Complex event processing under uncertainty using Markov chains, constraints, and sampling

Romain Rincé, Romain Kervarc, Philippe Leray

Complex Event Processing under Uncertainty Using Markov Chains, Constraints, and Sampling

Romain Rincé[1,2], Romain Kervarc[1], and Philippe Leray[2]

[1] ONERA – The French Aerospace Lab, Palaiseau
[2] LS2N – Laboratoire des Sciences du Numérique de Nantes, UMR CNRS 6004
Université de Nantes, France

Abstract. For the last two decades, complex event processing under uncertainty has been widely studied, but, nowadays, researchers are still facing difficult problems such as combinatorial explosion or lack of expressiveness while inferring about possible outcomes. Numerous approaches have been proposed, like automaton-based methods, stochastic context-free grammars, or mixed methods using first-order logic and probabilistic graphical models. Each technique has its own pros and cons, which rely on the problem structure and underlying assumptions. In our case, we want to propose a model providing the probability of a complex event from long data streams produced by a simple, but large system, in a reasonable amount of time. Furthermore, we want this model to allow considering prior knowledge on data streams with a high degree of expressiveness.

1 Introduction

Since the end of the last century, the explosion of computerisation leads to growing amount of data to monitor. Detecting interesting or suspicious behaviours along data streams is a key goal on safety-critical systems. As those systems might be composed of numerous agents, their behaviours might be hidden by the interwine data produced by each agent. Complex Event Processing (CEP) focuses on analysing these data streams to detect interesting behaviours along the whole duration of a system execution. We usually call the information produced by a system Low Level Events (LLEs) and the behaviours complex events (CEs), as they are LLEs compositions.

Many CEP techniques allow monitoring events from a data stream, but, as most information on a system is detected through sensors or human knowledge, it may contain errors, mostly erroneous detections (e.g. missing an event, leading to a specific behaviour not being recognized), or false detections (e.g. an event that actually never occurred) leading to false recognitions. Uncertainty may also appear when CEs are designed, since they are usually defined based on human knowledge or through statistical analysis, which may miss specific cases where the desired behaviour emerges. Many authors address uncertainty in various ways. [1], [2] use Non Deterministic Finite Automata associated to probabilities to compute the likelihood of a CE. Other authors represent CEP as a logical problem and use specific Probabilistic Graphical Models (PGMs) to calculate probabilities, such as Markov Logic Networks (MLNs) [3], [4], or Problog [5]. [6] represent CEs as stochastic context-free grammars. These techniques will be

discussed in deeper detail later, are roughly divided between exact inference (all possible outcomes are evaluated to compute the probabilities as in [5], [7]) and approximate inference (relying on PGMs and sampling to compute the probabilities like in [3], [8]).

The way uncertainty is dealt with in CEP also depends on system specifics: long time relation, kind of uncertainty (on events, attributes, time, model design...,) online inference, attributes with continuous values, discrete or continuous time, computational speed, exact or approximate evaluation, etc. Addressing all possible cases in a single method seems unachievable, so each work focuses on its own specific constraints. Consequently, literature offers a wide variety of approaches, not only technically but also conceptually. Some focus on event attribute uncertainties but without considering long time relations, while others do the opposite.

In this paper, we propose a method to estimate CE recognition likelihood in a reasonable amount of time from a potentially erroneous data-stream of LLEs produced by a system. We consider uncertainty on events, i.e. event detection is not guaranteed (they may be missed or incorrectly detected). More formally, we aim at producing probability estimations of CEs, even with long term relations, close to methods preforming exact computation, but with a better scalability to larger problems. We want prior knowledge to be provided to constrain possible executions of the system and taken into account for the probability computation. This knowledge might be expressed as LLEs or CEs.

On our model, we use the chronicle formalism [9], [10], which describes CEs by association of LLEs or previously defined CE using interval operators. Probability estimation will be computed using Markov Chain Monte-Carlo (MCMC) on a non-homogenous Markov process (NHM) constructed using an initial Markov chain (MC) M that describes the system producing the events of our stream. The noisy data stream is considered as prior knowledge on the execution of our system and hence as constraints on M. Regarding all constraints and M, we define the NHM as the representation of all possible executions. It produces samples for the MCMC using random walk. Creating the NHM requires prior knowledge that is all the LLEs observed in the data stream, but we want to extend our method to be able to use an external information represented as a CE to constrain the NHM. This CE would be a chronicle representing a specific behaviour that as be observed during the production of the data stream.

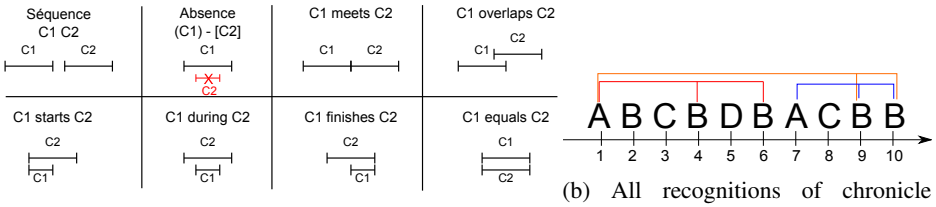
Sect. 2 presents the techniques and CEP representation that we use. Sect. 3 and 4 show how we estimate the likelihood of a chronicle on a stream and how we express knowledge at a high level. Performance comparison with Problog and MLNs is provided in Sect. 5 and Sect. 6 discusses our solution and compare it with other approaches.

2 Background

Here, we provide insight of the necessary background for our contribution. First, we present the chronicles which is the CEP system that we used for recognitions, then we remind basic knowledge on NHM and Deterministic Finite Automata (DFA).

2.1 The Chronicle Model

Many CEP techniques exist like SASE+, T-REX, Dousson's chronicles, Event Calculus. [11]–[14], but for our method we will focus on *Chronicles* [9], [10].



(a) Visual representation of Allen's operators [15]

(b) All recognitions of chronicle $A((BB) - [C])$

Fig. 1: Chronicle formalism

Like all the above methods, chronicles aim at detecting meaningful information within temporal data flows. This information is usually a behaviour or a specific activity that will be model into a CE. Many of these methods have been modified to deal, on their own way, with uncertainty and we aim to extend the Chronicle model too.

For the chronicle model, data is composed of LLEs, i.e. events with their detection times, upon which logical formulae, the chronicles (i.e. CEs) are built inductively using Allen's operators[3] (cf. Fig. 1a). Two events might be associated together, based on their time points, to compose a chronicle. Regarding the operator used, a chronicle describe a behaviour or activity on a time interval. As the operators associate intervals, it is possible to compose more complex chronicles using previously defined chronicles. These operators allow the chronicles to define long-term relations between events, since they represent more the positioning of events relative to each other than a specific duration of time. Consequently, it is possible to recognise CEs taking place on the full duration of a data stream. Fig. 1b presents the recognitions of a chronicle where an event A precedes two events B, without any event C between the two B, which is denoted $A((BB) - [C])$. Here, $(BB) - [C]$ is a sub-chronicle.

CEP techniques commonly manage attributes associated with LLEs and CEs, which may be used to compose specific recognition rules, but for our approach, we will ignore attributes i.e. consider only nullary events. We first want to focus on a method performing inference in relative short amount of time, even with numerous different types of event along relatively quite long data streams. Dealing with attributes under uncertainty is an additional subject that we will certainly study in future work but is not addressed here. To avoid confusion below, *events* will refer to LLE and *chronicles* to CEs.

2.2 Discrete Non-Homogeneous Markov Chains from Markov Process

An important part of our method consists in producing a NHM that represents a system modelled by a Markov process and constraints on its execution. We now outline the creation of the NHM as proposed by [16].

The method aims to produce sequences of states from a Markov process M defined over a finite set of states $\mathcal{A}$. Given a length L, S is the set of all sequences of length L that may be produced by M with non-zero probabilities. For a sequence $s = s_1, \ldots, s_L, s \in S, s_i \in \mathcal{A}$:

$$p_M(s) = p_M(s_1) \cdot p_M(s_2|s_1) \cdots p_M(s_L|s_{L-1})$$

[3] Chronicles use over 15 interval operators: Allen's and duration-related constraints – cf. [9].

We now consider a subset of sequences $S_C \subseteq S$ that should follow a set of constraints C. These constraints are unary, describing a set of states possible at a given time, or binary, describing a set of transitions between two consecutive times. The constraints are embedded into a NHM represented by a set $\tilde{M}$ of transition matrices $\tilde{M}^{(i)}, i = 1, \ldots, L$ where $p_{\tilde{M}}(s_i|s_{i-1}) = \tilde{M}^(i)$ and $\tilde{M}$ verifies the two properties :

- $p_{\tilde{M}}(s) = 0$ for $s \notin S_C$
- $p_{\tilde{M}}(s) = p_M(s|S_C)$ otherwise

Briefly, computation of the NHM works as follows: compute the intermediate matrices $Z^{(0)}, \ldots, Z^{(t)}, \ldots, Z^{(L-1)}$ that are the equals to M but with all the impossible transitions between $t-1$ et t, regarding the constraints, set to 0. Then $\tilde{M}$ is computed using the following formula:

$$\begin{aligned}
\tilde{m}_{j,k}^{(L-1)} &= \frac{z_{j,k}^{(L-1)}}{\alpha_j^{(L-1)}}, \quad & \alpha_j^{(L-1)} &= \textstyle\sum_{k=1}^{n} z_{j,k}^{(L-1)} \\
\tilde{m}_{j,k}^{(i)} &= \frac{\alpha_k^{(i+1)} z_{j,k}^{(i)}}{\alpha_j^{(i)}}, \quad & \alpha_j^{(i)} &= \textstyle\sum_{k=1}^{n} \alpha_k^{(i+1)} z_{j,k}^{(i)} \quad 0 < i < L-1 \\
\tilde{m}_{k}^{(0)} &= \frac{\alpha_k^{(1)} z_{j,k}^{(0)}}{\alpha_j^{(0)}}, \quad & \alpha_j^{(1)} &= \textstyle\sum_{k=1}^{n} \alpha_k^{(1)} z_{k}^{(0)}
\end{aligned} \tag{1}$$

where n is the number of states. As shown by Pachet *et al.* [16] $\tilde{M}$ generates exactly the sequences $s \in S_C$ and that these sequences have the same probabilities in M and $\tilde{M}$ up to a constant factor.

2.3 Deterministic Finite Automaton

In this paper, we use DFA to represent a set of event sequences S of same length L. This set may be considered as a language $\mathcal{L}$ over the alphabet Σ which is the set of all the events appearing on S. As all sequences have the same length L, $\mathcal{L}$ might be represented as an acyclic DFA and minimized in linear time [17]. As a reminder, a DFA $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$ is defined with $\mathcal{Q}$ a finite set of states, Σ an alphabet, $\delta : \mathcal{Q} \times \Sigma \mapsto \mathcal{Q}$ a transition function, $q_0 \in \mathcal{Q}$ an initial state, and $\mathcal{F} \subseteq \mathcal{Q}$ the set of final states.

3 Probability Estimation of a Chronicle on Noisy Data Stream

3.1 Process

Our goal is to define a model able to estimate the probability of a given chronicle ch on a specific data stream. This chronicle might be seen as a query on the stream. In this paper, we will focus on calculating the likelihood that at least one recognition of the chronicle appears on the stream, but we could ask for any specific number of recognitions. We also assume that the stream results from sensor observations of an actual system composed of sub-systems changing their states during time. We consider time as discrete and events are subsets of the current state of the sub-systems at given times. We suppose that our system may be described as a discrete, stationary, first-order MC M whose states are the cartesian product of sub-systems states.

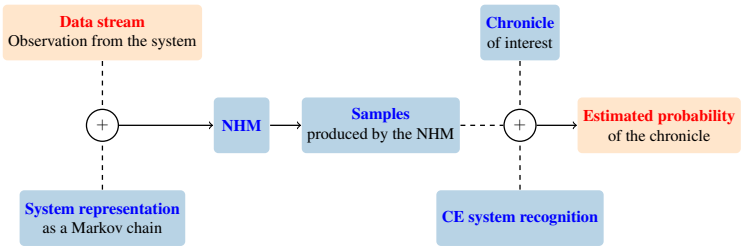


Fig. 2: Probability estimation process

Information provided on streams is supposed to be our observations and may be erroneous or noisy regarding the system observed. For instance, imagine that a system is observed using sensors sensible to errors. Note that this modelling handles missing or erroneous detections but not incorrect detection times: two state change detections swapped in time may not be represented.

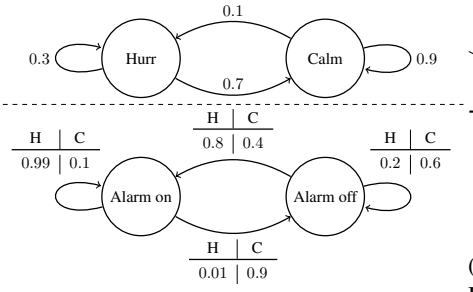
Our process to estimate probability (Fig. 2) consists in transforming the data stream into a set of unary and binary constraints and used them with the Markov model, as presented in Sect. 2.2, to construct the corresponding NHM $\tilde{M}$. Using random walk on this NHM, we generate a sufficient amount of possible streams of states that explains the prior data stream. With a sufficient amount of samples, the probability of each sequence should be close to the distribution provided by the NHM. Each sample produced may be analysed through a CE recognition system that provides the number of recognitions of ch. Estimating its probability just consists in counting the number of samples that recognise ch among all.

3.2 Toy Example: an Hurricane Alarm

In this section, we illustrate our process with a toy example. Suppose that we use the system described at Fig. 3a. The model is composed of two subsystems, `Hurricane` and `Alarm`, having different possible states. The `Hurricane` part might be on state `Hurr` or `Calm` meaning respectively that there is currently a hurricane or not. The `Alarm` part might be on state `On` or `Off` meaning respectively that the hurricane alarm is triggered or not. The alarm is not completely safe and might trigger without hurricane or reciprocally not triggering during a hurricane. It represents the behaviour of a hurricane alarm regarding the weather and its own state. Transition probabilities are provided in Fig. 3a. The alarm may be considered as the sensor used to detect a hurricane. So the data stream provides the timely state of the alarm, whilst the state of the weather might almost never be provided.

For now, we transform our model into a simple MC M where its states are combinations of the states of the two sub-systems. The resultant matrix is given on Table 3b.

According to [16], we can constrain the MC using binary and unary constraints at given times: to express e.g. that at $t = 2$ the alarm is off and at $t = 3$ a hurricane rises, the NHM will be constructed by removing inconsistent transitions at corresponding times. We also suppose that at $t = 0$ there is neither an alarm nor a hurricane. This



(a) Hurricane alarm model and transition probabilities

t \ $t+1$	C & Off	C & On	H & Off	H & On
C & Off	0.54	0.36	0.02	0.08
C & On	0.81	0.09	0.001	0.099
H & Off	0.42	0.28	0.06	0.24
H & On	0.63	0.07	0.003	0.297

(b) Hurricane alarm Markov chain (C:Calm, H:Hurr, Off:Alarm off, On:Alarm on)

Fig. 3: Hurricane alarm model representation

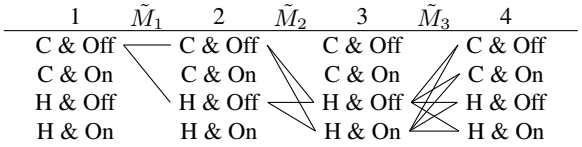


Fig. 4: Graphical representation of the NHM from instant 1 to 4. Constraints : Alarm off at t=1 & t=2, Hurricane at t=3

information is our constraint set S_C. Regarding constraints, the dependency graph of states between instants might be represented as Fig. 4. The probabilities are re-evaluated using back-propagation as presented in Sec. 2.2 (cf. Eq. 1).

Sampling is performed by random-walk, selecting a path regarding the probabilities given by the NHM. Each sample produced should then be parsed through a CEP engine. After enough samples, we will be able to estimate chronicle likelihood by counting the number of recognitions over all samples. This method provides the likelihood of a given chronicle but prior knowledge may only be represented as a succession of possible states at given times. But it could be interesting to express it as a behaviour. In our toy example, we might e.g. know that between two times a hurricane rised but the alarm never triggered. The next section provides a method to estimate a chronicle likelihood regarding this kind of knowledge.

4 Chronicle Probability Estimation under High-level Constraints

We described a method to approximate using sampling the probability of a chronicle recognition under unary and binary constraints, and now want to extend it to constraints expressed in a higher formalism, namely chronicles. Formally, it means that we do not provide the exact time of a known event but an interval where a CE appears.

In this section, we show how to express these constraints as a sub-stream set, which we then regroup into graphs to reduce the samples needed, and a method to construct graphs. These transformations from chronicles as constraints to a constraint sub-graph are detailed on Fig. 6, which focuses on a part of the full process (on Fig. 5).

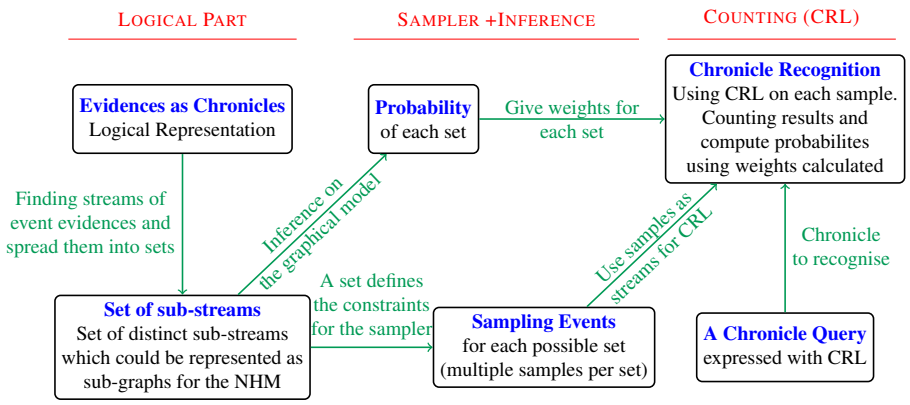


Fig. 5: Architecture of the sampling model

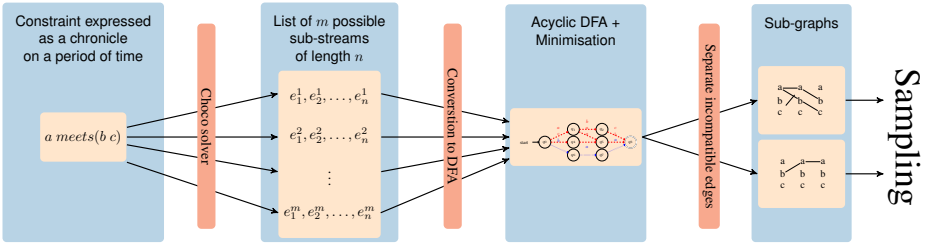


Fig. 6: Process road map to transform high-level constraints

4.1 Constraint Sub-streams to Represent Behaviours

Expressing a chronicle as a constraint over an interval is equivalent to consider as constraints all possible sub-streams Φ producing a recognition of this chronicle on this interval. In this paper, Φ will represent all the possible execution of the system regarding the constraints during two instants of time. Consequently, one sub-stream is a possible execution of the system on this interval of time. Finding all the sub-streams is not our main purpose here, and may be defined as a Constraint Satisfaction Problem (CSP) handled by a solver. To avoid combinatorial explosion of the solving, the chronicle is used as a constraint over a restricted time interval and not over the whole stream. For instance, specifying a sequence of two states as a constraint in a time interval gathers all the possible outcomes of stream where this chronicle should be recognised.

Considering that we have Φ, a naive approach would be to sample over each of these sub-streams $\varphi \in \Phi$, since each could be expressed with unary and binary constraints. Given the set Φ of sub-streams, the probability of a sample s on these constraints would be equal to:

$$P(s|\Phi) = \sum_{\varphi_i \in \Phi} P_M(\varphi_i|\Phi) \times P_{\tilde{M}_{\varphi_i}}(s) \tag{2}$$

where $P_{\tilde{M}_{\varphi_i}}(s) = P_M(s|\varphi_i)$ is the probability that the sample is generated by the model M under the constraint φ_i.

But, as no φ_i is included in another φ_j, the sequences that they might generate are independent. Consequently, if a stream s follows the constraints Φ, it can be generated by only one $\varphi' \in \Phi$, meaning only one $P_{\tilde{M}_{\varphi'}}(s)$ has a non-zero probability. Furthermore, because of the independence:

$$P_M(\varphi_i|\Phi) = \frac{P_M(\varphi_i)}{P_M(\Phi)} \tag{3}$$

if φ' is the sub-stream producing s, Eq. 2 may be rewritten:

$$P(s|\Phi) = \frac{P_M(\varphi')}{P_M(\Phi)} \times P_{\tilde{M}_{\varphi'}}(s) \tag{4}$$

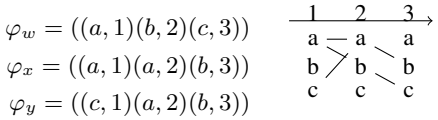
However, this approach requires sampling over all sub-streams and their number grows exponentially on their length (depending of course on the chronicle: the more it is constraining, the less possible outcomes), it could lead to a drastic amount of sampling even using parallel computing.

4.2 Representing Constraint Sub-streams as Sub-graphs

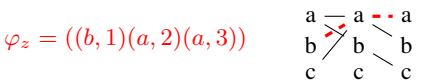
To reduce the number of necessary samplings, we take advantage of the NHM structure to represent the constraint streams: using local binary constraints, more than one sub-stream may be represented in an accepting sub-graph of the NHM.

Definition 1. A constraint sub-graph is *accepting* for the constraint sub-streams if all paths are in bijection with them.

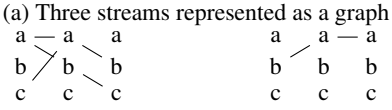
For instance, we define a constraint Φ from the streams $\varphi_w, \varphi_x, \varphi_y$ in Fig. 7a which maybe represented in one sub-graph. This could reduce drastically the number of needed sampling as the three streams are now embedded in only one sub-graph. But if we want to add a fourth constraint φ_z to Φ directly (dashed in Fig. 7b), the graph is no longer accepting as it recognizes sub-stream $((a,1)(a,2)(a,3))$ as a valid constraint. In that case, it is necessary to split the set of constrained streams into two sub-graphs (Fig. 7c). Indeed, conditions to split are highly correlated with the full set of constraint streams. E.g., assuming that we consider the three streams in Fig. 7d as constraints in Φ, the graph in Fig. 7b is accepting.



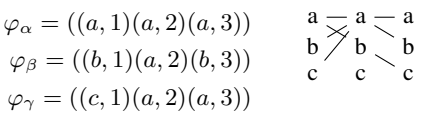
(a) Three streams represented as a graph



(b) Non accepting graph for $\Phi \bigcap \varphi_z$



(c) Accepting graphs for $\Phi \bigcap \varphi_z$



(d) Streams making the graph accepts φ_z

Fig. 7: Examples of accepting/ not accepting sub-graphs

4.3 Finding a Set of Constraint Sub-graphs

To perform the transformation, we will define our set of constraint sub-streams as a language $\mathcal{L} \subseteq \Sigma^n$ on the alphabet $\Sigma = \{a, b, c\}$ defined as the set of states $\mathcal{A}$ from a corresponding NHM, and n the length of the streams. As we pointed out, $\mathcal{L}$ might be represented as an acyclic DFA. E.g., the streams $\varphi_w, \varphi_x, \varphi_y, \varphi_z$ might be represented as the DFA in Fig. 8. Using this automaton, it is possible to identify streams that could not appear on the same sub-graph.

Definition 2. Two edges of the DFA are incompatible if they are at the same depth, share the same transition from Σ leading to two distinct states in $\mathcal{Q}$. Two sub-streams are incompatible if they each contain one of two incompatible edges.

In our DFA in Fig. 8, transitions between q_1, q_5 and q_4, q_5 lead to the same state and thus are not incompatible. By contrast, transitions between q_4, q_5 and q_6, q_7 are incompatible as they are at the same depth from the initial state, depend on the same symbol a, and lead to different states. It is now possible to separate streams according to the incompatibility expressed through the DFA (cf. an example in Fig. 8 where each set is dashed differently), which is performed by Alg. 1, a breadth-first recursive algorithm on depth: from the initial node it separates at each level incompatible edges into sets and creates for each a new automaton with all accessible ancestors and descendants, until reaching all final states.This algorithm does not necessarily provide a minimal splitting in terms of set number, but ensures that the resultant sets are indeed independent, and is sufficient for our purpose.

4.4 Sampling with Constraint Sub-graph

The likelihood has to be estimated differently, but is still similar to Eq. 2, as we do not sample over a simple constraint sub-stream but over a set of sub-streams. The marginal probability of a set $P_M(\Phi)$ is the product of the unnormalized constrained matrices of the NHM, which is easily evaluated during back-propagation. Given a sub-set $\Phi_s \subseteq \Phi$ that can produce a sequence s, the probability of s regarding Φ_s:

$$P(s|\Phi) = \frac{P_M(\Phi_s)}{P_M(\Phi)} \times P_{\tilde{M}_{\Phi_s}}(s) \tag{5}$$

5 Experimentations

In this section, we will compare our approach with Problog [18], which has been used successfully to manage uncertainty in CEP and produces exact results. As Problog is

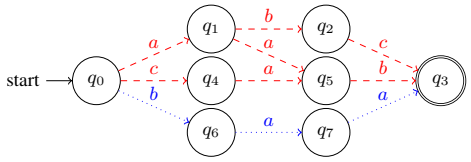


Fig. 8: Minimal automaton for the streams $\varphi_w, \varphi_x, \varphi_y, \varphi_z$ and resulting separation.

Algorithm 1: Split_automaton

inputs : $\mathcal{A}$, the automaton associated to the language
$C_{\mathcal{A}}$, a set of pairs of constrained edges in $\mathcal{A}$
d, depth of a layer
outputs: sol, a set of automata without inside incompatible edges

$edges\_sets \leftarrow$ separate incompatible edges at depth d into sets regarding $C_{\mathcal{A}}$
for $set \in edges\_sets$
    Create new automaton $\mathcal{A}' \leftarrow$ from set, descendants of set and ancestor of set
    Minimize $\mathcal{A}'$
    if $C_{\mathcal{A}'}$ *is empty*
        add $\mathcal{A}'$ in sol
    else
        add $Split\_automaton(\mathcal{A}', d+1)$ in sol
return sol

based on first order logic, it is relatively easy to model the hurricane system, design the chronicle, and set constraints. We compare our method with Problog as it provides the exact likelihood regarding the problem and the constraints, so it would be possible to estimate how close our approximate method is from exact computation.

Furthermore, we provide a comparison of inference times between our method, Problog, and an approach using MLNs [19]. MLNs have been quite often used in literature to represent uncertainty in CEP, particularly from the computer vision domain [8], [20], [21]. We do not provide a comparison of estimations between our method and MLN approach. Indeed, MLNs were not able to produce coherent results due to the sampling method used with MLNs and the size of the problem when converted into the MLN structure. In brief, computational issues rise when a model is design with circumscription which implies many loopy relationships between predicates making the MLN sampler inefficient. More details about this subject is addressed in [22].

For our experimentations, we used the hurricane model described above in Fig. 8 and Table 3b. We want to perform an inference on this model with the following chronicle as a query:

$$((H\ H) - [C])\ equals\ ((\textit{off}\ \textit{off}) - [on]) \tag{6}$$

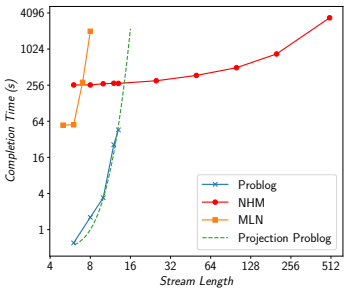
This chronicle describes a period, during which a hurricane strikes but the alarm is never raised. As observation, we state that for $t \in [2, 5]$ the alarm was never on, but meanwhile a hurricane happened, as formalised by this chronicle:

$$H\ during\ ((\textit{off}\ \textit{off}) - [on]) \tag{7}$$

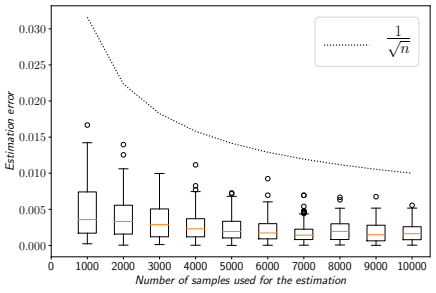
To constraint our model regarding this prior knowledge, we turn Eq. 7 into a set of sub-graphs as described in Sect. 4 by using *Choco* as constraint solver[4].

We set the number of samples to 10 000 per sub-graph and compared the inference time of our method with Problog, and a MLN-based approach. The results are shown in Fig. 9a. Due to a lack of memory, we could not produce results for Problog streams of

[4] `http://www.choco-solver.org/` [23]



(a) Inference time regarding stream length for our method, Problog, MLN.



(b) Estimation errors compared to the exact evaluation regarding the number of samples created.

Fig. 9: Inference time and estimation errors of our approach.

length above 13, but computation time was clearly exponential[5]. Even if our sampling method is slower to perform on really small problems, it appears clearly that problem dimension has less impact on the computation time of our method than Problog and MLN approach. It is noteworthy that the stream is in fact weakly constrained, so Problog might have performed better with more constraints. But it remains true that the maximum number of free events is quite low in comparison with our method. Moreover, we compare our estimation with the results provided by Problog on streams of length 13. Fig. 9b shows the results of the relative error of one hundred estimations when different amount of samples are used. As expected for a sampling approach, our error is decreasing in $O(\sqrt{n})$.

6 Discussion

Our results are promising and indicate that the approach scales for larger problems and long data streams. But it does not solve every problem that uncertainty may bring. Consequently, we present in this discussion a brief overview of the state of the art and the differences with our method.

Automata-based methods Those approaches are usually extensions of the SASE model or Cugola's work which use (N)DFA as pattern model. Given a stream, the recognition of a pattern corresponds to the termination of its associated automaton. The uncertainty computation depends on assumptions and goals; e.g., [24] calculates the marginal probability of a pattern by enumerating all possible recognitions and summing their probabilities, with events assumed independent. Other approaches ([1]) relax the independence assumption and use a MC to describe the dependencies between events.

[5] It was to be expected, as Problog computes all possible solutions, the set of which grows exponentially.

Usually, complex temporal constraints are not used or permitted with these methods, but some exceptions exist as [2], where the authors calculate the most probable explanation for activities using probabilities associated to the transitions of the automaton. In that case, the stochastic automaton allows specification of such temporal constraints. A similar approach from Fazzinga, Flesca, Furfaro, *et al.* [25] exists that uses a forward-backward algorithm to construct iteratively all the explanations into a tree structure where irrelevant explanations regarding the constraints are cut.

Automata-based methods provide a large spectrum of approaches but often tend to focus on representations of CE as sequences of LLEs with sometimes Kleene operator. Absence operator is a rare to be found. On uncertainty, three categories emerge: those assuming LLEs independence assumptions, those using first order Markovian assumptions and those using ad-hoc approaches of uncertainty (for instance [25] uses a bounded explorations of inconsistent explanations associate to a confidence approximation).

First-Order Logic (FOL) and PGMs Those methods usually rely on logic to describe models and rules, and on PGM to represent uncertainty. Two main graphical models are used to handle uncertainty: MLNs and Bayesian networks. Large amount of research has been produced around CEP and MLNs [3], [4] especially for human activity recognition. MLNs were introduced by [19] and use a FOL semantic to describe a Markov random field where a specific query on the model can be approximated using MCMC. The sampling relies on finding possible worlds with respect to the FOL problem. The FOL expressiveness made this model quite popular but, in practice, it has high computation costs on complex problems [22], [26].

Although less common, Bayesian approaches produced interesting results. They are generally used to describe the uncertainty on the relation between events in a defined complex event [27]. But some research [28] proposes solutions for dealing efficiently with uncertainty on simple events[6] and not just their relationship using a probability distribution on each event. However, these probability distributions should be given by an expert and the approach does not allow them to change over time. Moreover, the probability of events and the probability of the relation are just multiplied since they are supposed to be independent, but not all the system would satisfy this hypothesis. Anyway, this approach focused on a different uncertainty problem than ours, since it offers a representation of the confidence we have on CE modelling.

Another approach describes a CE system with Problog [29]. Problog does not rely on graphical models, but uses the knowledge from FOL domain to find all possible outcomes of a system described with a set of Horn clauses. all used Horn clauses might be associated with a probability that describes the chance that under the assumption the clause body is asserted, the head will be too. This approach was used with event calculus [3] and showed its efficiency. But, even if it provides the exact probability of the designed problem using Sentential Decision Diagram, it comes with expensive computation costs on large structures. An equivalent approach proposed by [7] used the MLN structure.

Grammars Grammars are an interesting way to represent CE. [6] uses stochastic context free grammars from [30] to describe complex event regarding a data stream. Each

[6] In particular, they propose an interesting first approach for the uncertainty on event attributes.

transition rule from the grammar is associated to a probability and the algorithm tries to explain the outcome data stream by looking at all the successions of rules matching it. The stream probability is the sum of the probabilities of each succession of rules.

This method has some drawbacks, as it does not manage attributes and could be overwhelmed when the number of possible explanations grows, but nonetheless the expressiveness of this approach remains really interesting.

Overview As we explained in the introduction, comparing these methods is not an easy task since each approach might focus on different aspects of uncertainty or CEP representation. But, we think our approach to be an interesting contribution as it proposes to efficiently estimate event uncertainty using long time relations to describe CEs in reasonable time compared to MLNs, that are though widely used. Furthermore, using CE as complementary prior knowledge has, to our knowledge, never been proposed in previous works of the community. Nonetheless we want to address some restrictions that should be pointed out. In our method, as we said in Sect. 2.1, we consider only systems with 0-ary events but many systems do not satisfy this condition. It is still possible to adapt systems with finite discrete attributes by defining a 0-ary event for each attribute combination. But it may lead on huge numbers of events and thus on massive transition matrices, hence storage issues. Furthermore, this simple approach is not convenient for attributes with continuous values.Another interest point is the number of matrices used to create the NHM. The analysis of really long data streams could lead to store a considerable amount of matrices, which could be easily reduced by an interleave process that calculate the NHM while producing the samples.

The high constraint expressiveness (cf. Sect. 4) is limited by the need to find all possible streams. This leads to pick small intervals where the chronicle should appear to avoid a combinatorial explosion. It would be interesting to have a more straight-forward way to find the corresponding automaton. As chronicles are expressed with a context-free grammar it might be possible to find incompatible sub-streams using push-down automata instead of DFA. Incidentally, it may let us express really strong constraints along the full stream.

7 Conclusion and Perspectives

In this paper, we introduce a method performing CE recognition and analysis under uncertainty detection of LLE more scalable on long data streams, with high-level knowledge description. The approach is in two steps: first, we estimate the probability of a chronicle recognition on an uncertain data stream. The observed data stream is used as a constraint on the system allowing a NHM to describe the system. This NHM is then used to perform a MCMC sampling, where each sample is analysed through a CE recognition system, thus leading to the estimation the likelihood of a chronicle. Second, we use chronicles as a prior knowledge on the system. Chronicles are transformed using an automaton representation into constraint sub-graphs for the NHM, which reduce substantially the needed sample amount, since a sub-graph embeds many possible sub-streams according to the chronicle used.

Our results showed that our techniques performed good probability estimations, scalable on long data streams to analyse regarding our criteria, whereas other approaches

tend to be intractable or inaccurate. In future work, we plan to investigate three improvement axes. First, we plan to avoid sampling by using model checking approaches, thus reducing computation time, even if such an approach would reduce the expressiveness of query (e.g. it would not be possible to express a query that looks for a certain number of CE recognitions per streams.). Second, the current structure of our method might be easily adapt to a Map/Reduce approach and might have an important impact on computation time. Third, we aim at representing our constraint automata directly from the chronicle representation without solving a CSP. Regarding some preliminary experimentations, this method, combined with the model checking approach, might provide an efficient solution, fast enough for online computation even with high level constraints.

References

[1] Y. Wang, K. Cao, and X. Zhang, "Complex event processing over distributed probabilistic event streams", *Computers & Mathematics with Applications*, vol. 66, no. 10, pp. 1808–1821, 2013.

[2] M. Albanese *et al.*, "Finding "Unexplained" Activities in Video.", in *IJCAI*, 2011, pp. 1628–1634.

[3] A. Skarlatidis *et al.*, "Probabilistic event calculus for event recognition", *ACM Transactions on Computational Logic (TOCL)*, vol. 16, no. 2, p. 11, 2015.

[4] F. Liu, D. Deng, and P. Li, "Dynamic Context-Aware Event Recognition Based on Markov Logic Networks", *Sensors*, vol. 17, no. 3, p. 491, 2017.

[5] A. Skarlatidis *et al.*, "A probabilistic logic programming event calculus", *Theory and Practice of Logic Programming*, vol. 15, no. 02, pp. 213–245, 2015.

[6] Y. A. Ivanov and A. F. Bobick, "Recognition of visual activities and interactions by stochastic parsing", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 852–872, 2000.

[7] V. Morariu, L. S. Davis, *et al.*, "Multi-agent event recognition in structured scenarios", in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, IEEE, 2011, pp. 3289–3296.

[8] Y. C. Song *et al.*, "A Markov Logic Framework for Recognizing Complex Events from Multimodal Data", in *Proc. of the 15th ACM on International Conference on Multimodal Interaction*, ser. ICMI '13, ACM, 2013, pp. 141–148.

[9] A. Piel, "Reconnaissance de comportements complexes par traitement en ligne de flux d'evenements", PhD thesis, U. Paris 13, 2014.

[10] P. Carle, C. Choppy, and R. Kervarc, "Behaviour Recognition Using Chronicles", in *5th International Symposium on Theoretical Aspects of Software Engineering (TASE)*, IEEE, 2011, pp. 100–107.

[11] Y. Diao, N. Immerman, and D. Gyllstrom, "Sase+: An agile language for Kleene closure over event streams", *UMass Technical Report*, 2007.

[12] G. Cugola and A. Margara, "Complex event processing with T-REX", *J. of Systems and Software*, vol. 85, no. 8, pp. 1709–1728, 2012.

[13] C. Dousson and P. Le Maigat, "Chronicle Recognition Improvement Using Temporal Focusing and Hierarchization.", in *IJCAI*, vol. 7, 2007, pp. 324–329.

[14] A. Artikis, M. Sergot, and G. Paliouras, "Run-time composite event recognition", in *Proc. of the 6th ACM International Conference on Distributed Event-Based Systems*, ACM, 2012, pp. 69–80.
[15] J. F. Allen, "Maintaining knowledge about temporal intervals", *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.
[16] F. Pachet, P. Roy, and G. Barbieri, "Finite-length Markov processes with constraints", in *IJCAI*, 2011.
[17] J. Bubenzer, "Minimization of Acyclic DFAs.", in *Stringology*, 2011, pp. 132–146.
[18] A. Dries *et al.*, "ProbLog2: Probabilistic logic programming", in *Joint European Conf. on Machine Learning and Knowledge Discovery in Databases*, Springer, 2015, pp. 312–315.
[19] M. Richardson and P. Domingos, "Markov logic networks", *Machine Learning*, vol. 62, no. 1-2, pp. 107–136, 2006.
[20] S. D. Tran and L. S. Davis, "Event modeling and recognition using markov logic networks", in *Computer Vision–ECCV 2008*, Springer, 2008, pp. 610–623.
[21] A. Skarlatidis *et al.*, "Probabilistic event calculus based on markov logic networks", in *Proc. of Rule-Based Modeling and Computing on the Semantic Web*, ser. LNCS, vol. 7018, Springer, 2011, pp. 155–170.
[22] R. Rincé, R. Kervarc, and P. Leray, "On the Use of WalkSAT Based Algorithms for MLN Inference in Some Realistic Applications", in *30th International Conf. on Industrial, Engineering, Other Applications of Applied Intelligent Systems*, 2017, pp. 121–131.
[23] C. Prud'homme, J.-G. Fages, and X. Lorca, *Choco Documentation*, 2017.
[24] H. Kawashima, H. Kitagawa, and X. Li, "Complex Event Processing over Uncertain Data Streams", in *2010 International Conf. on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, 2010, pp. 521–526.
[25] B. Fazzinga *et al.*, "Efficiently interpreting traces of low level events in business process logs", *Information Systems*, vol. 73, pp. 1–24, 2018.
[26] E. Alevizos *et al.*, "Probabilistic Complex Event Recognition: A Survey", *ACM Computing Surveys*, vol. 50, no. 5, pp. 1–31, 2017.
[27] X. Wang and Q. Ji, "Context augmented Dynamic Bayesian Networks for event recognition", *Pattern Recognition Letters*, vol. 43, pp. 62–70, 2014.
[28] G. Cugola *et al.*, "Introducing uncertainty in complex event processing: Model, implementation, and validation", *Computing*, vol. 97, no. 2, pp. 103–144, 2015.
[29] D. Fierens *et al.*, "Inference in probabilistic logic programs using weighted CNF's", *Theory and Practice of Logic Programming*, vol. 15, no. 03, pp. 258–401, 2012.
[30] A. Stolcke, "An efficient probabilistic context-free parsing algorithm that computes prefix probabilities", *Computational linguistics*, vol. 21, no. 2, pp. 165–201, 1995.