Probabilistic Complex Event Recognition: A Survey

ELIAS ALEVIZOS, National Center for Scientific Research (NCSR) "Demokritos"
ANASTASIOS SKARLATIDIS, National Center for Scientific Research (NCSR) "Demokritos"
ALEXANDER ARTIKIS, University of Pireaus and National Center for Scientific Research (NCSR) "Demokritos"
GEORGIOS PALIOURAS, National Center for Scientific Research (NCSR) "Demokritos"

Complex Event Recognition applications exhibit various types of uncertainty, ranging from incomplete and erroneous data streams to imperfect complex event patterns. We review Complex Event Recognition techniques that handle, to some extent, uncertainty. We examine techniques based on automata, probabilistic graphical models and first-order logic, which are the most common ones, and approaches based on Petri Nets and Grammars, which are less frequently used. A number of limitations are identified with respect to the employed languages, their probabilistic models and their performance, as compared to the purely deterministic cases. Based on those limitations, we highlight promising directions for future work.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Mathematics of computing** → **Bayesian networks**; **Markov networks**; **Probabilistic inference problems**; **Probabilistic algorithms**; **Stochastic processes**; • **Information systems** → **Data streams**; **Temporal data**; **Uncertainty**; • **Computing methodologies** → **Probabilistic reasoning**; • **Software and its engineering** → **Petri nets**; **Real-time systems software**; • **Theory of computation** → *Probabilistic computation*; *Grammars and context-free languages*; *Modal and temporal logics*; *Pattern matching*;

Additional Key Words and Phrases: Event Processing, Uncertainty, Probabilistic Logics, Probabilistic Automata, Probabilistic Graphical Models, Probabilistic Petri Nets, Stochastic Grammars

ACM Reference format:
Elias Alevizos, Anastasios Skarlatidis, Alexander Artikis, and Georgios Paliouras. 0000. Probabilistic Complex Event Recognition: A Survey. *ACM Comput. Surv.* 0, 0, Article 0 (0000), 31 pages.
https://doi.org/0000001.0000001

1 INTRODUCTION

Systems for complex event pattern matching, or Complex Event Recognition (CER), accept as input a stream of time-stamped, simple, derived events (SDE)s. A SDE (*low–level event*) is the result of applying a computational derivation process to some other event, such as an event coming from a sensor. Using SDEs as input, CER systems identify *complex events* (CE)s of interest– collections of events that satisfy some pattern [11]. The definition of a CE (*high–level event*) imposes temporal and, possibly, atemporal constraints on its sub-events (SDEs or other CEs). Consider, for example, the recognition of attacks on computer network nodes, given the TCP/IP messages. A CER system attempting to detect a Denial of Service attack has to identify (as one possible scenario) both a forged IP address that fails to respond and that the rate of requests is unusually high. In maritime

This work was funded partly by the EU H2020 datACRON project and partly by the EU FP7 SPEEDD project.

0360-0300/0000/0-ART0 $15.00
https://doi.org/0000001.0000001

monitoring, in order to detect an instance of illegal fishing, a CER system has to perform both some geospatial tasks, such as estimating whether a vessel is moving inside a protected area, and temporal ones, like determining whether a vessel spent a significant amount of time in this area. In this sense, CER is one of the functionalities of Complex Event Processing.

The SDEs arriving at a CER system almost always carry a certain degree of uncertainty and/or ambiguity. Information sources might be heterogeneous, with data of different schemas, they might fail to respond or even send corrupt data. Even if we assume perfectly accurate sensors, the domain under study might be difficult or impossible to model precisely, thereby leading to another type of uncertainty. Until recently, most CER systems did not make any effort to handle uncertainty (it is instructive to see the relevant discussion about uncertainty in [20]). This need is gradually being acknowledged and it seems that this might constitute a major line of research and development for CER.

The purpose of this paper is to present an overview of existing approaches for CER under uncertainty. Since this field is relatively new, without a substantial number of contributions coming from researchers directly involved with CER, we have chosen to adopt a broader perspective and include methods targeting activity recognition and scene understanding on image sequences coming from video sources. Although activity recognition is a field with its own requirements, it is related closely enough to CER so that some of the ideas and methods applied there might provide inspiration to CER researchers as well. However, it is not our intention to present a survey of video recognition methods and we have selectively chosen those among them that we believe are closer to CER, excluding those that rely on numerical approaches, such as [42] (for a survey of activity recognition methods from video sources, see [87]). We have used two basic criteria for our choice (applied to the CER methods as well). First, we require that the method employs some kind of relational formalism to describe activities, since purely propositional approaches are not sufficient for CER. Second, we require that uncertainty be handled within a probabilistic framework, since this is a framework that provides clear and formal semantics. In this respect, our work is related to previously conducted comparisons within the field of statistical relational learning, both theoretical [22, 38, 60] and practical [16]. The related field of query processing over uncertain data in probabilistic databases/streams is covered in other surveys (e.g., [88]) and, therefore, we will not include such papers in our survey.

Throughout the remainder of the paper, we are going to use a running example in order to assess the presented approaches against a common domain. Our example comes from the domain of video recognition. We assume that a CER engine receives as input a set of time-stamped events, derived from cameras recording a basketball game. However, we need to stress that our input events are not composed of raw images/frames and that the task of the CER engine is not to perform image processing. We assume availability of algorithms that can perform the corresponding tasks, such as object recognition and tracking. Therefore, our SDEs consist of events referring to objects and persons, like *walking*, *running* and *dribbling*. The purpose is to define patterns for the recognition of some high-level long-term activities, e.g., that a *matchup* between two players or a *double-teaming* is taking place.

The structure of the paper is as follows: Section 2 discusses briefly the types of uncertainty that may be encountered in a CER application. In Section 3 we present certain criteria based on which we included (or excluded) papers from our survey and in Section 4 we discuss the dimensions along which a proposed solution for handling uncertainty may be evaluated. Section 5 presents the reviewed approaches. Finally, Section 6 summarizes them in a tabular form and comments on their limitations. Some open issues and lines of potential future work are also identified.

2 UNCERTAINTY IN EVENT RECOGNITION

Understanding uncertainty in its different types is crucial for any CER system that aspires to provide an efficient way of handling it. The ideal CER system would be capable of handling all types of uncertainty within a unified and mathematically rigorous framework. However, this is not always possible and the current CER systems are still far from achieving such an ideal. Different domains might be susceptible to different types of uncertainty, while different CER engines employ various methods for responding to it, ranging from the ones that simply ignore it to those that use highly complex, fully-fledged, probabilistic networks. In this section, we give a brief description and classification of the various types of uncertainty that may be encountered by a CER system. For further discussion, see [8, 9, 90].

2.1 Data Uncertainty

The event streams that provide the input data to a CER engine can exhibit various types of uncertainty. One such type is that of incomplete or missing evidence. A sensor may fail to report certain events, for example due to some hardware malfunction. Even if the hardware infrastructure works as expected, certain characteristics of the monitored environment could prevent events from being recorded, e.g. an occluded object in video monitoring or a voice being drowned by stronger acoustic signals.

The events of the input stream may have a noise component added to them. In this case, events may be accompanied by a probability value. There are many factors which can contribute to the corruption of the input stream, such as the limited accuracy of sensors or distortion along a communication channel. Another distinction which might be important in certain contexts is that between stochastic and systematic noise, e.g. the video frames from a camera may exhibit a systematic noise component, due to different light conditions throughout the day.

When noise corrupts the input event stream, a CER system might find itself in a position where it receives events asserting contradictory statements. For example, in a computer vision application which needs to track objects, such as that of our running example, if there are multiple software classifiers, one of them may assert the presence of an object (e.g., the ball) whereas another may indicate that no such object has been detected.

Finally, when a CER system needs to learn the structure and the parameters of a probabilistic model from training data, quite often the data are inconsistently annotated. Therefore, the rules to be learned have to incorporate this uncertainty and carry a confidence value.

2.2 Pattern Uncertainty

Besides the uncertainty present in the input data, a noise-tolerant CER system should also be able to handle cases where the event patterns are not precise or complete. Due to lack of knowledge or due to the inherent complexity of a domain, it is sometimes impossible to capture exactly all the conditions that a pattern should satisfy. It might also be preferable and less costly to provide a more general definition of a pattern which is easy to implement rather than trying to exactly determine all of its conditions. A pattern with a wider scope, which does not have to check multiple conditions, may also be more efficient to compute and, in some cases, this performance gain could be more critical than accuracy. In both cases, we cannot infer an event with certainty and a mechanism is required to quantify our confidence.

For example, a rule for determining when a team is attempting an offensive move in basketball might be defined as a pattern in which one of the team's players has the ball and all other players are located in the opponent team's half-court. However, the same pattern could also be satisfied when a player is attempting a free throw or for an out-of-bounds play. Depending on our requirements, we

might or might not want to include all of these instances as cases of offensive moves. Defining all of these sub-cases would require more refined conditions, something which is not always possible. Yet, we might be still interested in capturing this pattern and provide a confidence value.

3 SCOPE OF THE SURVEY

Before presenting our framework and evaluation dimensions, we explain the rationale behind our choices and clarify some basic conceptual issues, which we deem important from the point of view of CER.

3.1 Probabilistic models

A seemingly simple method to handle uncertainty is to ignore or remove noise through pre-processing or filtering of the data, thus facilitating the use of a deterministic model. Other methods are available as well, such as possibilistic reasoning, conflict resolution (accept data according to the trustworthiness of a source) and fuzzy sets. For example, a logic-based method is proposed in [75, 76], where logic programming and the Bilattice framework [33] are employed. Another example is the work presented in [51], where the Dempster-Shafer theory is used in order to take into account the trustworthiness of heterogeneous event sources. We focus on probabilistic models because they provide a unified and rigorous framework and the bulk of research on CER under uncertainty employs such models.

3.2 Time representation

As far as time is concerned, some approaches resort to an implicit representation, whereby time slices depend on (some of) the previous slice(s), without taking into account time itself as a variable. Useful as this solution might be in domains characterized by sequential patterns, there are cases in CER where time constraints need to be explicitly stated. Although we include in our survey some approaches with an implicit time representation, our focus will mostly be on methods with explicit time representation.

3.3 Relational models

A substantial proportion of the existing probabilistic models are propositional by nature, as is the case with many Probabilistic Graphical Models, such as simple Bayesian Networks. Probabilistic graphical models have been successfully applied to a variety of CER tasks where a significant amount of uncertainty exists. Especially within the machine vision community, they seem to be one of the most frequently used approaches. Since CER requires the processing of streams of time-stamped SDEs, numerous CER methods are based on sequential variants of probabilistic graphical models, such as Hidden Markov Models [67] and their extensions (e.g., coupled [14], Dynamically Multi-Linked [34] and logical Hidden Markov Models [41]), Dynamic Bayesian Networks [62] and Conditional Random Fields [45].

As far as Hidden Markov Models are concerned, since they are generative models, they require an elaborate process of extracting the correct independence assumptions and they perform inference on the complete set of possible worlds. Moreover, their first-order nature imposes independence assumptions with regard to the temporal sequence of events (with only the current and the immediately previous states taken into account) that might not be realistic for all domains. On the other hand, Conditional Random Fields are discriminative models, a feature which allows them to avoid the explicit specification of all dependencies and, as a consequence, avoid imposing non-realistic independence assumptions [49, 86, 95]. However, both Hidden Markov Models and Conditional Random Fields assume a static domain of objects (with the exception of logical Hidden Markov Models [41]), whereas a CER engine cannot make the same assumption, since it is not

possible to determine beforehand all the possible objects that might appear in an input stream from a dynamic and evolving environment. Additionally, the lack of a formal representation language makes the definition of structured CEs (Complex Events) complicated and the use of background knowledge very hard. From a CER perspective, these issues constitute a severe limitation, since rules for detecting CEs often require relational and hierarchical structures, with complex temporal and atemporal relationships. For these reasons, we do not discuss Hidden Markov Models and Conditional Random Fields in a more detailed manner. Instead, we focus our investigation on methods with relational models.

4 EVALUATION DIMENSIONS

In this section, we provide a general framework for the discussion of the different approaches and establish a number of evaluation dimensions against which the strengths and weaknesses of each method may be assessed. We follow the customary division between representation and inference. As far as learning is concerned, although in general it is a very active research area within the statistical relational learning community, we have decided not to include a detailed discussion about the learning capabilities of the examined approaches in our survey. The reason is that very few of the probabilistic CER systems deal with learning (these exceptions are mentioned in Section 5). Instead, we will try to draw some conclusions about the performance of each system, taking into account the difficulties in making performance comparisons.

4.1 Representation

A simple unifying event algebra. We begin our discussion of representation by introducing a basic notation for CER. For a more detailed discussion of the theory behind CER, we refer readers to [19, 26, 50]. Following the terminology of [50], we define an event as an object in the form of a tuple of data components, signifying an activity and holding certain relationships to other events by time, causality and aggregation. An event with N attributes can be represented as $EventType(Attr_1, \ldots, Attr_N, Time)$, where $Time$ might be a point, in case of an instantaneous event, or an interval during which the event happens, if it is durative. Notice, however, that, when time-points are used, some unintended semantics might be introduced, as discussed in [63]. For our running example, events could be of the form $EventType(PlayerName, UnixTime)$ and one such an event could be the following: $Running(Antetokounmpo, 19873294673)$.

In CER, we are interested in detecting patterns of events among the streams of SDEs (Simple Derived Events). Therefore, we need a language for expressing such patterns. Based on the capabilities of existing CER systems and probabilistic CER methods, we adopt here a simple event algebra. Formalisms for reasoning about events and time have appeared in the past, such as the Event Calculus [10, 17, 44] and Allen's Interval Algebra [6, 7], and have already been used for defining event algebras (e.g., [64]). With the help of the theory of descriptive complexity, recent work has also identified those constructs of an event algebra which strike a balance between expressive power and complexity [97]. Our event algebra will be defined in a fashion similar to the above mentioned efforts, borrowing mostly from [17, 97]. Below, we present the syntax of the event

algebra:

$$
\begin{aligned}
ce ::= \; & sde && | \\
& ce_1 \; ; \; ce_2 && | \; \textit{Sequence} \\
& ce_1 \; \vee \; ce_2 && | \; \textit{Disjunction} \\
& ce^* && | \; \textit{Iteration} \\
& \neg \, ce && | \; \textit{Negation} \\
& \sigma_\theta(ce) && | \; \textit{Selection} \\
& \pi_m(ce) && | \; \textit{Projection} \\
& [ce]_{T_1}^{T_2} && \textit{Windowing} \text{ (from } T_1 \text{ to } T_2)
\end{aligned}
$$

where $\sigma_\theta(ce(v_1, \ldots, v_n))$ *selects* those ce whose variables v_i satisfy the set of predicates θ and $\pi_m(ce(a_1, \ldots, a_n))$ *returns* a ce whose attribute values are a possibly transformed subset of the attribute values of a_i of the initial ce, according to a set of mapping expressions m.

The following list explains the above operations:

- *Sequence*: Two events following each other in time.
- *Disjunction*: Either of two events occurring, regardless of temporal relations. Conjunction (both events occurring) may be expressed by combining *Sequence* and *Disjunction*.
- *Iteration*: An event occurring N times in sequence, where $N \geq 0$. This operation is similar to the *Kleene star* operation in regular expressions, the difference being that *Kleene star* is unbounded.
- *Negation*: Absence of event occurrence.
- *Selection*: Select those events whose attributes satisfy a set of predicates/relations, temporal or otherwise.
- *Projection*: Return an event whose attribute values are a possibly transformed subset of the attribute values of its sub-events.
- *Windowing*: Evaluate the conditions of an event pattern within a specified time window.

The above syntax allows for the construction of event hierarchies, a crucial capability for every CER system. Being able to define events at various levels and reuse those intermediate inferred events in order to infer other, higher-level events is not trivial. Theoretically, every event language could achieve this simply by embedding the patterns of lower-level events into those at higher levels, wherever they are needed. However, this solution would result in long and contrived patterns and would incur heavy performance costs, since intermediate events would need to be computed multiple times. Moreover, there are multiple ways a system could handle the propagation of probabilities from low-level to high-level events and these differences can affect both the performance and the accuracy of the system.

Probabilistic Data. The event algebra defined above is deterministic. We now extend it in order to take uncertainty into account. As we have already discussed, we can have uncertainty both in the data and the patterns. As far as data uncertainty is concerned, we might be uncertain about both the occurrence of an event and about the values of its attributes. For example, the ProbLog2 system [29] employs annotated disjunctions. Therefore, for a probabilistic event, we could write $Prob :: EventType(Value_1, \ldots, Value_N, Time)$, which means that this event with these values for its attributes might have occurred with probability $Prob$ and not have occurred at all with probability $1 - Prob$. In order to assign probabilities to attribute values, e.g. for two different values of $Attribute_1$,

we could write

$$Prob_1 :: EventType(Value_1^1, \ldots, Value_N, Time) \quad ;$$
$$Prob_2 :: EventType(Value_1^2, \ldots, Value_N, Time)$$

With respect to the probability space, a common assumption is that it is defined over the possible histories of the probabilistic SDEs. If SDEs are defined as discrete random variables, then one SDE history corresponds to making a choice about each of the SDEs among mutually exclusive alternative choices. The probability distribution is then defined over those SDE histories.

Probabilistic Model. In addition to handling uncertain data, we also require probabilistic rules. We express a probabilistic rule by appending its probability value as a prefix, e.g.

$$Prob :: ce(A,T) ::= \pi_{A=A_2,T=T_2}(ce_1(A_1,T_1); ce_2(A_2,T_2))$$

where, if ce_2 occurred after ce_1, then ce occurred at T_2 with probability $Prob$. The probability space is extended to include the inferred CE in the event histories. A probabilistic rule should then be understood as defining the conditional probability of the CE occurring, given that its sub-events occurred and satisfied its pattern. The attribute values of this CE are those returned by the *projection* operator π.

There are other ways to define the probability space and its semantics. For example, in the probabilistic programming literature it is common to use the possible worlds semantics for the probability space (e.g., in ProbLog [29]). The probability distribution is defined over the (possibly multi-valued) Herbrand interpretations of the theory, as encoded by the CE patterns. In this setting, we could assign non-zero probabilities even in cases where the rule is violated and we could end up with every Herbrand interpretation being a model/possible world. The existence of "hard" rules which must be satisfied excludes certain interpretations from being considered as models. When using grammars (and sometimes logic), the space might be defined over the possible proofs that lead to the recognition/acceptance of a CE.

4.2 Inference

In probabilistic CER, the task is often to compute the marginal probabilities of the CEs, given the evidence SDEs. Consider the following example:

$$P(offense(MilwaukeeBucks, [00:00, 00:24])|SDEs)$$

where we want to calculate the probability that the team of *MilwaukeeBucks* was on the offensive for the first 24 seconds of the game and we assume that $offense(team, [start, end])$ is a durative CE, defined over intervals and in terms of SDEs, such as *running*, *dribbling*, etc. Moreover, there are cases when we might be interested in performing maximum a posteriori (MAP) inference, in which the task is to compute the most probable states of some CEs, given the evidence SDE stream. A simple example from basketball is the query about the most probable time interval during which an offense by a team is taking place:

$$I_{offense} = \underset{I}{argmax} P(offense(MilwaukeeBucks, I)|SDEs)$$

Another dimension concerns the ability to perform approximate inference. For all but the simplest cases, exact inference stumbles upon serious performance issues, unless several simplifying assumptions are made. For this reason, approximate inference is considered essential. It is also possible for a system to provide answers with confidence intervals and/or the option of setting a confidence threshold above which an answer may be accepted.

4.3 Performance

CER systems are usually evaluated for their performance in terms of throughput and latency. Less often, the memory footprint is reported. When uncertainty is introduced, the complexity of the problem increases and other factors that affect performance enter the picture, such as the option of approximate inference. Unfortunately, standard benchmarks specifically targeting probabilistic CER have not yet been established. Therefore, in this survey we can report only what is available in the examined papers (see Section 6 for a further discussion).

Systems need to be evaluated along another dimension as well, that of accuracy. Precision and recall are the usual measures of accuracy. F-measure is the harmonic mean of precision and recall. The issue of accuracy is of critical importance and is not orthogonal to that of performance. A system may choose to sacrifice accuracy in favor of performance by adopting techniques for approximate inference. Another option would be to make certain simplifying assumptions with respect to the dependency relationships between events so that the probability space remains tractable.

5 APPROACHES

Surprisingly, there haven't been many research efforts devoted exclusively to the problem of handling uncertainty within the community of distributed event-based systems. The majority of research papers that could be deemed as relevant to our problem actually come from the computer vision community. Perhaps it is not much of a surprise if one takes into account the historical roots of CER systems. Stemming from the need to build more active databases and to operate upon streams of data that have a pre-defined schema, the problem of uncertainty, although present, was not as critical as in the case of efficiently processing events from sensors.

Our analysis has identified the following classes of methods: automata-based methods, Probabilistic Graphical Models, typically based on first-order logic, probabilistic/stochastic Petri Nets and approaches based on stochastic grammars (usually context-free).

5.1 Automata-based methods

Most research efforts targeting the problem of uncertainty in CER are based on extensions of crisp engines, the majority of which employ automata. In this section, we present these approaches. Compared to other methods, those based on automata seem better-suited to CER, since input events in CER are usually in the form of streams/sequences of events, similarly to strings of characters recognized by (Non-) Deterministic Finite Automata. CEs are usually expressed in a declarative way (similar to SQL) with the *sequence* operator playing a central role. These expressions are subsequently transformed into automata, using the stream of SDEs as input. In the probabilistic versions of automata-based methods, it is usually the SDEs that are uncertain, accompanied by probability values with respect to their occurrence and/or attributes, as opposed to the CE patterns. The goal is to use these probabilistic SDEs in order to determine the probabilities of CEs.

SASE-based approaches. SASE [94] and its extension, SASE+ [1], which includes support for the *Kleene plus* operation (similar to *Kleene star*, but with at least one event occurrence), is an automata-based CER engine which has frequently been amended in order to support uncertainty. The focus of SASE is on recognizing sequences of events through automata. For each CE pattern, an automaton is created whose states correspond to event types in the sequence part of the pattern, excluding possible negations. As the stream of SDEs is consumed, the automaton transitions to a next state when an event is detected that satisfies the sequence constraint. The recognized sub-sequences are pruned afterwards, according to the other non-sequence constraints (e.g., attribute equivalences), but, for some of these constraints, pruning can be performed early, while the automaton is active.

Those SDEs that triggered state transitions are recorded in a data structure that has the form of a directed acyclic graph, called *Active Instance Stack* (AIS), allowing for quick retrieval of those subsequences that satisfy the defined pattern. SASE+ [1] deviates somewhat from this scheme in that it employs NFA^b automata, i.e., non-deterministic finite automata with a buffer for storing matches. For the $skip-till-any-match$ selection strategy, where all possible SDE combinations that match the pattern are to be detected, the automaton is cloned when a SDE allows for a non-deterministic action. For example, a SDE whose type satisfies a *Kleene plus* operator, may be selected, in which case a new automaton is created.

For the probabilistic versions of SASE, the issue is how to correctly and efficiently calculate the probability of the produced CEs. In all of these versions [40, 74, 89, 96], this probability is calculated by conceptualizing a probabilistic stream as event histories, produced by making a choice among the alternatives of each SDE. For example, if there are 10 SDEs in a data stream and two alternatives for each of them – occurrence and non-occurrence – then there would be 1024 event histories. In [40, 89], SDEs are treated as having only these two alternatives. However, in other works (e.g., [74]), a SDE may have more alternatives, corresponding to different values for the arguments of the SDE. In [96], uncertainty about SDEs concerns their timestamps, which are described by a distribution, an issue not addressed in other works.

The probability of a CE could be calculated by enumerating all the histories, selecting those which satisfy the CE pattern and summing their probabilities. The probability of a history depends on the independence assumptions that each approach makes with respect to SDEs. Moreover, since a full enumeration is highly inefficient, optimization techniques are employed in order to calculate CE probabilities.

In the simplest case, all SDEs are assumed to be independent. In the work of [40], where this assumption is made, a matching tree is gradually constructed with SDEs that trigger state transitions. By traversing the tree, the sequence of SDEs producing a CE and its probability can be retrieved in a straightforward manner through multiplications, since all SDEs are independent. In this approach, as more SDEs arrive, probabilities can only become smaller and, by defining a confidence threshold, certain branches of the tree may be early pruned.

In [74], SDEs are again assumed to be independent, but a full enumeration is avoided by using a modified version of the Active Instance Stack, called the Active Instance Graph (AIG). The concept of lineage, borrowed from the field of probabilistic databases [12], is used for calculating CE probabilities. A similar approach is used in [89], where the assumption of complete SDE independence is relaxed and some SDEs may follow a first-order Markov process. In this case, the edges of the Active Instance Stack are annotated with the conditional probabilities. Note that the conditional probability tables in this approach are based on event types. For every specific SDE dependent on another SDE, probabilities must be explicitly provided. In this work, hierarchies of CEs are also allowed.

In [96], the issue of imprecise timestamps is addressed, while all the other attributes have crisp values. Again, SDEs are assumed to be independent. Complete enumeration of all possible worlds is avoided by employing an incremental, three-pass algorithm through the events in order to construct event matches and their intervals. This work was later extended [97], by adding *negation* and *Kleene plus* and by allowing for user-defined predicates.

All of these SASE-based methods perform marginal inference and use confidence thresholds for pruning results that fall below them. Only one attempts to increase performance through distribution. To the best of our knowledge, the work of [89] is one of the very few developing a CER system which is both probabilistic and distributed (PADUA, described below, is the only other such method).

Other automata-based approaches. Non SASE-based approaches have also appeared. A recognition method that models activities using a stochastic automaton language is presented in [5]. In this case, it is not the SDEs that are probabilistic, but the state transitions of the automaton, similarly to Markov chains. A possible world is now essentially defined over activity occurrences that are targeted for recognition, i.e. CEs. This method was later extended [4], in order to identify situations that cannot be satisfactorily explained by any of the known CEs. In particular, the stochastic automaton model is extended with temporal constraints, where subsequent SDEs can occur within a user-defined temporal interval. Using possible-worlds based modeling, the method finds (partially) unexplained activities. This is the only automata-based method that can perform both marginal and MAP inference.

Another extension of [5] is the PADUA system [57]. PADUA employs Probabilistic Penalty Graphs and extends the stochastic automaton presented in [5] with noise degradation. The edges that connect subsequent events in a Penalty Graph, forming the structure of a CE, are associated with a probability (noise) value that degrades the belief of the CE when other events intervene. As a result, under such situations, the CE is being recognized, but with reduced probability. Besides CER, the method can find patterns of events that do not belong to the set of known CEs. Additionally, for purposes of scalability, Probabilistic Penalty Graphs can be combined by merging common sub-Graphs, indexing and executing them in parallel.

Lahar [68] constitutes one of the earliest proposals and is based on the Cayuga engine [23]. Events are modeled by first-order Markov processes. The supported queries are categorized in three different classes of increasing complexity. For the first two types of queries (*regular* and *extended regular*), automata are used for recognition. For the most complex queries (*safe*), in which variables are not shared among all of the conditions, a version of the Probabilistic Relational Algebra [30] is used. A method which attempts to overcome the strict Markovian hypothesis of Lahar and apply certain optimizations, such as early pruning, through an algorithm called Instance Pruning and Filter-Detection Algorithm (IPF-DA), may be found in [18].

Commentary. Automata-based methods focus on recognizing sequences of events, in which some of those events may be related, via their attributes, to other events of the sequence. In general, time representation is implicit. As a result, and with the exception of [4], they do not include explicit temporal constraints, such as concurrency or inequalities between timestamps. *Windowing* is the only temporal constraint allowed. Moreover, they only address the issue of data uncertainty (the exception here is again [4]), lacking a treatment of other types of uncertainty, such as pattern uncertainty, and model relatively simple probabilistic dependencies between events.

To illustrate a case where concurrency and more complex dependencies may be required, consider a pattern trying to detect an attempted block by a defender:

$$\begin{aligned} attempt\_block(Y,T) ::= & \sigma_{opponents(X,Y)}(\\ & shooting(X,T) \wedge \\ & jumping(Y,T) \wedge \\ & close(X,Y,T) \ ) \end{aligned} \tag{1}$$

where player X is shooting at the same time that player Y is jumping, the distance between them is small at that time (we assume image recognition can provide such information as *close* SDEs) and the two players belong to different teams, i.e., they are *opponents*. Such a pattern would require explicit temporal constraints or, at least, an implicit constraint about concurrent events, a feature generally missing in automata-based methods. Moreover, *jumping* is clearly dependent on *shooting* (a player usually jumps at the same time or after another player shoots but not while all other players run), yet this dependence cannot be captured by assuming a Markov process that generated

those events. Note also that the above pattern makes use of the *opponents* predicate, assuming that the engine can take into account such background knowledge that is not part of the SDE stream. Such knowledge is relatively easy to model in logic-based systems, but the automata-based ones presented above have no such mechanism.

Now assume we want to express a rule stating that if two players are *close* to each other at the current timepoint, then they are likely to be close at the next timepoint (a first-order Markov assumption). This is not an event we would like to detect in itself, but domain knowledge which we would like our system to take into account. Such rules may be helpful in situations where SDEs may suddenly be missing, for example due to some sensor failure, but the activity has not ceased. We could introduce the following two patterns:

$$1 :: close\_m(X, Y, T) ::= close(X, Y, T) \tag{2}$$

$$0.6 :: close\_m(X, Y, T) ::= \sigma_{next(T, T_{previous})}(close\_m(X, Y, T_{previous})) \tag{3}$$

and use the *close_m* predicate instead of *close* in the definition of pattern (1) for *attempt_block*. The first of these patterns simply transfers the "detection" probability of *close* to that of *close_m* (pattern probability is 1), whereas the second one expresses the Markov assumption. In automata-based methods where Markov assumptions are allowed, the conditional probabilities need to be provided for every "ground" pair of SDEs. Uncertain patterns allow us to describe such dependencies in a more succinct manner, as "templates".

Assume also that we need some patterns to detect maneuvers in which the offender attempts to avoid the defender. Two of these patterns could be the following:

$$0.9 :: avoid(X, Y, T_2) ::= waiting(X, Y, T_1); crossover\_dribble(Y, T_2) \tag{4}$$

$$0.7 :: avoid(X, Y, T_2) ::= waiting(X, Y, T_1); running(Y, T_2) \tag{5}$$

where *crossover_dribble* and *waiting* are assumed to be CEs detected by their respective patterns.

In this case, we have a hierarchy of CEs, defined by probabilistic patterns, starting with the SDEs, on top of them the *waiting* and *crossover_dribble* CEs and finally the *avoid* CE. An efficient mechanism for propagating probabilities among the levels of the CE hierarchy would be required. Among the presented methods, CE hierarchies are allowed only in [89]. Moreover, combining rules would also be required, both for patterns (2) – (3) and patterns (4) – (5), i.e. functions for computing the probabilities of CEs with multiple patterns (common head, different bodies). For example, pattern (4) provides the probability of *avoid* given *waiting* and *crossover_dribble* and pattern (5) the probability of *avoid* given *waiting* and *running*, but we do not know this probability given all of the lower-level CEs. A combining rule could help us in computing such probabilities, without adding them explicitly.

5.2 First-order logic & Probabilistic Graphical Models

Another line of research revolves around methods which employ probabilistic graphical models in order to handle uncertainty. These models take the form of networks whose nodes represent random variables and edges encode probabilistic dependencies. The two main classes of probabilistic graphical models used in CER are Markov Networks and Bayesian Networks, the former being undirected models whereas the latter are directed. When used for CER, Markov Networks may be combined with first-order logic, in which case they are called Markov Logic Networks (MLNs). The nodes in a MLN represent ground logical predicates, as determined by the (weighted) formulas that

express CE patterns. When Bayesian Networks are used, the nodes usually represent events (SDEs and CEs).

Markov Logic Networks. Since their first appearance [69], Markov Logic Networks (MLNs) have attracted increasing attention as a tool that can perform CE recognition under uncertainty. MLNs are undirected probabilistic graphical models which encode a set of weighted first-order logic formulas (for a comprehensive description of MLNs, see [24]). CEs are expressed as logic formulas which may even contain existential and universal quantifiers (although the former can prove quite expensive), as with first-order logic. However, in first-order logic, a possible world (i.e., an assignment of truth values to all ground predicates) that violates even one formula is considered as having zero probability. In order to avoid this strict requirement, the methods described in this section may allow a formula to be "soft", meaning that it is accompanied by a probability/weight value, indicating how "strong" we need it to be compared to other formulas. As a consequence, possible worlds can have non-zero probability, even when violating some formulas, albeit a lower one than those without violations.

There is a substantial body of work on CER with MLNs, mostly concerned with human activity recognition, with input events derived from video sources (and less frequently from GPS or RFID traces). As a result, many of them have developed solutions that are domain–dependent. Here we focus on those representative papers that are more closely related to CER, by providing a more generic way for handling events, as in [59] and [82], where Allen's Interval Algebra [6] is used and in [79, 80], where a version of the Event Calculus [44] is used. With the use of such formalisms, temporal constraints are not captured in the simplistic way implied by the *greaterThan* predicate. Instead, built-in predicates about the temporal relations of events are provided, e.g., the *after* relation in Allen's Interval Algebra, for indicating that an event succeeds in time another event.

Contrary to automata-based solutions, MLNs focus on encoding probabilistic rules. This allows both for incorporating background knowledge and for building hierarchies of CEs with correct probability propagation. On the other hand, they use the less intuitive weights instead of probabilities, which indicate how strong a rule is compared to the others. While it might be possible for certain simple domains to manually define weights, usually a learning phase is required to optimize them.

As far as data uncertainty is concerned, it is possible to include probabilistic SDEs as well. Since a node in a MLN is not directly associated with a probability (it is the formulas/graph cliques that have weights), these SDEs must be expressed as formulas too. Such formulas connect each observed SDE with a "generated" SDE, with an appropriate weight (see the *Commentary* section below for more details). Moreover, MLNs allow for more complex reasoning about SDEs. For example, in the work of [85], besides handling noisy SDEs, missing SDEs may be inferred through rules about what must have happened for an event to have occurred.

A similar approach is proposed by [59], where the Interval Algebra is employed and the most consistent sequence of CEs are determined, based on the observations of low-level classifiers. In order to avoid the combinatorial explosion of possible intervals, as well as to eliminate the existential quantifiers in CE patterns, a bottom-up process eliminates the unlikely event hypotheses. The elimination process is guided by the observations and the interval relations of the CE patterns.

In [81, 82], MLNs are again combined with Allen's Interval Algebra, upon which a set of domain-independent axioms is proposed. *Abstraction axioms* define hierarchies of events in which an instance of an event with a given type is also an instance of all abstractions of this type. *Prediction axioms* express that the occurrence of an event implies the occurrence of its parts. *Constraint axioms* ensure the integrity of the (temporal) relations among CE and its parts. Finally, *abduction axioms*

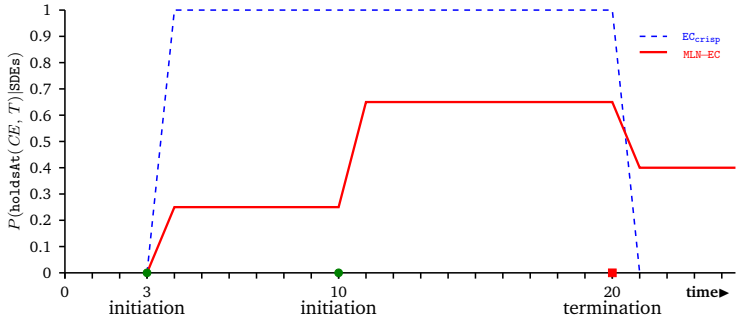

Fig. 1. CE probability estimation in the Event Calculus. The solid line concerns the Event Calculus in MLNs, MLN-EC. The dashed line concerns a crisp (non-probabilistic) version of the Event Calculus. Due to the law of inertia, the CE probability remains constant in the absence of SDEs. Each time the initiation conditions are satisfied (e.g., in timepoints 3 and 10), the CE probability increases. Conversely, when the termination conditions are satisfied (e.g., in timepoint 20), the CE probability decreases.

allow CE to be inferred on the basis of their parts, i.e. inferring when some events are missing. In this case, SDEs are not probabilistic.

The work of [79, 80] represents one of the first attempts to provide a general probabilistic framework for CER via MLNs. In order to establish such a framework, a version of the Event Calculus is used whose axioms are domain-independent. Combined with the probabilistic domain-dependent rules, inference can be performed regarding the time intervals during which activities of interest (fluents in the terminology of the Event Calculus) hold. The problem of combinatorial explosion due to the multiple timepoints that need to be taken into account is addressed by employing a discrete version of the Event Calculus, using only integer timepoints and axioms that relate only successive timepoints. For similar reasons, existential quantifiers are not allowed. Due to the law of inertia of the Event Calculus (something continues to hold unless explicitly terminated or initiated with a different value), this model increases the probability of an inferred event every time its corresponding rule is satisfied and decreases this probability whenever its terminating conditions are satisfied, as shown in Figure 1. In [78] the Event Calculus was again used, but this time the focus was on probabilistic SDEs rather than probabilistic rules. The ProbLog framework was used [43], a probabilistic extension of the logic programming language Prolog. ProbLog allows for assigning probabilities to SDEs, i.e. for probabilistic facts. Intuitively, the success probability of a query (CE rule) is the probability that this query is provable, starting from the (probabilistic) facts and any other rules that might be present as background knowledge. The axioms of the Event Calculus were expressed in ProbLog as background knowledge and a set of CE rules for human activity recognition from video sources was tested. The method exhibited improved accuracy performance with respect to crisp versions of the Event Calculus, when tested against noisy SDE streams.

The work of [15, 73] presents the Probabilistic Event Logic that defines a log-linear model from a set of weighted formulas. It does not directly employ MLNs but it is very close in spirit. The formulas that describe CEs are represented in Event Logic, a formalism for defining interval-based events [77], by employing the operators of Allen's Interval Algebra. Each formula defines a soft constraint over some events, in a manner similar to MLNs. However, instead of building a ground network with all the time variables, the inference algorithm works with a specialized data structure, called *spanning intervals*. This data structure allows for a compact representation of event occurrences that satisfy a formula. The inference algorithm can work with set operations on intervals and

performs local-search, based on MaxWalkSAT [39]. Moreover, a method is presented for learning the weights of the formulas.

A probabilistic activity language for expressing CEs on top of an image processing suite is proposed in [2]. This work does not employ graphical models, but we include it here because it is based on first-order logic. CE patterns are flexible enough to incorporate both universal and existential quantifiers, as well as alternations of quantifiers. As far as the input SDEs are concerned, they can be either Boolean (with the usual true/false values) or probabilistic, accompanied by an occurrence probability. The method can handle both instantaneous events and events that span over intervals. In order to compute the probabilities of CEs, the dependencies between the SDEs must be modeled as well. For this purpose, triangular norms [27] are used. Triangular norms are binary functions that can model probabilistic dependencies more general than those of exclusivity or independence For example, the *minimum* function can be used as such a norm ($min(x, y)$, where x and y are the probabilities of the two SDEs whose dependence we want to model). The off-line detection algorithm works by recursively decomposing a CE formula into its parts, finding the valid substitutions and keeping those that have a probability above a certain threshold. An on-line algorithm is also presented, which can also detect partially completed activities.

Bayesian Networks. Bayesian Networks are directed graphical models (in contrast to MLNs, which are undirected) whose structure can encode probabilistic dependencies between random variables, represented as nodes in the network. When used for CER, the nodes of the network usually correspond to SDEs and/or CEs. The work presented in [91–93] employs the technique of knowledge-based model construction (KBMC), whereby knowledge representation is separated from the inference process. Each event is assigned a probability, denoting how probable it is that the event occurred with specific values for its attributes. In turn, CE patterns are encoded in two levels, with a selection operation performing an initial filtering, mostly based on event type, followed by a pattern-detection schema for more complex operations, based on temporal and attribute constraints. The selection mechanism imposes certain independence properties on the network. CEs are conditioned only on selectable lower-level events (as determined by the selection operation), preventing the network from being cluttered with many dependency edges. This framework is not limited to representing only propositional or even first-order knowledge. It could potentially handle higher-order knowledge, since the pattern-matching step may, in principle, be defined in any kind of language. However, the system presented [91–93] allows only predicates expressing temporal constraints on event timestamps and equality constraints on event attributes.

The calculation of probabilities for the CEs is done by a Bayesian Network that is dynamically constructed upon each new event arrival. The nodes of the network correspond to SDEs and CEs. First, SDEs are added. Nodes for CEs are inserted only when a rule defining the CE is crisply satisfied, having as parents the events that triggered the rule, which might be SDEs or even other CEs, in case of hierarchical CE patterns. The attribute values of the inferred CEs are determined by mapping expressions associated with the corresponding rule, i.e. functions mapping attributes of the triggering events to attributes of the inferred event. In order to avoid the cost of exact inference, a form of sampling is followed, that bypasses the construction of the network by sampling directly according to the CE patterns.

A more recent effort extends the TESLA [19] event specification language with probabilistic modeling, in order to handle the uncertainty both in input SDEs and in the CE patterns [21]. The semantics of the TESLA language is formally specified by using a first-order language with temporal constraints that express the length of time intervals. At the input level, the system, called CEP2U, supports uncertainty regarding the occurrence of the SDEs, as well as the uncertainty regarding the content of the SDEs. SDEs are associated with probabilities that indicate a degree of confidence,

while the attributes of a SDE are modeled as random variables with some measurement error. The probability distribution function of the measurement error is assumed to be known. The method also models the uncertainty of CE patterns, by automatically building a Bayesian Network for each rule. The probabilistic parameters of the network are manually estimated by domain experts.

Other methods based on Bayesian Networks, which could be used for CE recognition include Bayesian logic programming [32], relational Bayesian Networks [37] and relational dynamic Bayesian Networks [72]. Towards this direction, Dynamic Bayesian Networks have been extended using first-order logic [52, 53]. A tree structure is used, where each node corresponds to a first-order logic expression, e.g., a predicate representing a CE, and can be related to nodes of the same or previous time instances. Compared to their propositional counterparts, the extended Dynamic Bayesian Networks methods can compactly represent CE that involve various entities.

Commentary. Probabilistic graphical models, such as Markov Logic Networks and Bayesian Networks, can provide a substantial degree of flexibility with respect to the probability distributions that they can encode. On the one hand, they are very expressive and they don't require restrictive independence assumptions to be made. On the other hand, this increased flexibility comes at a cost with respect to efficiency. In general, a rule which references certain random variables implies that, before inference can begin, the Cartesian product of all the values of these variables needs to be taken into account. For human activity recognition, one may assume that the number of persons involved in a scene is relatively limited. However, this is not the case for all domains. For a fraud detection scenario, involving transactions with credit cards, a CER system may receive thousands of transactions per second, most of them having different card IDs. The demands of CER exacerbate this problem, since time is a crucial component in these cases. The possible combinations of time points with the other random variables can quickly lead to intractable models. All of the papers discussed in this section employ low-arity (or even 0-arity) predicates, whose arguments have small domain sizes, except for that of time. In order to reduce the unavoidable complexity introduced by the existence of time, they develop special techniques, such as the bottom-up technique in [59].

With respect to probabilistic SDEs, although they can be incorporated into graphical models, correctly encoding their dependencies can be far from obvious, especially with MLNs. Assume we want to assign an occurrence probability of 80% to the *close* SDE of rules (2)–(3). With MLNs, we could replace rule (2) with an equivalence rule, such as:

$$1.39\ \forall X, Y, T\ close(X, Y, T) \leftrightarrow close\_m(X, Y, T) \tag{6}$$

with the appropriate weight (log-odds of occurrence and non-occurrence probabilities), where *close* are the observed SDEs and *close_m* the inferred probabilistic events to be used in other rules, such as (1). It would not be sufficient to directly express rules (3) and (6) in first-order logic and use them to construct an MLN, since the dependencies introduced would mean that the marginal probability of the *close* SDE could be affected by the probability of the $close\_m(X, Y, T_{previous})$ predicate. Bayesian Networks could be used to avoid such problems, due to their directionality. On the other hand, MLNs can be trained as discriminative models, which means that it is not necessary to explicitly encode all the probabilistic dependencies.

5.3 Petri Nets

In order to address issues of concurrency and synchronization, some methods have employed probabilistic extensions of Petri Nets. Formally, a Petri Net may be described as a bipartite directed graph (see [61, 66] for more complete discussions). It has two types of nodes:

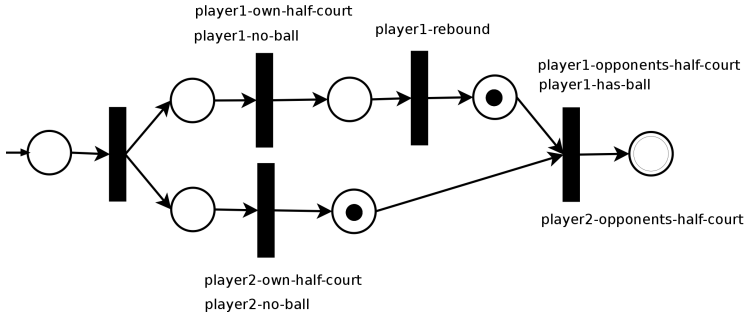

Fig. 2. Basketball transition pattern as a Petri Net.

- *Place nodes*, representing states of the modeled activity (usually depicted as circles). Each such node may hold multiple *tokens*, usually depicted as small, black circles inside place nodes.
- *Transition nodes* (usually depicted as rectangles), which, as the name suggests, connect place nodes. A transition node is said to be *enabled* when all of its input place nodes have tokens, in which case the node may fire. Conditions may be *imposed* on transition nodes to determine exactly when they should fire. *Firing* removes all enabling tokens from the input place nodes and writes a new token at each output node.

A *marking* is a function that assigns tokens to place nodes. When a transition fires, the marking of the Petri Net changes to a new one. A simple example of a rule describing a transition in basketball from defense to offense, encoded as a Petri Net, is shown in Figure 2. The marking in this figure means that *player1* has grabbed a rebound and *player2* is in his team's half-court and does not have the ball. Note that, for the activity to complete, all three conditions of the final transition node must hold, i.e., *player1* must have the ball and both players must be in the opponent team's half-court. This is a transition node that enforces synchronization between the two players' activities. The ability to model constraints about synchronization and concurrency is a powerful feature of Petri Nets.

A probabilistic extension to Petri Nets has been proposed in [3], for recognizing CEs that represent human activities. A Petri Net expresses a CE and is formed by SDEs that are connected with constraints, like temporal durations. The SDEs and other possible constraints are encoded as conditions on transition nodes. Special *dead-end* place nodes are used in order to represent forbidden actions that must immediately trigger an alarm. The transition from one state to another is associated with a probability value. Note that there might be multiple possible transitions from a certain place node (in our example, after *player1* grabs the rebound, he could also make a pass to *player2* as an alternative). The sum of the probabilities of these transitions is 1. Given a sequence of SDEs, the method can identify segments of the sequence in which a CE occurs with probability above a specified threshold or infer the most likely CE in the sequence.

A stochastic variant of Petri-Nets that models the uncertainty of input SDEs is proposed in [46], an issue not addressed in [3]. Specifically, SDEs are recognized with some certainty, using lower level classification algorithms. CEs are represented by Petri Nets, in terms of SDEs that are associated with certainties and temporal constraints. The probability of a CE being recognized is computed through a Bayesian Particle Filter approach. It combines a dynamic model for the probability of moving to a new state (marking) given the previous one and a measurement model for the probability of observing SDEs, given the current state. However, the transitions themselves

cannot be assigned arbitrary probabilities by the user. If there are paths from one marking to other markings (via the SDEs), then the dynamic model assigns equal probabilities to all these paths.

Commentary. One limitation of the approaches that use Petri Nets is the lack of a mechanism for modeling a domain in a truly relational manner, i.e. by allowing relations to be defined between attributes of events. These methods treat events as 0-arity predicates, related only through temporal constraints, as implied by the structure of the Petri Net. As is the case with automata too, Petri Nets tend to make a significant number of independence assumptions. The domain on which they have been tested is that of human activity recognition, in which the sequential nature of some activities allows for the adoption of first-order Markov models. As far as inference is concerned, both MAP and marginal are possible and optimization techniques, such as confidence thresholds and approximate inference have been employed.

5.4 Grammars

A number of research efforts have focused on syntactic approaches to CE recognition. These approaches typically convert a stream of input SDEs to a stream of symbols upon which certain user-defined rules may be applied. Rules are defined via a stochastic grammar (see [84] for a description of stochastic context-free grammars) in order to take uncertainty into account. This can be achieved by assigning a probability value to each production of the grammar. The probabilities of all productions with the same non-terminal symbol on their left-hand side must sum to 1. Stochastic grammars, like the ones described in the rest of this section, have some attractive features. For example, it is easy to define CEs within a single hierarchical grammar. Moreover, they can provide probability evaluations even for "partial matches" of a CE, e.g., for its prefixes, a useful feature for predicting events that might follow.

A two-step approach along these lines is proposed in [36]. Low-level detectors, based on Hidden Markov Models, are used to generate a symbol stream which is then fed into a parser constructed from a stochastic context-free grammar (CFG). This parser in turn employs stochastic production rules in order to determine the probability of a high-level event. Each production rule is accompanied by a probability measure, which defines the conditional probability of this production being chosen, given that its non-terminal is up for expansion. Such probability measures determine how typical the corresponding rules are. However, the presented method is extended in order to take into account probabilistic SDEs as well, so that the competing derivations are weighed not only by the probability measures of the rules, but by the likelihood of the SDEs too. This method can also incorporate production rules that act as background knowledge and produce a grammar that is more "robust", in the sense that it can handle erroneous SDEs. Extensions of this method are presented in [58] and in [56]. In [58], new techniques for error detection and recovery are proposed, without the need to modify the grammar when errors happen (absence or presence of an event that was or was not expected), as is the case in [36]. This allows determine how and when an error occurred and thus also allows for reasoning about such errors. Additionally, an efficient method for parsing activities involving multiple agents is proposed, by partitioning activities into separable groups which can then be handled independently. In [56] context-sensitive grammars are used, i.e., symbols/events may have arguments, allowing for a compact and succinct representation of complex activities.

A hierarchical method is proposed in [71] that combines a syntax for representing the CE patterns of [70] with probabilistic recognition. A syntax similar to that of context-free grammars is used for describing CEs, but the actual recognition is treated as a constraint satisfaction problem. The method aims to probabilistically detect the time intervals in which CEs occur. Input SDEs are associated with probabilities, indicating a degree of belief. Based on a context-free grammar representation

scheme, CE patterns are expressed in terms of other events (SDEs or CEs) and form a hierarchical model. Furthermore, events are related with temporal interval, logical and spatial constraints [6]. In situations where the input stream is incomplete, the method generates the missing SDEs with low confidence. The probability of a CE is calculated by exploiting the dependency information between the CE and its sub-events, as encoded in the hierarchy tree. This method for calculating probabilities is similar to that of Bayesian Networks, the main difference being that siblings are not assumed to be conditionally independent. When the calculated probability of the CE is above a specified threshold, it is considered as recognized.

Moving up to context-sensitive grammars, the work in [65] represents CEs as temporal And-Or Graphs (T-AOG) and attempts both to recognize events from video streams and to infer a person's intentions. In a T-AOG, terminal nodes represent atomic actions (SDEs), i.e., elementary sequences of spatial relations between agents and objects. On top of these atomic actions, a hierarchy of events is constructed, using And-nodes and Or-nodes. And-nodes define temporal (sequential) relations that regulate the durations of its sub-events. All of the sub-events of an And-node must occur. On the contrary, Or-nodes represent different options between events and each such option has an associated probability (for a complete discussion of And-Or Graphs, see [99]). Given a sequence of SDEs, a parsing algorithm incrementally constructs all the possible partial graphs that can interpret the input sequence. In order to limit the number of partial graphs, those whose probability falls below a certain threshold are pruned. The probability of a graph is computed by taking into account the probabilities of the SDEs, the frequency that an Or-node follows a path and the probability that an And-node's temporal relations are satisfied. As a graph is incrementally constructed, the system can also make predictions about the events that are expected to follow. Finally, this work is one of the few that learns CE patterns, by using an unsupervised learning algorithm to build the T-AOG.

Commentary. In [36] the SDEs (terminal symbols) are represented as 0-arity, hence no relations may be defined on attributes. Moreover, when defining a (production) rule, all the possible sub-scenarios (expansions) must be explicitly stated, with probability values that sum to 1. For example, a rule for detecting the *avoid* event, as in rule (4), cannot be written, as:

$$\begin{aligned} 0.9 :: \mathit{Avoid\_Player1\_Player2} \quad &\rightarrow \\ \mathit{Waiting\_Player1\_Player2}, \quad & \\ \mathit{Crossover\_Dribble\_Player2} & \end{aligned} \tag{7}$$

This rule has a probability of 0.9 and therefore we need extra scenarios. All of these scenarios for *avoid* need to be explicitly provided. In this example, rule (5) should be added as:

$$\begin{aligned} 0.1 :: \mathit{Avoid\_Player1\_Player2} \quad &\rightarrow \\ \mathit{Waiting\_Player1\_Player2}, \quad & \\ \mathit{Running\_Player2} & \end{aligned} \tag{8}$$

Note that now the probabilities of the above two production rules must sum to 1. The two scenarios are considered as mutually exclusive. Note also that events are not relational. On the other hand, the addition of sensitivity in [56] and [65] can provide a more expressive power.

6 DISCUSSION

In this last section, we summarize the reviewed approaches and present the weaknesses and strengths of each class of methods. We also discuss the open issues that remain and possible directions of future research.

	Language Expressiveness									
Approach	σ	π	$\vee$	$\neg$	;	*	**W**	**Hierarchies**	**Temporal Model**	**Background Knowledge**
Automata										
SASE+ [40]	✓	✓			✓	✓	✓		Points, Implicit.	
SASE+ AIG [74]	✓	✓			✓	✓	✓		Points, Implicit.	
SASE+ optimized AIS [89]	✓		✓	✓	✓		✓	✓	Points, Implicit.	
SASE++ [96, 97]	✓	✓	✓	✓	✓	✓	✓		Points, Implicit.	
Lahar [68]	✓				✓	✓			Points, Implicit.	
IPF-DA [18]					✓	✓	✓		Points, Implicit.	
Stochastic Automaton [4, 5] (+ PADUA [57])					✓	✓			Points, Implicit.	
First-order logic & Probabilistic Graphical Models										
MLN-Allen [59]	✓	✓	✓	✓	✓		✓	✓	Intervals, Explicit. Allen's Interval Algebra.	✓
MLN-Event Calculus [79, 80]	✓	✓	✓	✓				✓	Points, Explicit. Event Calculus.	✓
MLN-hierarchical [81, 82]			✓	✓	✓			✓	Intervals, Explicit. Allen's Interval Algebra.	✓
ProbLog Event Calculus [78]	✓	✓	✓	✓				✓	Points, Explicit. Event Calculus.	✓
Probabilistic Event Logic [15, 73]			✓	✓	✓			✓	Intervals, Implicit. Allen's Interval Algebra.	✓
Probabilistic Activity Detection [2]	✓		✓	✓	✓				Intervals, Explicit.	
KBMC [91–93]	✓	✓	✓		✓			✓	Points, Explicit.	
CEP2U [21]	✓	✓		✓	✓		✓	✓	Points,Implicit.	
Petri Nets										
Probabilistic Petri Net [3]	✓		✓	✓	✓	✓			Points, Implicit.	
Particle Filter Petri Net [46]			✓		✓				Points, Implicit.	
Grammars										
Stochastic CFG [36]			✓		✓	✓		✓	Intervals, Implicit.	✓
Hierarchical CFG [71]	✓		✓	✓	✓			✓	Intervals, Implicit. Allen's Interval Algebra.	
Temporal And-Or Graphs [65]	✓		✓		✓			✓	Intervals, Implicit.	
Approach	σ	π	$\vee$	$\neg$	;	*	**W**	**Hierarchies**	**Temporal Model**	**Background Knowledge**

Table 1. Expressive power of CER systems. σ: selection, π: projection, $\vee$: disjunction, $\neg$: negation, ;: sequence, *: iteration, W: windowing.

Probabilistic Expressiveness					
Approach	**Model**	**Independence assumptions**	**D.U.**	**P.U.**	**H.C.**
Automata					
SASE+ [40]	Simple multiplication.	All events independent.	Occ.		
SASE+ AIG [74]	Lineage.	SDEs independent.	Occ./ Att.		
SASE+ optimized AIS [89]	Multiplication on Markov chain.	SDEs independent or Markovian. Different streams independent.	Occ.		
SASE++ [96, 97]	Probability distribution on time attribute.	SDEs independent.	Occ.		
Lahar [68]	Probabilistic Relational Algebra.	1st-order Markov for SDEs. Different streams independent.	Occ./ Att.		
IPF-DA [18]	Simple multiplication.	1st-order Markov (+ some extensions).	Occ./ Att.		
Stochastic Automaton [4, 5] (+ PADUA [57])	Patterns modeled as stochastic processes, similar to Markov chains.	1st-order Markov within CE. Different CEs independent.		✓	
First-order logic & Probabilistic Graphical Models					
MLN-Allen [59]	Markov Logic Networks. Bottom-up hypothesis generation.	None.	Occ.	✓	✓
MLN-Event Calculus [79, 80]	Markov Logic Networks.	None.		✓	✓
MLN-hierarchical [81, 82]	Markov Logic Networks.	None		✓	✓
ProbLog Event Calculus [78]	Probabilistic Logic Programming.	SDEs independent.	Occ.		
Probabilistic Event Logic [15, 73]	Weights, as in Conditional Random Fields.	None		✓	✓
Probabilistic Activity Detection [2]	Probabilities assigned to predicates for object equality.	Depends on t-norm.	Occ.		
KBMC [91–93]	Bayesian Networks.	SDEs independent.	Occ./ Att.	✓	
CEP2U [21]	Bayesian Networks.	Event attributes independent. SDEs independent. CEs dependent only on events immediately below in hierarchy.	Occ./ Att.	✓	
Petri Nets					
Probabilistic Petri Net [3]	Hard constraints as forbidden actions. Activities as stochastic processes.	Conditioned on previous event in CE pattern.		✓	✓
Particle Filter Petri Net [46]	Bayesian recursive filter	1st-order Markov. SDEs independent.	Occ.		
Grammars					
Stochastic CFG [36]	Stochastic production rules.	Rules conditionally independent.	Occ.	✓	
Hierarchical CFG [71]	Similar to Bayesian Networks (but siblings not cond. independent)	Conditional independence of SDEs.	Occ.		
Temporal And-Or Graphs [65]	As in Markov Chains for Or-nodes. As in graphical models for And-nodes.	None.	Occ.	✓	
Approach	**Model**	**Independence assumptions**	**D.U.**	**P.U.**	**H.C.**

Table 2. Expressive power of CER systems with respect to their probabilistic properties. D.U.: Data Uncertainty, Occ.:Occurrence, Att.:Attributes, P.U.: Pattern Uncertainty, H.C.: Hard Constraints.

Inference						
Approach	**Type**	**C.T.**	**Ap.**	**D.**	**Efficiency**	**Accuracy**
Automata						
SASE+ [40]	Marginal	✓			0.8-1.1 K events/sec with *Kleene+*.	
SASE+ AIG [74]	Marginal	✓			1000K events/sec, almost constant for varying window size (3 to 15 timepoints). 1000K-100K events/sec for experiments with 1 up to 10 alternatives of a SDE.	
SASE+ optimized AIS [89]	Marginal	✓		✓	8K-13K events/sec for 2-6 nodes.	
SASE++ [96, 97]	Marginal	✓			Reduction from exponential to close-linear cost w.r.t to selectivity / window size.	
Lahar [68]	Marginal	✓			100K events/sec for Extended Regular Queries.	> 10 points increase in accuracy w.r.t. a standard deterministic approach.
IPF-DA [18]	Marginal	✓			4-8K events/sec for patterns of length 6 down to 2.	
Stochastic Automaton [4, 5] (+ PADUA [57])	Marginal and MAP	✓		✓	Running time linear in video length. Parallel version (PADUA) reached 335K events/sec with 162 computing nodes.	PADUA [57] achieved better F-measure than [4, 5].
First-order logic & Probabilistic Graphical Models						
MLN-Allen [59]	Marginal					F-measure > 70% for varying window sizes.
MLN-Event Calculus [79, 80]	Marginal		✓			Increased precision, slight decrease in recall, w.r.t. deterministic solution.
MLN-hierarchical [81, 82]	MAP					
ProbLog Event Calculus [78]	Marginal					Improved F-measure w.r.t. crisp version.
Probabilistic Event Logic [15, 73]	MAP		✓			Smooth accuracy degradation when noise in time intervals of SDEs added. Relative robustness against false positives/negatives.
Probabilistic Activity Detection [2]	Marginal	✓			Running time at most linear in the number of objects in the video sequence.	Better precision/recall than Hidden Markov Models/ Dynamic Bayesian Networks, higher computation time.
KBMC [91–93]	Marginal		✓		Sub-linear decay of event rate w.r.t possible worlds.	CEs within desired confidence interval.
CEP2U [21]	Marginal	✓			50% overhead w.r.t deterministic case.	
Petri Nets						
Probabilistic Petri Net [3]	Marginal and MAP	✓			~ 3 seconds to process videos ~ with 60 different SDE types.	
Particle Filter Petri Net [46]	Marginal		✓			Increased true positive rate w.r.t. deterministic solution. Slight increase in false positive rate.
Grammars						
Stochastic CFG [36]	Marginal	✓				
Hierarchical CFG [71]	Marginal	✓				Increased accuracy when noisy SDEs present, w.r.t. case with crisp SDEs.
Temporal And-Or Graphs [65]	MAP	✓				87%–90% accuracy for predicting human intentions.
Approach	**Type**	**C.T.**	**Ap.**	**D.**	**Efficiency**	**Accuracy**

Table 3. Inference capabilities of probabilistic CER systems. C.T.: Confidence Thresholds, Ap.: Approximate Inference, D.: Distributed

6.1 Summary

Table 1 summarizes the expressive power of the presented CER systems. Its first columns correspond to the list of operators presented in Section 4.1. The other columns assess various aspects of the functionality supported by each system. These are:

- Hierarchies: The ability to define CEs at various levels and reuse those intermediate inferred events in order to infer other higher-level events.
- Temporal Model: Events may be represented by timepoints or intervals. Moreover, the time attribute might be explicitly included in the constraints (e.g. $(T_2 > T_1) \wedge (T_2 - T_1 > 100)$) or temporal constraints may be defined by referring implicitly to time in the rules (e.g. the *Sequence* operator implicitly defines $T_2 > T_1$). When a specific temporal formalism is used (e.g., Allen's Interval Algebra), we mention that as well.
- Background Knowledge: Does the system support knowledge, besides the CE patterns?

In Table 2 we present the probabilistic properties of each method, along the following lines:

- Model: The probabilistic model used.
- Independence assumptions: What are the independence/dependence assumptions made? Note that when we write that a method makes no independence assumptions (cell filled with "None"), the meaning is not that it cannot make such assumptions, but that it does not have to and that it does not enforce them on the probabilistic model.
- Data uncertainty: Does the system support data uncertainty? If yes, is it about the *occurrence* of events or *both* about the occurrence and the attributes?
- Pattern uncertainty: Is there support for uncertain patterns?
- Hard constraints: Is there support for rules that may not be violated?

Table 3 presents the inference capabilities of the presented systems, along the following lines:

- Type: Can the system perform *Marginal* inference, *MAP* inference or *both*?
- Confidence Thresholds: Is there support support for confidence thresholds above which a CE is accepted?
- Approximate: Does the system support techniques for approximate inference?
- Distribution: Is there a distributed version of the proposed solution?
- Efficiency: Remarks about performance with respect to throughput. Note that information about performance is not always available. Moreover, since standard benchmarks for probabilistic CER are not available, the presented figures are those reported by the authors of each approach, who may choose metrics and datasets for their own purposes. This means that a direct comparison is not possible.
- Accuracy: Remarks about performance with respect to precision and/or recall.

Our review of probabilistic CER systems identified a number of strengths and limitations for the proposed approaches. We summarize our conclusions in Table 4. As a note of caution though, when a weakness is reported, this does not mean that the corresponding method cannot in general support a feature (e.g., as might be the case for *iteration* in first-order logic), but that the presented methods have not incorporated it, although it might be possible (for example, the absence of hierarchies in automata-based methods).

The current systems for probabilistic CER need to deal with a trade-off between language expressiveness, probabilistic expressiveness and complexity. As can be seen in Table 4, automata-based methods can easily handle *sequence* and *iteration* operators, but they usually model only data uncertainty, without taking into account pattern uncertainty. Moreover, they rarely move beyond the 1st-order Markov assumption. Therefore, when more expressive power is required, one of the other three approaches should be preferred. For example, Petri Nets are quite powerful for modeling

Approach	Strengths	Weaknesses
Automata	*Iteration*, *Windowing*, formal Event Algebra.	Limited support for event hierarchies. No background knowledge. Implicit time representation (hence no explicit constraints on time attribute).
	Data uncertainty, both with respect to occurrence of events and event attributes.	Limited or no support for rule uncertainty. Too many independence assumptions. No hard constraints.
	Support for confidence thresholds. High throughput values.	Throughput figures come from experiments with simplistic event patterns.
First-order logic & Probabilistic Graphical Models	Complex temporal patterns, with explicit time constraints. Event hierarchies. Background knowledge. Usually provide a formal Event Algebra.	No *Iteration*. Limited support for *Windowing*.
	Pattern uncertainty. Limited independence assumptions. Hard constraints possible.	Harder (but not impossible) to express data uncertainty. Often training is required to assign probabilities/weights to rules.
	MAP and approximate inference.	Low (or unknown) throughput.
Petri Nets	Concurrency and synchronization.	Not truly relational. No *Windowing*, hierarchies or background knowledge. Implicit time representation.
	Support for both data and pattern uncertainty (but not both in the same model).	Strict independence assumptions.
	Can perform both MAP and Marginal inference. Confidence thresholds and approximate inference possible.	Low (or unknown) throughput.
Grammars	Very easy to model hierarchies and *Iteration*. Recursive patterns.	Not truly relational (unless context-sensitive grammars are used). No *Negation*. Implicit time representation. No background knowledge.
	Both data and pattern uncertainty.	Limited methodology for efficient probabilistic inference.
	Confidence thresholds.	Unknown performance for throughput/latency.

Table 4. Strengths and weaknesses of the reviewed probabilistic CER approaches.

concurrency and synchronization. Grammars are very well suited for expressing hierarchies and recursive patterns and can also model both data and pattern uncertainty. If a truly relational model is needed, than first-order logic approaches can provide a solution and can also handle intervals, hierarchies and background knowledge. When combined with probabilistic graphical models, they can also be very flexible as far as their independence assumptions are concerned. However, this increased expressive power comes at the cost of high computational complexity and, as a result, of low throughput.

6.2 Open Issues

Finally, we discuss a few issues that, according to our survey, still remain open and thus provide possible directions for future research.

6.2.1 Performance (Evaluation). Achieving high throughput figures with complex patterns/models so that real-time inference is possible still remains a significant challenge. Moreover, until now very few approaches have explored distributed solutions. The performance trade-offs between throughput and latency have also not been explored yet. There is a general lack of understanding about how the various aspects of probabilistic CER affect not only the accuracy of the produced CEs, but also the efficiency, in terms of latency and throughput. The last two columns of Table 3 present some performance results for each method, but in an informal way. Unfortunately, there is a lack of standard datasets and benchmarks for probabilistic CER (for crisp CER, a set of performance affecting factors is presented in [54] and a benchmark suite in [55]). Therefore, testing and comparing various probabilistic CER methods in a rigorous and formal way still remains an open issue. In order to assess the capabilities of probabilistic CER systems, there is a need for standard benchmarks that would allow for empirical comparisons in various scenarios. Based on our review so far, Table 5 presents a list of factors (first column) that should be taken into account and that could act as possible dimensions upon which a benchmark for probabilistic CER could be developed. First, the behavior of a system when ingesting events whose occurrence is uncertain should be tested. The parameters that can be varied in this case are:

- The occurrence probabilities of the SDEs. When (some of) these probabilities are low, can the system produce meaningful results or does it run into problems (e.g., rounding errors)? When they are high and the events are almost certain, does it produce results that are similar to the ones produced by a crisp system?
- The percentage of SDEs that might be missing or corrupted (incorrect labeling) in the input stream. In this case, certain CEs might not be detected at all (false negatives) or wrong CEs might be produced (false positives).
- Probabilistic dependencies among SDEs. As more complex dependencies are introduced, from independent, identically distributed (i.i.d.) SDEs to SDEs generated by a 1-order (or 2-order, etc.) Markov process and to other processes, can the system maintain acceptable precision and recall scores?
- The number of different event types that may appear in the input stream can affect the complexity of a probabilistic model and thus affect both its accuracy and efficiency.

Another dimension concerns events whose attributes may be uncertain. In this case, the factors that may affect the performance of a system are:

- The type of distribution the attribute values may follow, which can affect both accuracy and efficiency (some distributions might be more complex to handle).
- As in the case of SDE occurrence, some attributes may also be missing or corrupted, even if the event type is correct.
- The way the attributes may depend probabilistically on each other.
- The number of uncertain attributes that an even may have.

With respect to pattern uncertainty, the performance affecting factors are:

- The weights/probabilities of the patterns, which can affect both precision/recall and throughput/latency, especially when optimization techniques are used. For example, a simple optimization technique is that of pruning some processing paths once it is known that a complex event, if detected, will have a low probability. The weights of the patterns can affect such

	Datasets				
Factors	**DEBS'15 Taxi**	**DEBS'14 Energy**	**DEBS'13 Football**	**ILP'16 Agent traces**	**ECML'15 Taxi**
Data uncertainty / Occurrence					
Occurrence probability					
Erroneous/missing SDEs	✓	✓			✓
SDEs i.i.d., m-order Markov process, other processes					
Varying # of event types involved	(1)	(2)	(2)	(1)	(1)
Data uncertainty / Attributes					
Distribution on attribute values (e.g., uniform, Gaussian)					
Erroneous/missing attributes	✓	✓	✓		
Dependencies between attributes					
Varying # of attributes	(17)	(7)	(13)	(3)	(9)
Pattern uncertainty					
Pattern weight/probability (+ hard constraints)					
Arity of predicates involved and domain size					
CE dependencies (independent, Conditional Probability Tables, distributions, etc.)					
Uncertain background knowledge				✓	
Pattern size					
Varying # of patterns	(2)	(3)	(6)	(1)	(2)

Table 5. Factors under test for a possible benchmark for probabilistic CER systems.

techniques, especially when the patterns are unknown and have to be learned, in which case complex dependencies may arise, as CEs may be structurally related.

- The arity of the predicates involved in a pattern (number of attributes) and the number of different values that these attributes may take (domain size). The arity and domain size significantly affect the size of the resulting probability space.
- The probabilistic dependencies between CEs which affect how efficiently and correctly probabilities may propagate, especially when deep hierarchies are present.
- The presence of rules for background knowledge, which may themselves be probabilistic.
- The number of patterns that must be (simultaneously) evaluated and the size of each pattern (i.e., the number of body literals).

A benchmark suite should be able to vary the values corresponding to the above described factors in order to assess their impact on performance. For example, the occurrence probabilities of SDEs

could be varied in a step-wise fashion, within the range 5% – 100% (low probability to certainty). Similarly, the number of event types involved could range from a single event type to numbers that are typical for real-world applications.

The remaining columns of Table 5 correspond to a number of datasets that have been provided in the past for the purpose of running challenges in various conferences. We looked for suitable datasets in the challenges held since 2013 in the following conferences: the International Conference on Distributed and Event–Based Systems (DEBS), the International Conference on Inductive Logic Programming (ILP), the International Conference on Machine Learning (ICML), the European Conference on Machine Learning (ECML) and the Uncertainty in Artificial Intelligence (UAI) conference. Not all of the above conferences actually hold a challenge and access to the challenges and their respective datasets was not always possible. Out of the datasets that we were able access, we have chosen to include datasets that are closer in spirit to CER (e.g., a dataset for predicting house prices based on static features is not useful for CER) and that contain some form of uncertainty. If a dataset includes an evaluation factor, we fill the corresponding cell with a check-mark. Note, though, that none of the datasets has any variation on the listed factors, like SDEs with varying occurrence probabilities or patterns whose number is incrementally increased. For some of the factors, however, we include in parentheses the respective (fixed) numbers for each dataset, as an indication for the order of magnitude. These factors are the following: *number-of-event-types*, *number-of-attributes* and *number-of-patterns*. Below, we provide a brief description of each dataset:

- **DEBS'15**: This was a taxi-related challenge. The dataset was a stream of trip reports from New York City. The SDEs were of a single event type with attributes relaying information about each trip, like its duration, the distance covered, the pick–up and drop–off locations, etc. The goal was two-fold: the identification of recent frequent routes and identification of high profit regions.
- **DEBS'14**: This dataset came from the smart grid domain and focused on analyzing energy consumption measurements, like short-term load forecasting, based on measurements from households. The dataset contained two SDE types, one containing current load (in Watt) and the other accumulated work (in kWh), both for each energy consumption sensor. Three CEs were required to be detected. Two of them concerned load prediction (on a per-house and per-plug basis) and the third one targeted house-based, outlier detection. Semi-formal patterns were given for them, in the form of mathematical formulas.
- **DEBS'13**: This dataset was derived from football matches. Data were collected from sensors located near the players' shoes and in the ball. SDEs consisted of a stream of events with kinematic information from the sensors, such as location, velocity and acceleration. In addition, there were SDEs concerning referee actions. The challenge required the detection of six CEs in total, providing information about player and team statistics, such as ball possession and shots on goal.
- **ILP'16**: This was an artificial dataset concerning an agent learning about its environment. The agent had to move around a grid consisting of cells, with some moves being valid and some not. The SDEs consisted of a series of positions of the agent (a trace) at different timepoints. The goal was to learn a set of rules that could decide (with a confidence value) whether or not a (test) trace was valid. In the probabilistic scenarios of the challenge, there was extra uncertain background knowledge about the grid, e.g., some cells had a certain probability of teleporting the agent to other cells.
- **ECML'15**: The goal of this challenge was to test the ability of a method to predict the final destination of taxis, based on their spatial trajectories. A one-year dataset was provided, consisting of the trajectories performed by 442 taxis running in the city of Porto, in Portugal.

> The SDEs consisted of a single event type and each event contained information about a trip, like its start timestamp, the type of day (e.g., holiday or normal), the trajectory as a polyline, etc. The CEs to be detected had to accurately predict trip destination and travel time.

Note that these datasets were not accompanied by formal definitions for the patterns under test. These had to be either learned or manually defined by the challenge participants. As can be seen in Table 5, the type of uncertainty that currently available datasets usually incorporate is the one that has to do with missing/erroneous events and/or attributes. Ideally, a benchmark should be able to test all possible performance affecting factors. The above listed factors can affect either efficiency (latency/throughput) or accuracy (precision/recall) or both of them. As can be seen in Table 3, automata-based methods tend to focus their experimental evaluation on efficiency (although they can be tested for accuracy as well, e.g. [68] and [4, 5]) whereas the three other classes of methods usually report accuracy results. A comprehensive benchmark should allow for the possibility of exploring the possible trade-offs between efficiency and accuracy when testing the various performance affecting factors.

Towards this goal, the preparation of appropriate datasets would be an important task. As can be seen in Table 5, there are many factors that need to be taken into account. Moreover, CE annotation at an appropriate scale is crucial for testing predictive accuracy. It is likely that not all real-world datasets exhibit all these features. Consequently, benchmarking for probabilistic CER should be augmented by synthetic streams, produced by data generators according to the pre-specified parameters and models displayed in Table 5.

6.2.2 Learning. Learning constitutes another open issue. Only a few of the systems reviewed attempt to learn the weights or structure of CE patterns [15, 65, 73]. Instead, they usually rely on experts to manually define the patterns themselves and/or their weights/probabilities. Manual definitions might be sufficient for simple patterns, but would not scale well in case multiple, complex rules with complicated dependencies are required. This problem is exacerbated for some of the methods (like MLNs) which use the less intuitive notion of weights, encoding the relative strength of a rule inside a theory.

The inherent presence of time in CER presents another challenge. Some statistical relational learning methods might perform well in static domains, but the combinatorial explosion introduced by the presence of time might render them inefficient. Moreover, the patterns may evolve over time and, therefore, they need to be able to adapt to changing conditions if they are to sustain acceptable performance scores. Developing methods for automatically generating and updating pattern definitions in an online fashion that are also resilient to noisy and incomplete data (as is common in Big Data applications) is thus another challenge.

Note that traditional (numerical) machine learning techniques might not always be directly applicable to CER. It is often the case that learning should be able to produce symbolic, relational pattern definitions within some formalism. The reason behind this requirement is that users might need to be able to interpret the learned patterns. Additionally, human experts and analysts often have background knowledge, acquired through experience. It is not always straightforward to encode this knowledge in approaches that are purely numerical. As a result, although numerical techniques (such as neural networks and deep learning; see [35] for a survey on action recognition from visual sources) might be quite powerful for some domains and tasks, they might not suffice in themselves for the purposes of CER learning. There is, however, a trend for using neural networks for relational learning, e.g., as in [13, 83]. Such hybrid techniques, possibly amenable to deep learning architectures, could be a promising direction for future research.

6.2.3 Forecasting. As a final note, we briefly discuss the issue of event forecasting. The field of predictive analytics has gained considerable traction in the last years. Having the ability to forecast that an event will probably occur in the future, given a history of past SDEs, could allow a system to make proactive decisions. The need for event forecasting as a means for proactive behavior has led to proposals about how forecasting could be conceptualized and integrated within a complex event processing system. However, such proposals still remain largely at a conceptual level, without providing concrete algorithms [25, 31]. Most systems for probabilistic CER do not yet possess any forecasting capabilities ([65] is an exception). Some relevant, but non-relational, methods for forecasting have been developed within the field of temporal mining (for a relatively old review, see [47], and for some more recent attempts at event forecasting, see [28, 48, 98]). Relational forecasting, involving events with complex dependencies, is a research area that remains largely unexplored.

REFERENCES

[1] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman. 2008. Efficient pattern matching over event streams. In *SIGMOD*. 147–160.

[2] Massimiliano Albanese, Rama Chellappa, Naresh Cuntoor, Vincenzo Moscato, Antonio Picariello, V. S. Subrahmanian, and Octavian Udrea. 2010. PADS: A Probabilistic Activity Detection Framework for Video Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 12 (2010), 2246–2261.

[3] Massimiliano Albanese, Rama Chellappa, Naresh P. Cuntoor, Vincenzo Moscato, Antonio Picariello, V. S. Subrahmanian, and Octavian Udrea. 2008. A Constrained Probabilistic Petri Net Framework for Human Activity Detection in Video. *IEEE Transactions on Multimedia* 10, 6 (2008), 982–996.

[4] Massimiliano Albanese, Cristian Molinaro, Fabio Persia, Antonio Picariello, and V. S. Subrahmanian. 2011. Finding "Unexplained" Activities in Video. In *IJCAI*. 1628–1634.

[5] Massimiliano Albanese, Vincenzo Moscato, Antonio Picariello, V. S. Subrahmanian, and Octavian Udrea. 2007. Detecting Stochastically Scheduled Activities in Video. In *IJCAI*. 1802–1807.

[6] James F. Allen. 1983. Maintaining Knowledge about Temporal Intervals. *Commun. ACM* 26, 11 (1983), 832–43.

[7] James F. Allen. 1984. Towards a general theory of action and time. *Artificial Intelligence* 23, 2 (July 1984), 123–154.

[8] Alexander Artikis, Opher Etzion, Zohar Feldman, and Fabiana Fournier. 2012. Event Processing Under Uncertainty. In *DEBS*. ACM, 32–43.

[9] Alexander Artikis, Marek Sergot, and Georgios Paliouras. 2010. A logic programming approach to activity recognition. In *Proceedings of the 2nd ACM international workshop on Events in multimedia*. ACM, 3–8.

[10] Alexander Artikis, Marek Sergot, and Georgios Paliouras. 2015. An event calculus for event recognition. *IEEE Transactions on Knowledge and Data Engineering* 27, 4 (2015), 895–908.

[11] Alexander Artikis, Anastasios Skarlatidis, François Portet, and Georgios Paliouras. 2012. Logic-based event recognition. *The Knowledge Engineering Review* 27, 4 (2012), 469–506.

[12] Omar Benjelloun, Anish Das Sarma, Alon Halevy, and Jennifer Widom. 2006. ULDBs: Databases with uncertainty and lineage. In *VLDB*. 953–964.

[13] Hendrik Blockeel and Werner Uwents. 2004. Using neural networks for relational learning. In *ICML-2004 Workshop on Statistical Relational Learning and its Connection to Other Fields*.

[14] Matthew Brand, Nuria Oliver, and Alex Pentland. 1997. Coupled hidden Markov models for complex action recognition. In *CVPR*. IEEE, 994–999.

[15] William Brendel, Alan Fern, and Sinisa Todorovic. 2011. Probabilistic Event Logic for Interval-based Event Recognition. In *CVPR*. IEEE, 3329–3336.

[16] Maurice Bruynooghe, Broes De Cat, Jochen Drijkoningen, Daan Fierens, Jan Goos, Bernd Gutmann, Angelika Kimmig, Wouter Labeeuw, Steven Langenaken, Niels Landwehr, Wannes Meert, Ewoud Nuyts, Robin Pellegrims, Roel Rymenants, Stefan Segers, Ingo Thon, Jelle Van Eyck, Guy Van den Broeck, Tine Vangansewinkel, Lucie Van Hove, Joost Vennekens, Timmy Weytjens, and Luc De Raedt. 2009. An exercise with statistical relational learning systems. (2009).

[17] I. Cervesato and A. Montanari. 2000. A calculus of macro-events: progress report. In *TIME*. 47–58.

[18] Xu Chuanfei, Lin Shukuan, Wang Lei, and Qiao Jianzhong. 2010. Complex Event Detection in Probabilistic Stream. In *APWEB*. 361–363.

[19] G. Cugola and A. Margara. 2010. TESLA: a formally defined event specification language. In *DEBS*. 50–61.

[20] G. Cugola and A. Margara. 2011. Processing Flows of Information: From Data Stream to Complex Event Processing. *Comput. Surveys* 44, 3 (2011).

[21] Gianpaolo Cugola, Alessandro Margara, Matteo Matteucci, and Giordano Tamburrelli. 2014. Introducing uncertainty in complex event processing: model, implementation, and validation. *Computing* (2014), 1–42.

[22] Luc De Raedt and Kristian Kersting. 2003. Probabilistic Logic Learning. *SIGKDD Explor. Newsl.* 5, 1 (2003), 31–48.
[23] Alan Demers, Johannes Gehrke, Mingsheng Hong, Mirek Riedewald, and Walker White. 2006. Towards Expressive Publish/Subscribe Systems. In *EDBT*. Number 3896. 627–644.
[24] Pedro Domingos and Daniel Lowd. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence.* Morgan & Claypool Publishers.
[25] Yagil Engel and Opher Etzion. 2011. Towards Proactive Event-driven Computing. In *ACM DEBS*.
[26] Opher Etzion and Peter Niblett. 2010. *Event Processing in Action.* Manning Publications Company. I–XXIV, 1–360 pages.
[27] Ronald Fagin. 1996. Combining Fuzzy Information from Multiple Systems. In *In Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. ACM Press, 216–226.
[28] Lina Fahed, Armelle Brun, and Anne Boyer. 2014. Efficient Discovery of Episode Rules with a Minimal Antecedent and a Distant Consequent. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management*. Springer.
[29] Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. 2013. Inference and learning in probabilistic logic programs using weighted boolean formulas. *TPLP* (2013), 1–44.
[30] Norbert Fuhr and Thomas Rölleke. 1997. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. *ACM Trans. Inf. Syst.* 15, 1 (1997), 32–66.
[31] Lajos Jenő Fülöp, Árpád Beszédes, Gabriella Tóth, Hunor Demeter, László Vidács, and Lóránt Farkas. 2012. Predictive Complex Event Processing: A Conceptual Framework for Combining Complex Event Processing and Predictive Analytics. In *Proceedings of the Fifth Balkan Conference in Informatics*. ACM.
[32] Lise Getoor and Ben Taskar. 2007. *Introduction to Statistical Relational Learning.* MIT Press.
[33] Matthew L. Ginsberg. 1988. Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence* 4 (1988), 265–316.
[34] Shaogang Gong and Tao Xiang. 2003. Recognition of Group Activities using Dynamic Probabilistic Networks. In *ICCV*, Vol. 2. IEEE, 742–749.
[35] Samitha Herath, Mehrtash Harandi, and Fatih Porikli. 2017. Going deeper into action recognition: A survey. *Image and Vision Computing* (2017).
[36] Y.A. Ivanov and A.F. Bobick. 2000. Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 8 (2000), 852–872.
[37] Manfred Jaeger. 1997. Relational Bayesian Networks. In *13th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 266–273.
[38] Manfred Jaeger. 2008. Model-theoretic expressivity analysis. In *Probabilistic inductive logic programming*. Number 4911. 325–339.
[39] Henry Kautz, Bart Selman, and Yueyen Jiang. 1997. A General Stochastic Approach to Solving Problems with Hard and Soft Constraints. In *The Satisfiability Problem: Theory and Applications*. Vol. 35. AMS, 573–586.
[40] H. Kawashima, H. Kitagawa, and Xin Li. 2010. Complex Event Processing over Uncertain Data Streams. In *3PGCIC*. 521–526.
[41] Kristian Kersting, Luc De Raedt, and Tapani Raiko. 2006. Logical Hidden Markov Models. *JAIR* 25, 1 (2006), 425–456.
[42] S. Khokhar, I. Saleemi, and M. Shah. 2013. Multi-agent event recognition by preservation of spatiotemporal relationships between probabilistic models. *Image and Vision Computing* (2013).
[43] A. Kimmig, B. Demoen, L. De Raedt, V. Santos Costa, and R. Rocha. 2011. On the Implementation of the Probabilistic Logic Programming Language ProbLog. *TPLP* 11 (2011), 235–262.
[44] Robert Kowalski and Marek Sergot. 1986. A Logic-based Calculus of Events. *New Generation Computing* 4, 1 (1986), 67–95.
[45] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*. Morgan Kaufmann, 282–289.
[46] G. Lavee, M. Rudzsky, and E. Rivlin. 2013. Propagating Certainty in Petri Nets for Activity Recognition. *IEEE Transactions on Circuits and Systems for Video Technology* 23, 2 (2013), 326–337.
[47] Srivatsan Laxman and P Shanti Sastry. 2006. A survey of temporal data mining. *Sadhana* (2006).
[48] Srivatsan Laxman, Vikram Tankasali, and Ryen W. White. 2008. Stream Prediction Using a Generative Model Based on Frequent Episodes in Event Sequences. In *ACM SIGKDD*.
[49] Lin Liao, Dieter Fox, and Henry A. Kautz. 2005. Hierarchical Conditional Random Fields for GPS-Based Activity Recognition. In *ISRR*, Vol. 28. 487–506.
[50] David C. Luckham. 2001. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems.* Addison-Wesley.
[51] Jianbing Ma, Weiru Liu, and Paul Miller. 2010. Event modelling and reasoning with uncertain information for distributed sensor networks. In *Scalable Uncertainty Management*. Springer, 236–249.

[52] Cristina Manfredotti. 2009. Modeling and Inference with Relational Dynamic Bayesian Networks. In *Advances in Artificial Intelligence*. Vol. 5549. 287–290.
[53] Cristina Manfredotti, Howard Hamilton, and Sandra Zilles. 2010. Learning RDBNs for Activity Recognition. In *NIPS Workshop on Learning and Planning from Batch Time Series Data*.
[54] Marcelo RN Mendes, Pedro Bizarro, and Paulo Marques. 2009. A performance study of event processing systems. In *Performance Evaluation and Benchmarking*. 221–236.
[55] Marcelo R.N. Mendes, Pedro Bizarro, and Paulo Marques. 2013. Towards a Standard Event Processing Benchmark. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*. ACM.
[56] D. Minnen, I. Essa, and T. Starner. 2003. Expectation grammars: leveraging high-level expectations for activity recognition. In *CVPR*, Vol. 2. II–626–II–632 vol.2.
[57] Cristian Molinaro, Vincenzo Moscato, Antonio Picariello, Andrea Pugliese, Antonino Rullo, and V. S. Subrahmanian. 2014. PADUA: Parallel Architecture to Detect Unexplained Activities. *ACM (TOIT)* 14, 1 (2014), 3:1–3:28.
[58] Darnell Moore and Irfan Essa. 2002. Recognizing multitasked activities from video using stochastic context-free grammar. In *AAAI/IAAI*. 770–776.
[59] Vlad I. Morariu and Larry S. Davis. 2011. Multi-agent event recognition in structured scenarios. In *CVPR*. 3289–3296.
[60] Stephen Muggleton and Jianzhong Chen. 2008. A Behavioral Comparison of Some Probabilistic Logic Models. In *Probabilistic Inductive Logic Programming*. Number 4911. 305–324.
[61] T. Murata. 1989. Petri nets: Properties, analysis and applications. *Proc. IEEE* 77, 4 (1989), 541–580.
[62] Kevin P. Murphy. 2002. *Dynamic Bayesian Networks: representation, inference and learning*. Ph.D. Dissertation. University of California.
[63] Adrian Paschke. 2006. ECA-RuleML: An approach combining ECA rules with temporal interval-based KR event/action logics and transactional update logics. *arXiv preprint cs/0610167* (2006).
[64] Adrian Paschke and Martin Bichler. 2008. Knowledge representation concepts for automated SLA management. *Decision Support Systems* 46, 1 (2008), 187–205.
[65] Mingtao Pei, Zhangzhang Si, Benjamin Z. Yao, and Song-Chun Zhu. 2013. Learning and parsing video events with goal and intent prediction. *Computer Vision and Image Understanding* 117, 10 (2013), 1369–1383.
[66] James Lyle Peterson. 1981. *Petri net theory and the modeling of systems*. Prentice-Hall.
[67] Lawrence R. Rabiner and Biing-Hwang Juang. 1986. An introduction to Hidden Markov Models. *ASSP Magazine* 3, 1 (1986), 4–16.
[68] Christopher Ré, Julie Letchner, Magdalena Balazinksa, and Dan Suciu. 2008. Event Queries on Correlated Probabilistic Streams. In *ACM SIGMOD*. 715–728.
[69] M. Richardson and P. Domingos. 2006. Markov logic networks. *Machine Learning* 62, 1–2 (2006), 107–136.
[70] Michael S. Ryoo and Jake K. Aggarwal. 2006. Recognition of Composite Human Activities through Context-Free Grammar Based Representation. In *CVPR*. 1709–1718.
[71] Michael S. Ryoo and Jake K. Aggarwal. 2009. Semantic Representation and Recognition of Continued and Recursive Human Activities. *International Journal of Computer Vision* 82, 1 (2009), 1–24.
[72] Sumit Sanghai, Pedro Domingos, and Daniel Weld. 2005. Relational dynamic Bayesian networks. *JAIR* 24, 2005 (2005), 759–797.
[73] Joseph Selman, Mohamed R. Amer, Alan Fern, and Sinisa Todorovic. 2011. PEL-CNF: Probabilistic event logic conjunctive normal form for video interpretation. In *ICCVW*. IEEE, 680–687.
[74] Zhitao Shen, Hideyuki Kawashima, and Hiroyuki Kitagawa. 2008. Probabilistic event stream processing with lineage. In *Proc. of Data Engineering Workshop*.
[75] Vinay D. Shet, Jan Neumann, Visvanathan Ramesh, and Larry S. Davis. 2007. Bilattice-based Logical Reasoning for Human Detection. In *(CVPR)*. IEEE Computer Society, 1–8.
[76] Vinay D. Shet, Maneesh Singh, Claus Bahlmann, Visvanathan Ramesh, Jan Neumann, and Larry S. Davis. 2011. Predicate Logic Based Image Grammars for Complex Pattern Recognition. *International Journal of Computer Vision* 93, 2 (2011), 141–161.
[77] Jeffrey Mark Siskind. 2001. Grounding the Lexical Semantics of Verbs in Visual Perception using Force Dynamics and Event Logic. *JAIR* 15 (2001), 31–90.
[78] Anastasios Skarlatidis, Alexander Artikis, Jason Filippou, and Georgios Paliouras. 2013. A Probabilistic Logic Programming Event Calculus. *TPLP* (2013).
[79] Anastasios Skarlatidis, Georgios Paliouras, Alexander Artikis, and George A. Vouros. 2015. Probabilistic Event Calculus for Event Recognition. *ACM Trans. Comput. Logic* 16, 2 (2015).
[80] Anastaios Skarlatidis, Georgios Paliouras, George Vouros, and Alexander Artikis. 2011. Probabilistic Event Calculus Based on Markov Logic Networks. In *RuleML*, Vol. 7018. 155–170.
[81] Young Chol Song, Henry Kautz, James Allen, Mary Swift, Yuncheng Li, Jiebo Luo, and Ce Zhang. 2013. A Markov Logic Framework for Recognizing Complex Events from Multimodal Data. In *ACM ICMI*. 141–148.

[82] Young Chol Song, Henry A. Kautz, Yuncheng Li, and Jiebo Luo. 2013. A General Framework for Recognizing Complex Events in Markov Logic. In *AAAI Workshop: Statistical Relational Artificial Intelligence*.

[83] Gustav Šourek, Vojtech Aschenbrenner, Filip Železny, and Ondřej Kuželka. 2015. Lifted relational neural networks. In *Proceedings of the 2015th International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches-Volume 1583*.

[84] Andreas Stolcke. 1995. An Efficient Probabilistic Context-free Parsing Algorithm That Computes Prefix Probabilities. *Comput. Linguist.* 21, 2 (1995), 165–201.

[85] Son Dinh Tran and Larry S. Davis. 2008. Event Modeling and Recognition Using Markov Logic Networks. In *ECCV*, Vol. 5303. 610–623.

[86] Douglas L. Vail, Manuela M. Veloso, and John D. Lafferty. 2007. Conditional Random Fields for Activity Recognition. In *AAMAS*. 1331–1338.

[87] Sarvesh Vishwakarma and Anupam Agrawal. 2013. A survey on activity recognition and behavior understanding in video surveillance. *The Visual Computer* 29, 10 (2013), 983–1009.

[88] Yijie Wang, Xiaoyong Li, Xiaoling Li, and Yuan Wang. 2013. A survey of queries over uncertain data. *Knowledge and Information Systems* 37, 3 (2013), 485–530.

[89] Y. H. Wang, K. Cao, and X. M. Zhang. 2013. Complex event processing over distributed probabilistic event streams. *Computers & Mathematics with Applications* 66, 10 (2013), 1808–1821.

[90] Segev Wasserkrug, Avigdor Gal, and Opher Etzion. 2006. A Taxonomy and Representation of Sources of Uncertainty in Active Systems. In *Next Generation Information Technologies and Systems*. Number 4032. Springer Berlin Heidelberg, 174–185.

[91] Segev Wasserkrug, Avigdor Gal, and Opher Etzion. 2012. A Model for Reasoning with Uncertain Rules in Event Composition Systems. *arXiv:1207.1427 [cs]* (2012).

[92] Segev Wasserkrug, Avigdor Gal, Opher Etzion, and Yulia Turchin. 2008. Complex event processing over uncertain data. In *DEBS*. ACM, 253–264.

[93] S. Wasserkrug, A. Gal, O. Etzion, and Y. Turchin. 2012. Efficient Processing of Uncertain Events in Rule-Based Systems. *IEEE TKDE* 24, 1 (2012), 45–58.

[94] Eugene Wu, Yanlei Diao, and Shariq Rizvi. 2006. High-performance Complex Event Processing over Streams. In *ACM SIGMOD*. 407–418.

[95] Tsu-yu Wu, Chia-chun Lian, and Jane Yung-jen Hsu. 2007. Joint Recognition of Multiple Concurrent Activities using Factorial Conditional Random Fields. In *PAIR*. 82–88.

[96] Haopeng Zhang, Yanlei Diao, and Neil Immerman. 2010. Recognizing Patterns in Streams with Imprecise Timestamps. *VLDB* 3, 1-2 (2010), 244–255.

[97] Haopeng Zhang, Yanlei Diao, and Neil Immerman. 2014. On complexity and optimization of expensive queries in complex event processing. ACM Press, 217–228.

[98] Cheng Zhou, Boris Cule, and Bart Goethals. 2015. A pattern based predictor for event streams. *Expert Systems with Applications* (2015).

[99] Song-Chun Zhu and David Mumford. 2007. *A stochastic grammar of images*. Number 2:4. Now.

Received ; revised ; final version ; accepted