distinguish between:

- uncertainty in the event input,
- certainty in the event input and uncertainty in the composite event pattern, and
- uncertainty in both input and pattern.

5. UNCERTAINTY IN EVENT PROCESSING

5.1 Uncertainty Representation

In traditional event processing, events are always certain, have deterministic values, and are typically viewed as occurring at time-points. However, in many applications this semantics can be limiting in several ways. First, events may occur during an interval, or more generally inside a union of intervals. Second, the time in which the event occurred or even the fact that it occurred at all might be uncertain. For example, this might happen if a crime is reported as having occurred at some point between 8AM and 10AM, but the exact time is uncertain, or there is a positive probability that it didn't happen at all (for example, it was wrongly interpreted). In what follows, we propose a way to extend the event processing terminology to express the above characteristics.

Table 3: Event header representation.

Attribute	Type
Event Name	String
Occurrence Time	Time-stamp/Interval
Detection Time	Time-stamp
Certainty	Double $(0, 1]$

The *event header* is a list of all the attributes that any event should carry. In order to enrich the event semantics to include intervals and uncertainty, we propose the attributes shown in Table 3. The 'certainty' attribute is used to denote how certain we are about the event actually occurring. An event with certainty equal to 1 means that we are positive the event occurred, which is the traditional semantics of events. 0 certainty means that there is no chance the event occurred, which is the same as not taking it into account at all. Therefore, there is no sense in having events with 0 certainty. For this reason, the range of this attribute value is $(0, 1]$.

To express the case where an event occurs during some period and not at a specific point in time, we use the notion of 'Interval'. An interval is composed of 'Start Time' and 'End Time', both of which are of type 'Time Stamp', with an implicit restriction that the end time is greater than the start time. In addition, for both start and end time, we must specify whether or not they are inclusive. For convenience, it is also possible to define an interval by the triplet (Start Time, End Time, Duration), where any subset of only two of them is sufficient to define the interval. The 'occurrence time' attribute may either be a time stamp, which is the traditional semantics of occurrence time, or it can be an interval.

We now have a way to represent the case when it is uncertain whether or not an event occurs or when it occurs during a time interval. Nevertheless, we are still not able to express the situation that the event occurs at some point in time during some period, but the exact time is not certain. For example, we may only know that an event occurs at some time-point in an interval, say between 8AM and 9AM, but the exact time remains unknown. This notion can be expressed by introducing probabilistic attribute values. We may say, for instance, that there is some probability of the event occurring at each point in this hour interval. Without any additional information, it is reasonable to assume that each point in the interval has the same likelihood, and therefore the uniform distribution may be the most natural choice to represent this case. In other cases, certain times may be more likely than others, and more general distributions such as triangular or (truncated) Gaussian may be plausible. Note that this has a different meaning from event duration. When we set the value of occurrence time to be $U(8AM, 9AM)$, where U denotes the uniform distribution, we mean that the event happens any time between 8AM and 9AM, with equal probability. If we want to say the event occurs during the entire interval, we should set the occurrence time to be 8AM, and the duration to be 1 hour.

In general, not only the occurrence time an event can be probabilistic, but also other attributes. In fact, the duration of an event may also be probabilistic. We indicate for each attribute whether it can be probabilistic or not using a simple flag. Table 4 presents the 'observation' event metadata from our running example, whereas Table 5 depicts a possible instance of this event.

Table 4: 'observation' event definition.

Attribute	Type	Probabilistic
Event Name	String	No
Occurrence Time	Time-stamp	Possibly
Detection Time	Time-stamp	No
Certainty	Double (0,1]	No
Location	Array[2]	Possibly
Observed-id	Integer	Possibly
Picture	String	No
Criminal-id	String	Possibly
Crime indication	Boolean	Possibly

Table 5: 'observation' event instance.

Attribute	Value
Event Name	Observation
Occurrence Time	$U\,(8AM, 9AM)$
Detection Time	9AM
Certainty	0.85
Location	$(349845.2, 7654345.4)$
Observed-id	5
Picture	NA
Criminal-id	"Jon Doe, 12345678"
Crime indication	P("true", 0.6, "false", 0.4)

It is possible that the values of different attributes are dependent. In this case, a joint distribution over a set of attributes may be used to express the dependency.

5.2 Uncertainty Handling

Traditional event processing needs to be enhanced in order to make use of event attributes that are represented by probabilities and distributions rather than standard native types. In the following subsections, we outline some of the main event processing functions and constructs that need to be re-factored to account for uncertainty. This is by no means intended to serve as a comprehensive study covering all aspects of uncertainty handling in event processing, nor does it provide a thorough doctrine that addresses all aspects. Nevertheless, we hint towards some possible uncertainty handling methods that can be roughly divided into two main approaches. The first approach is uncertainty propagation, according to which the uncertainty of the input events is propagated to the derived events in a coherent way from a mathematical (probabilistic) perspective. In contrast, the second approach is to eliminate the uncertainty, whenever it arises, before the derivation is carried out. (This process should not be confused with data cleansing.) Uncertain attributes are replaced by deterministic equivalents, and uncertain events may be screened out, according to some predefined policy. An exemplary policy

is to apply a threshold, such that any event with a certainty smaller than this threshold should be discarded, or otherwise treated as certain. When the uncertainty is removed, the events can be processed regularly. However, removing uncertainty may compromise event recognition accuracy as (crucial) information is lost. We demonstrate below how uncertainty propagation and uncertainty elimination can be applied in event processing systems.

5.2.1 Expressions

Expressions are used extensively in event processing in derivation and assertions. Expressions are composed of a combination of standard operators, such as arithmetic functions (for instance, sum, min), logical operators (for example, =, >), textual operators (such as concatenation), and operands or terms that reference particular event attributes or general constants.

Derivation. One of the main purposes of expressions is to calculate the value of derived event attributes. While expressions that are evaluated on deterministic terms result in a well-defined deterministic value, this is not the case for stochastic terms. Let us consider, for example, the expression

$$suspicious\_location1.num\_observations + suspicious\_location2.num\_observations$$

which sums the number of observations from two different locations. $e.a$ denotes the value of attribute a of event e. Since at least one of the terms in the above expression is stochastic, the expression has a stochastic value. Taking the uncertainty propagation approach, we may want to overload or extend the standard operators to accommodate stochastic values, such that stochastic expressions would evaluate to a distribution over a certain domain. Overloading operators, however, is not a trivial task. In some special cases, such as the sum of two independent normal random variables, the result is simply given by a normal distribution with the proper parameters. When considering general distributions, and probabilistic dependencies between the attributes, the result may need to be calculated by some numerical procedure, such as convolution. Such numerical procedures may be fairly cumbersome and complicated. In these cases, *Monte-Carlo* methods may serve as a good approach for evaluating stochastic expressions. Put simply, Monte-Carlo methods work by generating various samples from the attributes' distributions, evaluating the derivation on each generated sample, and determining the overall result based on the those evaluations.

In the uncertainty elimination approach, expressions should evaluate to a deterministic value. There are many possible ways to achieve this. Some examples include the following:

- $X+Y \Rightarrow expectation(X+Y)$ — the uncertain sum of X and Y is replaced by its expectation.
- $X+Y \Rightarrow percentile(X+Y, 0.9)$ — the uncertain sum of X and Y is replaced by its 0.9-percentile, representing a confidence upper-bound.
- $X > 5 \Rightarrow$ 'true' if $\mathbb{P}\{X > 5\} > 0.8$, and 'false' otherwise — the condition is satisfied if and only if its probability is greater than 0.8.

Table 6: Probabilistic functions.

Function	Parameters	Description
$expectation\,(X)$	X-Stochastic term	The expectation of the stochastic term X
$variance\,(X)$	X-Stochastic term	The variance of the stochastic term X
$pdf\,(X, x)$	X-Stochastic term, x-value	The probability of X taking value x
$cdf\,(X, x)$	X-Stochastic term, x-numeric value	The probability that X takes a value not larger than x
$percentile\,(X, \alpha)$	X-Stochastic term, α-numeric value	The smallest number z such that $cdf(X,z) \geq \alpha$
$frequent\,(X)$	X-Stochastic term	A value x that maximizes $pdf(X,\cdot)$

The set of functions that may be used in expressions can now be enriched with additional probabilistic functions that are applied to stochastic terms and attributes. A partial list of these probabilistic functions is depicted in Table 6. We note that probabilistic functions can be applied to standard deterministic values, as each deterministic attribute can be viewed as a degenerate stochastic attribute (that is, taking a certain value with probability 1).

Assertion. Expressions are also used in assertions that appear in filtering and patterns. In our illustrative example, we filter 'observations' that are suspected of being a crime, using the assertion *observation.crime_indication=true*, where the Boolean attribute *crime_indication*, derived by computer vision techniques, is uncertain. The threshold assertion $suspicious\_location1.num\_observations > 5$ is also stochastic since *num_observations* is uncertain. If we choose to propagate the uncertainty, similar to what is suggested above, the result of these expressions would be a distribution over the possible values 'true' and 'false', or, in a more compact form, the probability that the assertion is true. uncertainty that comes from satisfying the threshold. If the probability of the assertion is zero then the certainty is zero, which yields the expected result.

5.2.2 Context Assignment

Every EPA processes events in a particular context. As already mentioned, the context can be temporal or segmentation-oriented.

The temporal context groups events falling in a time window that may be fixed or repeating, and its boundaries may be determined by the occurrence of events that satisfy some conditions. When the occurrence time of an event is uncertain, like in the case of 'crime report', its association with a temporal context is inconclusive. For instance, if the crime reports are grouped by the hour, the assertion that verifies

whether the reported occurrence time T_o of any event falls within a certain hour, say between 7AM and 8AM, results in the probability $\mathbb{P}\{7 \leq T_o \leq 8\}$. One possible way to handle this situation in the case of fixed interval or sliding fixed interval, is to encapsulate the probability that the event's occurrence time falls in the interval under consideration in the 'certainty' attribute p_e of the event. In particular, if the probability that the event's occurrence time falls in the interval evaluates to a strictly positive value, the event is included in the context and the 'certainty' attribute of the event is updated to $p_e = p_e \cdot \mathbb{P}\{T_o \in [T_s, T_e]\}$, where T_s and T_e are the window start time and end time, respectively. This approach is suited to the propagation approach. Regarding the elimination approach, one may apply a threshold to the probability that the event actually occurred in this time window.

The case in which the time window boundaries are determined by events is more complex. For example, consider the context of the 'suspicious location detection' EPA. The context is initiated by a suspicious observation in a location where there was no such observation before, and ends three days later. Since the 'suspicious observation' event is uncertain, the question of whether the context should be initiated or not is raised. If, instead of terminating the context after 3 days, the termination was defined to be after 30 occurrences of 'suspicious observations', it would also be unclear when the context needs to be terminated. Obviously, the elimination approach would simplify things significantly here, but, as always, at the expense of losing information.

The segmentation-oriented context assigns events to context partitions based on one or more of the event attributes. The partitions could be pre-determined, say according to geographical area (up-town, down-town, industrial area, etc). Partitions may also be determined dynamically for each newly encountered distinct value of the partition attributes. For example, the context of 'mega suspect detection' EPA will have a distinct partition for any suspect detected in the system. Special care is needed in case the values of the partition attributes are uncertain. In our example, the identity of the suspect can be uncertain, possibly represented as a distribution over several possible personae. One possible way to handle this situation, is to assign the event to all the partitions corresponding to the possible identities id with probability $\mathbb{P}\{Identity = id\} > 0$. The 'certainty' attribute of the event could be updated in each partition to reflect the certainty in the identity of the person. In particular, the 'certainty' attribute would now be $p_e \cdot \mathbb{P}\{Identity = id\}$, or in general it would be $p_e \cdot p_\pi$, where p_π is the probability of taking values that correspond to a specific partition π.

With respect to the elimination approach, there are several possible ways to deal with probabilistic association to a context partition. One way is to simply associate the event with the context that has the most probable value. Alternatively, one may refer to quantities such as the percentiles or expectation of the partition attributes, to determine context assignment.

5.2.3 Event Recognition

Given a collection of events that comprise the state of a stateful EPA, there could be one or more subsets of those events that match a certain pattern. Patterns are defined by operators that refer to deterministic values, and when those are not so, errors are bound to occur. In our illustrative example, we have examples of several types of pattern. 'Most frequent locations' is a *Top-K* pattern which outputs the three most frequent locations for massive drug dealing in the last month. The value by which locations are ordered is the count of 'suspicious location' detections, which is stochastic. Therefore, the regular ordering cannot be used. Instead, a stochastic ordering needs to be considered here. There are several stochastic orders that may be used. Some possible choices are:

- Expectation order — events are ordered by the expected value of the pattern attribute.
- Tail order — events are ordered by the probability that the pattern attribute is smaller than, or larger than some value.
- Percentile order — events are ordered by the smallest value of the pattern attribute in which the cumulative distribution function is greater or equal to some probability.

Note that the expectation order coincides with the uncertainty elimination approach, which replaces the stochastic value with a deterministic value.

The 'split mega suspect' EPA emits an event that can be one of two types, depending on the attribute 'criminal record'. If a suspect has an indication of a criminal record, then a 'known criminal' event is emitted; otherwise a 'discovered criminal' event is emitted. Given that the identity of the suspect and his match to a known criminal from the police database is probabilistic, the split expression is uncertain. It may make sense in this case to propagate the uncertainty by emitting two events, one for each possible outcome.

In this section we discussed possible ways of extending standard event processing techniques to deal with various types of uncertainty. An area of research in which uncertainty has been already studied is Artificial Intelligence (AI). In the following section we discuss the main AI-based systems that support probabilistic reasoning.

6. EVENT RECOGNITION UNDER UNCERTAINTY IN ARTIFICIAL INTELLIGENCE

Event processing and, in particular, event recognition — event pattern matching — has been attracting considerable attention in the AI field. Event recognition systems from this field are typically logic-based and, therefore, exhibit formal, declarative semantics. Formal semantics allow for proving various properties of the composite event patterns (definitions), whereas declarative semantics can be more easily applied to a variety of settings, not just those that satisfy some low-level operational criteria. AI-based event recognition systems have also proven to be very efficient and scalable. Consider, for example, the Chronicle Recognition System [13] that has been successfully applied to cardiac monitoring [10], in addition to intrusion detection and mobility management in computer networks [14], and distributed diagnosis of web services [26]. A comprehensive introduction to AI-based event recognition systems may be found in [3, 5]. In this section we focus on systems that handle various types of uncertainty.

As already mentioned, event recognition based on multimedia content, such as crime detection using a video surveillance system, includes various types of uncertainty. It is not surprising, therefore, that most AI-based systems handling uncertainty have been evaluated for this type of application. Shet et al. [40, 41], for example, presented a logic programming (Prolog) approach to recognize theft, entry violation, unattended packages, and so on, given video content. Within their event processing system, Shet and colleagues incorporated mechanisms for reasoning over rules and facts that have an uncertainty value attached. Uncertainty in rules defining composite events corresponds to a measure of rule reliability; it is often the case that we only have imprecise knowledge about a composite event pattern. On the other hand, uncertainty in facts represents the detection probabilities of the simple events. In the VidMAP system [40], a mid-level module that generates Prolog facts, automatically 'filters out' data that a low-level image processing module has misclassified (such as a tree mistaken for a human). Shet and colleagues noted for the 'filtering' carried out by this module that '...it does so by observing whether or not the object has been persistently tracked' [40, p. 2].

In [41], an algebraic data structure known as a bilattice [18] is used to detect human entities based on uncertain output of part-based detectors, such as head or leg detectors. Bilattices may serve as a foundation of reasoning with imprecise information and fuzzy set theory [2]. For example, this structure supports inference in the presence of contradictory information from different event producers. The automatic human detection system of [41] is treated as a passive rational agent capable of reasoning under uncertainty. Uncertainties assigned to the rules, as well as detection uncertainties reported by the event producers, are taken from a set structured as a bilattice. The more confident the information provided, the more probable the respective composite event (human detection) becomes.

The Event Calculus (EC) has also been used for event recognition under uncertainty [17]. EC, originally proposed in [23], is a logic programming language for representing and reasoning about events and their effects. A key feature of EC is its built-in representation of the law of 'inertia': a fluent F — a property that is allowed to have different values at different points in time — holds at a particular time-point if F has been *initiated* by an event at some earlier time-point and not *terminated* by another event in the meantime.

EC has been used in the context of the ProbLog[4] [22] probabilistic logic programming framework to support event recognition under uncertainty. ProbLog differs from Prolog in that it allows for probabilistic facts, which are facts of the form $p_i::f_i$. In the expression $p_i::f_i$, p_i is a real number in the range [0, 1] and f_i is a Prolog fact. ProbLog, therefore, may be used for event recognition in applications in which events are not detected with certainty. Probabilistic facts in a ProbLog program represent random variables. Furthermore, ProbLog makes an independence assumption on these variables. The probability that a query q holds in a ProbLog program — for example, a composite event takes place at a particular point in time — is equal to the probability that at least one of the proofs of q is sampled. ProbLog's approach consists of using Binary Decision Diagrams (BDDs) [9] to compactly represent the proofs of a query. A BDD is a binary decision tree with redundant nodes removed and isomorphic subtrees merged. The BDD nodes represent the probabilistic facts of the ProbLog program. With the help of BDDs, ProbLog inference can scale up to handle queries containing thousands of different proofs.

[4] `http://dtai.cs.kuleuven.be/problog/`

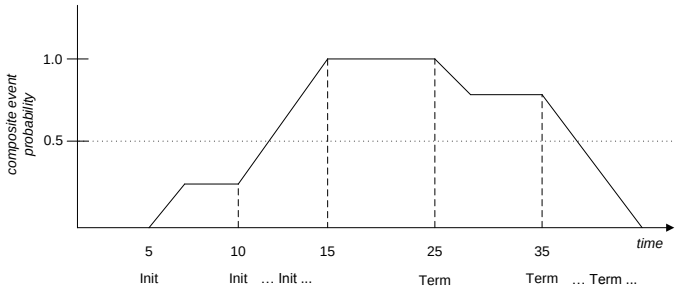

Figure 3: Composite event recognition in ProbLog-EC.

Figure 3 uses a simple example to illustrate the process of event recognition when EC operates on top of ProbLog — hereafter ProbLog-EC. Assume that we are interested in recognizing drug deals. At time-point 5 this composite event is said to be initiated, that is, there is some evidence that the composite event has indeed taken place — for instance, a suspect was observed with some probability to deliver items to 5 different people in the same day. Consequently, the probability of this composite event increases. The increase is proportionate to the confidence (probability) of the simple events that are detected by the underlying video processing system and initiate the recognition of drug dealing. Between time-points 5 and 10 no simple event is detected that affects the recognition of this composite event. In the absence of information, the law of inertia maintains the probability of the composite event. Between time-points 10 and 15, the composite event is repeatedly initiated — for example, the alleged drug dealer is making additional deliveries. The probability of drug dealing, therefore, continuously increases in these time-points. This is one of the characteristics of ProbLog-EC: the continuous presence of initiation conditions of a particular composite event causes an increase in the probability of this event. In other words, given continuous indication that an activity has possibly occurred, we are more inclined to agree that it has indeed taken place.

At time-point 25 the composite event is terminated, that is, there is evidence that the drug dealer has stopped operating. Consequently, ProbLog-EC decreases the probability of the composite event; the decrease is proportionate to the probabilities of the simple events that terminate the composite event. Due to the absence of information between time-points 25 and 35, the probability of the composite event remains the same. From time-point 35, the composite event is repeatedly terminated. Similar to the steady probability increase given continuous initiation conditions, when faced with subsequent termination conditions, the probability of the composite event steadily decreases. The slope of the descent (ascent, in the case of initiations) is defined by the probabilities of the events terminating (initiating) the composite event.

The dotted horizontal line at probability 0.5 in Figure 3 represents the recognition threshold that is commonly used

to discern between composite event instances that are considered to be trustworthy enough — these are the composite event recognitions — and those that are not.

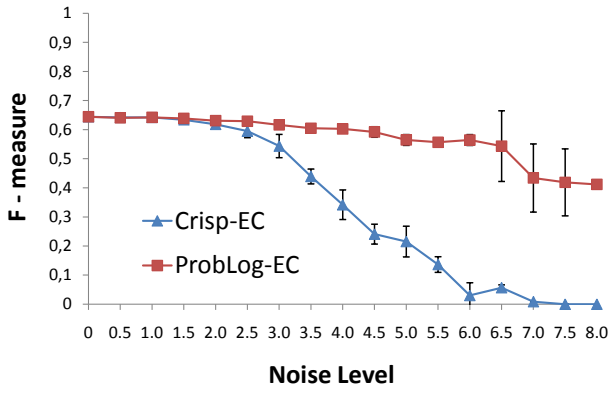

Figure 4: Composite event recognition accuracy: uncertainty elimination vs uncertainty propagation.

ProbLog-EC has been empirically evaluated against standard EC, that is, an EC dialect that does not handle uncertainty — hereafter Crisp-EC. Figure 4, for example, shows the recognition accuracy of ProbLog-EC and Crisp-EC for some composite event in terms of F-measure, that is, the harmonic mean of precision and recall, given increasing noise levels — the higher the noise level, the lower the probability attached to the input simple events. The vertical bars represent standard deviation. Uncertainty is eliminated in Crisp-EC: input events with a probability below 0.5 are discarded whereas events with a probability greater or equal to 0.5 are considered certain. On the hand, in ProbLog-EC all input events are taken into consideration. In [17] the authors identified a set of conditions in which uncertainty propagation (ProbLog-EC, in this case) outperforms uncertainty elimination (Crisp-EC) in terms of recognition accuracy, and vice versa.

Apart from logic programming techniques, probabilistic graphical models have been applied on a variety of event recognition applications where uncertainty exists (typically when simple events are detected on multimedia content). Event recognition requires processing streams of timestamped events and, therefore, numerous recognition methods are based on sequential variants of probabilistic graphical models, such as Hidden Markov Models [35], Dynamic Bayesian Networks [32] and Conditional Random Fields [24]. Graphical models can naturally handle uncertainty but their propositional structure provides limited representation capabilities. To model composite events that involve a large number of relations among other events, such as interactions between multiple persons and/or objects, the structure of the model may become prohibitively large and complex. To overcome such limitations, these models have been extended to support more complex relations. Examples of such extensions include representing interactions involving multiple domain objects [47, 45], capturing long-term dependencies between states [20], as well as describing a hierarchical composition of events [33, 27]. However, the lack of a formal representation language makes the representation of composite event patterns complicated and the integration of domain background knowledge very difficult.

Markov Logic Networks (MLNs)[5] [36] have also been used to deal with uncertainty in event recognition. MLNs combine first-order logic and probabilistic graphical models. The use of first-order logic allows for the representation of event patterns including complex (temporal) constraints. In MLNs each first-order formula may be associated with a weight, indicating the confidence we have in the formula. The main idea behind MLNs is that the probability of a world increases as the number of formulas it violates decreases. Therefore, a world violating formulas becomes less probable, but not impossible as in first-order logic. A set of Markov logic formulas represents a probability distribution over possible worlds.

Event recognition in MLNs involves querying a ground Markov network. Such a network is produced by grounding all rules expressing composite events. This is done using a finite set of constants that typically come from the input simple event streams. The MLN inference algorithms take into consideration not only the weights attached to the composite event patterns, but also the weights (if any) attached to the input simple events by the underlying event producers.

MLNs and ProbLog are closely related. A notable difference between them is that MLNs, as an extension of first-order logic, are not bound by the closed-world assumption. There exists a body of work that investigates the connection between the two frameworks. Bruynooghe et al. [8], for example, developed an extension of ProbLog that is able to handle first-order formulas with weighted constraints. Fierens et. al [16] converted probabilistic logic programs to ground MLNs and then used state-of-the-art MLN inference algorithms to perform inference on the transformed programs.

Event recognition using MLNs has been performed by Biswas et al. [6], for example. An approach that can represent persistent and concurrent composite events, as well as their starting and ending time-points, is proposed in [19]. The method in [37] employs hybrid-MLNs [46] to recognize successful and failed interactions between multiple humans using noisy location data. Similar to pure MLN-based methods, the knowledge base consists of composite event patterns. However, hybrid formulas aimed at removing the noise from the location data are also included. Hybrid formulas are defined as normal formulas, but their weights are also associated with a real-valued function, such as the distance between two people. As a result, the confidence of the formula is defined by both its weight and function. Although these methods incorporate first-order logic representation, the presented composite event patterns have a limited temporal representation — for instance, temporal constraints are defined over successive instants of time.

A method that uses interval-based temporal relations is proposed in [30]. The aim of the method is to determine the most consistent sequence of composite events based on the observations of event producers. Similar to [44, 21], the method uses MLNs to express composite events using common sense rules. In contrast to [44, 21], it employs temporal relations based on Allen's Interval Algebra (IA) [1]. To avoid the combinatorial explosion of possible intervals

[5]Software support for MLNs may be found at:
`http://alchemy.cs.washington.edu/`
`http://research.cs.wisc.edu/hazy/tuffy/`
`http://code.google.com/p/thebeast/`

that IA may produce, a bottom-up process eliminates the unlikely composite event hypotheses. In [7, 38] a probabilistic extension of Event Logic [42] is proposed to compute the intervals of the recognized composite events. Similar to MLNs, the method defines a probabilistic model from a set of weighted composite event patterns. However, the Event Logic representation avoids the enumeration of all possible interval relations.

MLNs have also been used to express EC [43]. This approach and ProbLog-EC tackle the problem of probabilistic inference from different viewpoints. ProbLog-EC handles noise in the input stream, represented as detection probabilities of the simple events. On the other hand, the MLN-based EC dialect — hereafter MLN-EC — emphasizes uncertainty in composite event patterns in the form of rule weights. In this way, MLN-EC supports various types of inertia. For example, in the absence of information, the probability of a composite event may increase or decrease, depending on the requirements of the application into consideration. The inertia behaviour of a composite event may be customized by appropriately setting the weight values of the corresponding MLN-EC rules. The ability to customize inertia improves recognition accuracy in various settings.

An important feature of MLNs is that they are supported by a variety of machine learning algorithms. It is possible, for example, to estimate the weights of the first-order rules expressing a composite event pattern, given a set of training data, that is, simple events annotated with composite events. Weight learning in MLNs is performed by optimizing a likelihood function, which is a statistical measure of how well the probabilistic model (MLN) fits the training data. Weights can be learned by either generative or discriminative estimation. Generative learning attempts to optimize the joint distribution of all variables in the model. In contrast, discriminative learning attempts to optimize the conditional distribution of a set of outputs, given a set of inputs. Discriminative learning is better suited to event recognition as in this case we know a-priori the input (simple or composite) events and output (composite) events.

In addition to weight learning, the structure of an MLN, that is, the rules expressing composite events, can be learned from training data. A variety of structure learning methods have been proposed for MLNs. These methods build upon the techniques of Inductive Logic Programming (ILP) [31]. In brief, the MLN structure learning methods can be classified into top-down and bottom-up methods [12]. Top-down structure learning starts from an empty or existing MLN and iteratively constructs clauses by adding or revising a single predicate at a time, using typical ILP operations and a search procedure. However, as the structure of an MLN may involve complex long-term activities, the space of potential top-down refinements may become intractable. For this reason, bottom-up structure learning can be used instead, starting from the training data and searching for more general hypotheses. This approach usually leads to a more specialized model, following a search through a manageable set of generalizations.

7. SUMMARY AND OPEN ISSUES

Event processing systems have to deal with various types of uncertainty. Consider, for example, applications processing events that arrive from sources such as sensors and social media that have inherent uncertainties associated with them. In these cases, the streams of events may be incomplete or inaccurate — both the time and location of events may be inaccurate.

We presented the common types of uncertainty that may be found in event processing. We discussed possible ways of extending traditional event processing systems to deal with these types of uncertainty. Moreover, we presented the main event processing systems from AI that already support probabilistic reasoning.

There are several directions for further research. AI systems handling uncertainty cannot always meet the requirements for real-time reasoning in real-world applications. It is necessary, therefore, to develop a framework for efficient and scalable event processing in the presence of various types of uncertainty.

Representing and reasoning about uncertainty is also quintessential in *forecasting* systems, that is, systems identifying events that are likely to occur in the near future (see, for example, [13]). To facilitate precisely-informed decision-making, forecasting should indicate not only which event will happen and with what probability, but also *when* it is expected to happen; more generally, forecasting should provide a probability distribution over the expected occurrence time.

A number of challenging issues remain open in learning composite event patterns. In principle, the structure of a composite event can be learned in two stages, using any ILP method, as discussed in the previous section, and then performing weight learning. However, separating the two learning tasks in this way may lead to suboptimal results, as the first optimisation step (ILP) needs to make assumptions about the weight values, which have not been optimised yet. Better results can be obtained by combining structure learning with weight learning in a single stage.

Acknowledgments

We have benefited from discussions with and the work of Anastasios Skarlatidis and Jason Filippou on AI-based event processing systems. Alexander Artikis is funded by the EU PRONTO project (FP7-ICT 231738).

8. REFERENCES

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[2] O. Arieli, C. Cornelis, G. Deschrijver, and E. E. Kerre. Bilattice-based squares and triangles. In *Proceedings of Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 563–575, 2005.

[3] A. Artikis, G. Paliouras, F. Portet, and A. Skarlatidis. Logic-based representation, reasoning and machine learning for event recognition. In *Proceedings of Conference on Distributed Event-Based Systems (DEBS)*, pages 282–293. ACM Press, 2010.

[4] A. Artikis, M. Sergot, and G. Paliouras. A logic programming approach to activity recognition. In *Proceedings of ACM Workshop on Events in Multimedia*, 2010.

[5] A. Artikis, A. Skarlatidis, F. Portet, and G. Paliouras. Logic-based event recognition. *Knowledge Engineering Review*, 2012.

[6] R. Biswas, S. Thrun, and K. Fujimura. Recognizing activities with multiple cues. In A. M. Elgammal,

B. Rosenhahn, and R. Klette, editors, *Workshop on Human Motion*, volume 4814 of *Lecture Notes in Computer Science*, pages 255–270. Springer, 2007.

[7] W. Brendel, A. Fern, and S. Todorovic. Probabilistic event logic for interval-based event recognition. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3329–3336. IEEE, 2011.

[8] M. Bruynooghe, T. Mantadelis, A. Kimmig, B. Gutmann, J. Vennekens, G. Janssens, and L. D. Raedt. Problog technology for inference in a probabilistic first order logic. In *ECAI 2010 - 19th European Conference on Artificial Intelligence*, 2010.

[9] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[10] L. Callens, G. Carrault, M.-O. Cordier, É. Fromont, F. Portet, and R. Quiniou. Intelligent adaptive monitoring for cardiac surveillance. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*, pages 653–657, 2008.

[11] G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 2011.

[12] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool Publishers, 2009.

[13] C. Dousson and P. L. Maigat. Chronicle recognition improvement using temporal focusing and hierarchisation. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 324–329, 2007.

[14] C. Dousson, K. Pentikousis, T. Sutinen, and J. Mäkelä. Chronicle recognition for mobility management triggers. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)*, pages 305–310, 2007.

[15] O. Etzion and P. Niblett. *Event Processing in Action*. Manning, Aug. 2010.

[16] D. Fierens, G. V. den Broeck, I. Thon, B. Gutmann, and L. D. Raedt. Inference in probabilistic logic programs using weighted cnf's. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI), July 2011*, 2011.

[17] J. Filippou, A. Artikis, A. Skarlatidis, and G. Paliouras. A probabilistic logic programming event calculus. Technical report, Cornell University Library, 2012. `http://arxiv.org/abs/1204.1851v1`.

[18] M. L. Ginsberg. Bilattices and modal operators. In *TARK*, pages 273–287, 1990.

[19] R. Helaoui, M. Niepert, and H. Stuckenschmidt. Recognizing interleaved and concurrent activities: A statistical-relational approach. In *PerCom*, pages 1–9. IEEE, 2011.

[20] S. Hongeng and R. Nevatia. Large-scale event detection using semi-hidden markov models. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1455–1462. IEEE, 2003.

[21] A. Kembhavi, T. Yeh, and L. S. Davis. Why did the person cross the road (there)? scene understanding using probabilistic logic models and common sense reasoning. In *ECCV (2)*, pages 693–706, 2010.

[22] A. Kimmig, B. Demoen, L. D. Raedt, V. S. Costa, and R. Rocha. On the implementation of the probabilistic logic programming language ProbLog. *Theory and Practice of Logic Programming*, 11:235–262, 2011.

[23] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–96, 1986.

[24] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In C. E. Brodley and A. P. Danyluk, editors, *ICML*, pages 282–289. Morgan Kaufmann, 2001.

[25] R. Lawrence, P. Melville, C. Perlich, V. Sindhwani, E. Meliksetian, P. Hsueh, and Y. Liu. Social media analytics. *ORMS today*, 37(1), February 2010.

[26] X. Le Guillou, M.-O. Cordier, S. Robin, and L. Rozé. Chronicles for on-line diagnosis of distributed systems. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*, pages 194–198, 2008.

[27] L. Liao, D. Fox, and H. Kautz. Hierarchical conditional random fields for gps-based activity recognition. *Robotics Research*, pages 487–506, 2007.

[28] D. Luckham. *The power of events: an introduction to complex event processing in distributed enterprise systems*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[29] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers. Big data: The next frontier for innovation, competition, and productivity, May 2011.

[30] V. I. Morariu and L. S. Davis. Multi-agent event recognition in structured scenarios. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.

[31] S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[32] K. Murphy. *Dynamic Bayesian Networks: representation, inference and learning*. PhD thesis, University of California, 2002.

[33] P. Natarajan and R. Nevatia. Hierarchical multi-channel hidden semi markov models. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 2562–2567, 2007.

[34] S. Prentice. Ceo advisory: Three technology trends at the leading edge you cannot afford to overlook in 2012. Technical report, Gartner, January.

[35] L. Rabiner and B. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[36] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[37] A. Sadilek and H. Kautz. Location-based reasoning about complex multi-agent behavior. *Journal of Artificial Intelligence Research*, 43:87–133, 2012.

[38] J. Selman, M. Amer, A. Fern, and S. Todorovic. Pel-cnf: Probabilistic event logic conjunctive normal form for video interpretation. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 680–687. IEEE, 2011.

[39] G. Sharon and O. Etzion. Event-processing network

model and implementation. *IBM System Journal*, 47(2):321–344, 2008.

[40] V. Shet, D. Harwood, and L. Davis. VidMAP: video monitoring of activity with Prolog. In *Proceedings of International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 224–229. IEEE, 2005.

[41] V. Shet, J. Neumann, V. Ramesh, and L. Davis. Bilattice-based logical reasoning for human detection. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2007.

[42] J. Siskind. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of Artificial Intelligence Research*, 15:31–90, 2001.

[43] A. Skarlatidis, G. Paliouras, G. A. Vouros, and A. Artikis. Probabilistic event calculus based on markov logic networks. In *RuleML America*, pages 155–170, 2011.

[44] S. D. Tran and L. S. Davis. Event modeling and recognition using markov logic networks. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, pages 610–623, 2008.

[45] D. Vail, M. Veloso, and J. Lafferty. Conditional random fields for activity recognition. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 235. ACM, 2007.

[46] J. Wang and P. Domingos. Hybrid markov logic networks. In *Proceedings of the 23rd national conference on Artificial intelligence*, volume 2, pages 1106–1111, 2008.

[47] T. Wu, C. Lian, and J. Hsu. Joint recognition of multiple concurrent activities using factorial conditional random fields. In *Proc. 22nd Conf. on Artificial Intelligence (AAAI-2007)*, 2007.