Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)
Applications and Systems

WOLED: A Tool for Online Learning Weighted Answer Set Rules for Temporal Reasoning Under Uncertainty

Nikos Katzouris[1] , **Alexander Artikis**[2,1]
[1]National Center for Scientific Research "Demokritos", Greece
[2]University of Pireaus, Greece
{nkatz, a.artikis}@iit.demokritos.gr

Abstract

Complex Event Recognition (CER) systems detect event occurrences in streaming time-stamped input using predefined event patterns. Logic-based approaches are of special interest in CER, since, via Statistical Relational AI, they combine uncertainty-resilient reasoning with time and change, with machine learning, thus alleviating the cost of manual event pattern authoring. We present WOLED, a system based on Answer Set Programming (ASP), capable of probabilistic reasoning with complex event patterns in the form of weighted rules in the Event Calculus, whose structure and weights are learnt online. We compare our ASP-based implementation with a Markov Logic-based one and with a crisp version of the algorithm that learns unweighted rules, on CER datasets for activity recognition, maritime surveillance and fleet management. Our results demonstrate the superiority of our novel implementation, both in terms of efficiency and predictive performance.

1 Introduction

Complex Event Recognition (CER) systems (Cugola and Margara 2012) detect occurrences of *complex events* (CEs) in streaming input, defined as spatio-temporal combinations of *simple events* (e.g. sensor data), using a set of CE patterns. Since such patterns are not always known beforehand and existing ones often need to be updated, machine learning algorithms for the automatic construction/revision of CE patterns are highly useful. Such algorithms should ideally operate in an online fashion, by using the current CE pattern set for inference (CER) in the incoming data stream, and the labeled portions of the stream for updating the CE pattern set. Moreover, such algorithms should be resilient to noise & uncertainty, which are ubiquitous in temporal data streams (Alevizos et al. 2017), and support reasoning with existing domain knowledge, while taking into account commonsense phenomena (Mueller 2014), which often characterize dynamic application domains, such as CER.

Logic-based CER systems (Artikis et al. 2012) stand up to the aforementioned challenges. They combine reasoning under uncertainty with machine learning, via Statistical Relational AI techniques (De Raedt et al. 2016), while they are capable of reasoning with time and change, and incorporating commonsense principles via action formalisms, such as the Event Calculus (Artikis, Sergot, and Paliouras 2015).

A number of online learning algorithms, capable of temporal reasoning with a set of CE patterns, while continuously updating these patterns in the face of new data, have already been proposed (Katzouris, Artikis, and Paliouras 2016; Michelioudakis et al. 2016; Katzouris et al. 2018). We advance the state of the art by proposing WOLED (Online Learning of Weighted Event Definitions), an algorithm that learns CE patterns in the form of weighted rules in the Event Calculus. In contrast to a predecessor algorithm based on Markov Logic Networks (MLNs) (Katzouris et al. 2018), WOLED is based entirely on Answer Set Programming (ASP), which allows to take advantage of the grounding, solving, optimization and uncertainty modeling abilities of modern answer set solvers, while employing structure learning techniques from non-monotonic Inductive Logic Programming (ILP) (De Raedt 2008), which are easily implemented in ASP, towards more robust learning.

We compare WOLED's ASP-based implementation to an MLN-based one, and to crisp version of the algorithm that learns unweighted rules, on three CE datasets for *activity recognition*, *maritime surveillance* and *vehicle fleet management*. Our results demonstrate the superiority of our novel implementation, both in terms of efficiency and predictive performance.

2 Related Work

Event Calculus-based CER (Artikis, Sergot, and Paliouras 2015) was combined with MLNs in (Skarlatidis et al. 2015), in order to deal with the uncertainty of CER applications. An inherent limitation of this approach is the fact that the non-monotonic semantics of the Event Calculus is incompatible with the open-world semantics of MLNs. Therefore, performing inference with Event Calculus-based MLN theories calls for extra, costly operations, such as computing the completion of a theory (Mueller 2014), in order to endow the first-order logic representations on which MLNs rely with a non-monotonic semantics. We bridge this gap via translating probabilistic inference with MLNs into an optimization task in ASP, which naturally supports non-monotonic and commonsense reasoning. This also allows to delegate probabilistic temporal reasoning and machine learning tasks to sophisticated, off-the-shelf answer set solvers.

Translating MLN inference in ASP has been put forth in (Lee and Wang 2016; Lee, Talsania, and Wang 2017). This

(a)

Predicate	**Meaning**
happensAt(E, T)	Event E occurs at time T.
initiatedAt(F, T)	At time T, a period of time for which fluent F holds is initiated.
terminatedAt(F, T)	At time T, a period of time for which fluent F holds is terminated.
holdsAt(F, T)	Fluent F holds at time T.

(b)

The axioms of the Event Calculus

holdsAt$(F, T + 1) \leftarrow$ initiatedAt(F, T) (1)

holdsAt$(F, T + 1) \leftarrow$ holdsAt(F, T), not terminatedAt(F, T) (2)

(c)

Observations I_1 at time 1:

happensAt$(walk(id_1), 1)$
happensAt$(walk(id_2), 1)$
$coords(id_1, 201, 454, 1)$
$coords(id_2, 230, 440, 1)$
$direction(id_1, 270, 1)$
$direction(id_2, 270, 1)$

Target CE instances at time 1:

holdsAt$(move(id_1, id_2), 2)$
holdsAt$(move(id_2, id_1), 2)$

(d)

Weighted CE patterns:

1.234 initiatedAt$(move(X, Y), T) \leftarrow$
happensAt$(walk(X), T)$,
happensAt$(walk(Y), T)$,
$close(X, Y, 25, T)$,
$orientation(X, Y, 45, T)$

0.923 terminatedAt$(move(X, Y), T) \leftarrow$
happensAt$(inactive(X), T)$,
not $close(X, Y, 30, T)$

Table 1: **(a), (b)** The basic predicates and the EC axioms. **(c)** Example CAVIAR data. For example, at time point 1 person with id_1 is *walking*, her (X, Y) coordinates are $(201, 454)$ and her direction is $270°$. The query atoms for time point 1 ask whether persons id_1 and id_2 are moving together at the next time point. **(d)** An example of two domain-specific axioms in the EC. E.g. the first rule dictates that *moving together* between two persons X and Y is initiated at time T if both X and Y are walking at time T, their euclidean distance is less than 25 pixel positions and their difference in direction is less than $45°$. The second rule dictates that *moving together* between X and Y is terminated at time T if one of them is standing still at time T and their euclidean distance at T is greater that 30.

line of work is mostly concerned with theoretical aspects of the translation, limiting applications to simple, proof-of-concept examples. Although we do rely on the theoretical foundation of this work, we take a more application-oriented stand-point and investigate the usefulness of these ideas in challenging domains, such as CER. Another important difference from previous work on combining MLNs with ASP is that while the latter does not touch upon machine learning, we propose a methodology for learning both the structure and the weights of rules representing CE patterns, in an online fashion, using ASP tools.

Regarding machine learning, a number of algorithms in the non-monotonic branch of Inductive Logic Programming (ILP), such as XHAIL (Ray 2009), TAL (Athakravi et al. 2013) and ILASP (Law, Russo, and Broda 2018) are capable of learning Event Calculus theories. However, these algorithms are batch learners, they are thus poor matches to the online nature of CER applications. Moreover, they learn crisp logical theories, thus their ability to cope with noise and uncertainty is limited. Existing online learning algorithms (Katzouris, Artikis, and Paliouras 2016; Michelioudakis et al. 2016) rely on MLNs, so they suffer from the same limitations discussed earlier in this section, while a recent online learner based on probabilistic theory revision (Guimarães, Paes, and Zaverucha 2019) is limited to Horn logic and cannot handle Event Calculus reasoning.

3 Background

We assume a first-order language where atoms, literals (possibly negated atoms), rules and logic programs are defined as in (Gebser et al. 2012) and not denotes negation as failure. Rules, atoms, literals and programs are ground if they contain no variables. Rules are denoted by $\alpha \leftarrow \delta_1, \ldots, \delta_n$, where α is an atom and $\delta_1, \ldots, \delta_n$ a conjunction of literals. An interpretation I is a set of true ground atoms. I satisfies a ground literal a (resp. not a) iff $a \in I$ (resp. $a \notin I$) and it satisfies a ground rule iff it satisfies the head, or does not satisfy the body. I is a minimal (Herbrand) model of a logic program Π iff it satisfies every ground rule in Π and none of its strict subsets has this property. I is an answer set of Π iff it is a minimal model of the program that results from the ground instances of Π, after removing all rules with a negated literal not satisfied by I, and all negative literals from the remaining rules. A choice rule is an expression of the form $\{\alpha\} \leftarrow \delta_1, \ldots, \delta_n$, which is syntactic sugar for $\alpha \leftarrow \delta_1, \ldots, \delta_n$, not not α, with the intuitive meaning that whenever the body $\delta_1, \ldots, \delta_n$ is satisfied by an answer set I of a program that includes the choice rule, instances of the head α are arbitrarily included in I (satisfied) as well.

A weak constraint is an expression of the form $: \sim \delta_1, \ldots, \delta_n.[w]$, where δ_i's are literals and w is an integer. The intuitive meaning of a weak constraint c is that the satisfaction of the conjunction $\delta_1, \ldots, \delta_n$ by an answer set I of a program that includes c incurs a cost of w for I. Inclusion of weak constraints in a program triggers an optimization process that yields answer sets of minimum cost. We refer to (Gebser et al. 2012) for a formal account of choice rules and weak constraints' semantics. In what follows we use the Clingo[1] syntax for representing these constructs.

The Event Calculus is a temporal logic for reasoning about events and their effects. Its ontology comprises time points (integers), fluents, i.e. properties which have certain values in time, and events, i.e. occurrences in time that may affect fluents and alter their value. Its axioms incorporate the commonsense law of inertia, according to which fluents persist over time, unless they are affected by an event. Its basic predicates and axioms are presented in Table 1(a), (b). Axiom (1) states that a fluent F holds at time T if it has been initiated at the previous time point, while Axiom (2) states that F continues to hold unless it is terminated. Definitions of initiatedAt/2 and terminatedAt/2 predicates are provided in a application-specific manner.

Using the Event Calculus in a CER context allows to reason with CEs that have duration in time and are subject to commonsense phenomena, via associating CEs to fluents. In this case, a set of CE patterns is a set of conditions that

[1] https://potassco.org/

initiate/terminate a target CE, i.e., a set of initiatedAt/2 and terminatedAt/2 rules.

As an example we use the task of activity recognition, as defined in the CAVIAR project[2]. The CAVIAR dataset consists of videos of a public space, where actors perform some activities. These videos have been manually annotated by the CAVIAR team to provide the ground truth for two types of activity. The first type, corresponding to simple events, consists of knowledge about a person's activities at a certain video frame/time point (e.g. *walking, standing still* and so on). The second type, corresponding to CEs/fluents, consists of activities that involve more than one person, for instance two people *moving together, meeting each other* and so on. The aim is to detect CEs as of combinations of simple events and additional domain knowledge, such as a person's position and direction.

Table 1(c) presents an example of CAVIAR data, consisting of *observations* for a particular time point, in the form of an interpretation I_1. A stream of interpretations is matched against a set of CE patterns (initiation/termination rules – see Table 1(d)), to infer the truth values of CE instances in time, using the Event Calculus axioms as a reasoning engine. We henceforth call the atoms corresponding to CE instances whose truth values are to be inferred/predicted, *target CE instances*. Table 2(c) presents the target CE instances corresponding to the observations in I_1. Note that at time t the corresponding target CE instances refer to $t+1$, in accordance to the Event Calculus axioms, which infer the truth value of a CE instance at a time point, base on what happens at the previous time point.

In WOLED, the CE patterns included in a logic program Π are associated with real-valued weights, defining a probability distribution over answer sets of Π. Similarly to Markov Logic, where a possible world may satisfy a subset of the formulae in an MLN, and the weights of the formulae in a unique, maximal such subset determine the probability of the possible world, an answer set of a program with weighted rules may satisfy subsets of these rules, and these rules' weights determine the answer set's probability. Based on this observation, (Lee and Wang 2016) propose to assign probabilities to answer sets of a program Π with weighted rules as follows: For each interpretation I, first find the maximal subset R_I of the weighted rules in Π that are satisfied by I. Then, assign to I a weight $W_\Pi(I)$ proportional to the sum of weights of the rules in R_I, if I is an answer set of R_I, else assign zero weight. Finally, define a probability distribution over answer sets of Π by normalizing these weights.

Formally, let w_r be the weight of rule r and $\mathsf{ans}(\Pi)$ the set of all interpretations I which are answer sets of R_I and which, moreover, satisfy all hard-constrained rules in Π (rules without weights). Then

$$W_\Pi(I) = \begin{cases} \exp\left(\sum_{r \in R_I} w_r\right) & \text{if } I \in \mathsf{ans}(\Pi) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

[2]http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

$$P_\Pi(I) = \frac{W_\Pi(I)}{\sum_{J \in \mathsf{ans}(\Pi)} W_\Pi(J)} \quad (2)$$

4 Structure & Weight Learning in ASP

The task that WOLED addresses is to online learn the structure and weights of CE patterns, while using their current version at each point in time to perform CER in the streaming input. We adopt a standard online learning approach consisting of the following steps: at time t the learner maintains a theory H_t (weighted CE pattern set, as in Table 1(c)), has access to some static background knowledge (e.g. the axioms of the Event Calculus – Table 1(a)) and receives an interpretation I_t, consisting of a data mini-batch (as in Table 1(b)). Then (i) the learner performs inference (CER) with $B \cup H_t$ on I_t (B is the background knowledge) and generates a "predicted state", consisting of inferred holdsAt/2 instances of the target predicate. Via closed-world assumption, all such instances not present in the predicted state are false; (ii) if available, the true state, consisting of the actual truth values of the predicted atoms is revealed; (iii) the learner identifies erroneous predictions via comparing the predicted state to the true one, and uses these mistakes to update the structure and the weights of the CE patterns in H_t, yielding a new theory H_{t+1}.

We next discuss each of these steps and their implementation using ASP tools.

4.1 Generating the Inferred State

To make predictions with the weighted CE patterns in the incoming data interpretations, WOLED uses MAP (Maximum A Posteriori) probabilistic inference[3], which amounts to computing a most probable answer set $\mathcal{A}$ of $\Pi = B \cup H_t \cup I_t$. From Equations (1), (2) it follows that

$$\mathcal{A} = \arg\max_{I \in \mathsf{ans}(\Pi)} P_\Pi(I) = \arg\max_{I \in \mathsf{ans}(\Pi)} W_\Pi(I) = \arg\max_{I \in \mathsf{ans}(\Pi)} \sum_{r \in R_I} w_r \quad (3)$$

that is, a most probable answer set is one that maximizes the sum of weights of satisfied rules, similarly to the MLN case, for possible worlds. This is a weighted MaxSat problem that may be delegated to an answer set solver using built-in optimization tools. Since answer set solvers only optimize integer-valued objective functions, a first step is to convert the real-valued CE pattern weights to integers. We do so by scaling the weights, via multiplying them by a positive factor, while preserving their relative differences, and rounding the result to the closest integer.

Note that as it may be seen from Equation (3), weight scaling by a positive factor does not alter the set of most

[3]Marginal inference, i.e. computing the probability of each target CE instance is also possible, but it is computationally expensive since it requires a full enumeration of a program's answer sets, of utilizing techniques for sampling from such answer sets. We are not concerned with marginal inference in this work.

(a) Axioms of the Event Calculus & other BK:

$\mathsf{holdsAt}(E, T+1) \leftarrow \mathsf{initiatedAt}(E, T), targetCE(E).$

$\mathsf{holdsAt}(E, T+1) \leftarrow \mathsf{holdsAt}(E, T), \mathsf{not\ terminatedAt}(E, T), targetCE(E).$

$targetCE(a).\ time(1..10).$

(b) Current CE pattern set H_t with weights converted to integers:

$11\ \mathsf{initiatedAt}(a, T) \leftarrow \mathsf{happensAt}(b, T).$ $(rule_1)$

$13\ \mathsf{terminatedAt}(a, T) \leftarrow \mathsf{happensAt}(c, T).$ $(rule_2)$

$-2\ \mathsf{initiatedAt}(a, T) \leftarrow \mathsf{happensAt}(d, T).$ $(rule_3)$

(c) Current data interpretation I_t:

$\mathsf{happensAt}(b, 2).\ \mathsf{happensAt}(c, 5).\ \mathsf{happensAt}(d, 8).$

(e) Inferred state with crisp logical inference:

$\mathsf{holdsAt}(a, 3).\ \mathsf{holdsAt}(a, 4).\ \mathsf{holdsAt}(a, 5).\ \mathsf{holdsAt}(a, 9).\ \mathsf{holdsAt}(a, 10).$

(f) Inferred state with probabilistic MAP inference:

$\mathsf{holdsAt}(a, 3).\ \mathsf{holdsAt}(a, 4).\ \mathsf{holdsAt}(a, 5).$

$\mathsf{satisfied}(\mathsf{vars}(5), rule_2).\ \mathsf{satisfied}(\mathsf{vars}(2), rule_1).$

(d) Syntactically transformed program $T(H_t)$ for MAP inference:

$\mathsf{initiatedAt}(a, T) \leftarrow \mathsf{satisfied}(\mathsf{vars}(T), rule_1).$

$\{\mathsf{satisfied}(\mathsf{vars}(T), rule_1)\} \leftarrow \mathsf{happensAt}(b, T).$

$:\sim \mathsf{satisfied}(\mathsf{vars}(T), rule_1).\ [-11, \mathsf{vars}(T), rule_1]$

$\mathsf{initiatedAt}(a, T) \leftarrow \mathsf{satisfied}(\mathsf{vars}(T), rule_2).$

$\{\mathsf{satisfied}(\mathsf{vars}(T), rule_2)\} \leftarrow \mathsf{happensAt}(c, T).$

$:\sim \mathsf{satisfied}(\mathsf{vars}(T), rule_2).\ [-13, \mathsf{vars}(T), rule_2]$

$\mathsf{initiatedAt}(a, T) \leftarrow \mathsf{satisfied}(\mathsf{vars}(T), rule_3).$

$\{\mathsf{satisfied}(\mathsf{vars}(T), rule_3)\} \leftarrow \mathsf{happensAt}(d, T).$

$:\sim \mathsf{satisfied}(\mathsf{vars}(T), rule_3).\ [2, \mathsf{vars}(T), rule_3]$

Table 2: ASP-based MAP inference with the Event Calculus.

Algorithm 1 $\mathsf{MAPInference}(B, H_t, I_t)$

Input: background knowledge B; the current CE pattern set H_t; the input interpretation I_t.

Output: Target CE instances included in the most probable answer set of $B \cup T(H_t) \cup I_t$.

1: $T(H_t) := \emptyset$
2: **for each** CE pattern $r_i = \alpha \leftarrow \delta_1, \ldots, \delta_n$ in H_t with integer weight w_i **do**
3: **let** $\mathsf{vars}(\alpha)$ be a term wrapping the variables of α.
4: Add to $T(H_t)$ the following rules:
5: $\alpha \leftarrow \mathsf{satisfied}(\mathsf{vars}(\alpha), i).$
6: $\{\mathsf{satisfied}(\mathsf{vars}(\alpha), i)\} \leftarrow \delta_1, \ldots, \delta_n.$
7: $:\sim \mathsf{satisfied}(\mathsf{vars}(\alpha), i).\ [-w_i, \mathsf{vars}(\alpha), i]$
8: Find an optimal answer set $\mathcal{A}_{opt}$ of $B \cup T(H_t) \cup I_t$.
9: **return** the target CE instances in $\mathcal{A}_{opt}$.

probable answer sets. Therefore, the inference result remains unaffected, provided that rounding the weights to integer values preserves their relative differences. To ensure the latter, we set the scaling factor to K/d_{min}, where $d_{min} = min_{i \neq j}|w_i - w_j|$ is the smallest distance between any pair of weights and K is a large positive constant, which reduces precision loss when rounding the scaled weights to integer values.

The MAP inference/weighted MaxSat computation is realized via a standard generate-and-test ASP approach, presented in Algorithm 1, whose input is the background knowledge B, the current CE pattern set H_t and the current interpretation I_t. First, H_t is transformed into a new program, $T(H_t)$, as follows: each CE pattern r_i in H_t of the form $r_i = head_i \leftarrow body_i$ is "decomposed", so as to associate $head_i$ with a fresh predicate, $\mathsf{satisfied}/2$, wrapping $head_i$'s variables and its unique id, i (line 5, Algorithm 1). The choice rule in line 6, the "generate" part of the process, generates instances of $\mathsf{satisfied}/2$ that correspond to groundings of $body_i$. The weak constraint in line 7, the "test" part of the process, decides which of the generated $\mathsf{satisfied}/2$ instances will be included in an answer set, indicating groundings of the initial CE pattern r_i, that will be true in the inferred state.

As it may be seen from line 7, the violation of a weak constraint by an answer set $\mathcal{A}$ of $\Pi = B \cup T(H_t) \cup I_t$, i.e. the satisfaction of a ground instance of r_i by $\mathcal{A}$, incurs a cost of $-w_i$ on $\mathcal{A}$, where w_i is r_i's integer-valued weight. The optimization process triggered by the inclusion of these weak constraints in a program generates answer sets of minimum cost. During the cost minimization process, costs of $-w_i$ are actually rewards for rules with a positive w_i, whose satisfaction by an answer set, via the violation of the corresponding weak constraint, reduces the answer set's total cost. The situation is reversed for rules with a negative weight, whose corresponding weak constraint is associated with a positive cost.

Obtaining the inferred state amounts to "reading-off" target CE instances from an optimal (minimum-cost) answer set of the program $B \cup T(H_t) \cup I_t$.

Example 1. *We illustrate the inference process via the example in Table 2. In (a) the Event Calculus axioms are presented, with an extra predicate, $targetCE/1$, indicating the target CE whose occurrences we wish to detect and which is subject to the effects of inertia; (b) presents a CE pattern set H_t, where we assume that the actual real-valued weights of the patterns have been converted to integers, as described earlier; (c) presents the current data interpretation I_t; (d) presents the program $T(H_t)$ obtained from H_t, via the transformation in Algorithm 1, to allow for MAP inference; (e) presents the inferred state obtained with crisp logical inference, i.e. the target CE instances included in the unique answer set of the program $BK \cup H_t \cup I_t$, where the CE patterns' weights have been disregarded. Note that the occurrence of $\mathsf{happensAt}(b, 2) \in I_t$ initiates the target CE a via $rule_1 \in H_t$, so a holds at the next time point, 3, and it also holds at time points 4 & 5 via inertia. Then, the occurrence of $\mathsf{happensAt}(c, 5) \in I_t$ terminates a, via $rule_2 \in H_t$, so a does not holds at times 6,7,8, while the occurrence of $\mathsf{happensAt}(d, 8) \in I_t$ re-initiates a, via $rule_3 \in H_t$, so a holds at times 9 & 10. Finally, (f) presents the MAP-inferred state, i.e. the target predicate instances included in an optimal (minimum-cost) answer set of the program $BK \cup T(H_t) \cup I_t$ (for illustrative purposes the $\mathsf{satisfied}/2$ instances in the optimal answer set are also presented). Note that the set of target CE inferences is reduced,*

as compared to the crisp case, since the negative-weight, $rule_3 \in H_t$ is not satisfied by the optimal answer set. The satisfied$/2$ *instances in the MAP-inferred state correspond to the ground atoms* terminatedAt$(a, 5)$, initiatedAt$(a, 2)$*, which, along with inertia, are responsible for the target CE inferences.*

4.2 Weight Learning

Once the learner makes a prediction on the incoming interpretation I_t and generates the inferred state, the true state is revealed, if available (i.e., if I_t is labeled), and the CE patterns' weights are updated by comparing their true groundings in the inferred and the true state. For a target CE α and an initiatedAt$/2$ (resp. terminatedAt$/2$) CE pattern r_i, a true grounding, either in the inferred, or in the true state, is a grounding of r_i at time t, such that holdsAt$(\alpha, t+1)$ is true (resp. false). CE patterns that contribute towards correct predictions (target CE inferences) are promoted, while those that make erroneous predictions are down-weighted.

As in (Katzouris et al. 2018), we use the AdaGrad algorithm (Duchi, Hazan, and Singer 2011) for weight updates, a version of Gradient Descent that dynamically adapts the learning rate, i.e. the magnitude of weight promotion/demotion, for each CE pattern individually, by taking into account the pattern's performance on the past data. AdaGrad updates a weight vector, whose coordinates correspond to a set of features (the CE patterns in our case), based on the subgradient of a convex loss function of these features. Our loss function is a simple variant of the hinge loss for structured prediction, originally used in (Huynh and Mooney 2011) for MLNs, whose subgradient is the vector with Δg_i in its i-th coordinate, denoting the difference in the i-th CE pattern's true groundings in the true and the inferred state respectively. The weight update rule for the i-th CE pattern is then:

$$w_i^{t+1} = sign(w_i^t - \frac{\eta}{C_i^t}\Delta g_i^t)\ max\{0, |w_i^t - \frac{\eta}{C_t^i}\Delta g_i^t| - \lambda\frac{\eta}{C_t^i}\}$$

where $t/t+1$-superscripts in terms denote respectively the previous and the updated values, η is a learning rate parameter, λ is a regularization parameter and $C_i^t = \delta + \sqrt{\sum_{j=1}^{t}(\Delta g_i^j)^2}$ is a term that expresses the CE pattern's quality so far, as reflected by the accumulated sum of Δg_i's, amounting to its past mistakes (plus a $\delta \geq 0$ to avoid division by zero in η/C_i^t). The C_i^t term is the adaptive factor that assigns a different learning rate to each CE pattern, since the magnitude of a weight update via the term $|w_i^t - \frac{\eta}{C_t^i}\Delta g_i^t|$ is affected by the CE pattern's previous history, in addition to its current mistakes, as expressed by Δg_i^t. The regularization term in Equation (1), $\lambda\frac{\eta}{C_t^i}$, is the amount by which the i-th CE pattern's weight is discounted when $\Delta g_i^t = 0$. This is to eventually push to zero the weights of irrelevant rules, which have very few, or even no groundings in the data.

4.3 Updating CE patterns' Structure

Similarly to OLED (Katzouris, Artikis, and Paliouras 2016), WOLED learns CE patterns via a classical in ILP, hill-climbing search process, generating a *bottom rule* (De Raedt 2008) $\perp_\alpha$ from a CE instance α and then searching for a high-quality CE pattern into the *subsumption lattice* defined by $\perp_\alpha$. It does so by progressively specializing an initially empty-bodied rule with the addition of one literal at a time from $\perp_\alpha$. To make this process online, the data in the incoming interpretations are used once, to evaluate a CE pattern and its current specializations. A Hoeffding test (Domingos and Hulten 2000) allows to identify, with high probability, the best specialization from a small subset of the input interpretations. Once the test succeeds, the parent rule is replaced by its best specialization and the process continues for as long as new specializations improve the current rule's performance. New bottom rules are generated over time from "missed" CE instances (not entailed by none of the existing CE patterns). Each such bottom rule instantiates a new subsumption lattice, which is searched for new CE patterns.

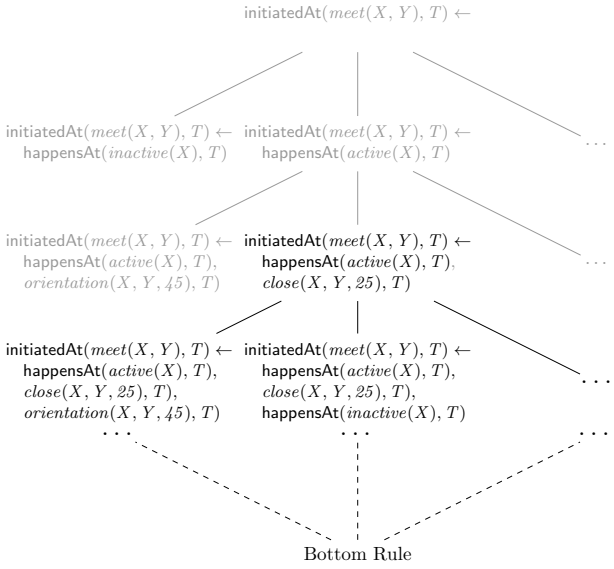


Figure 1: A subsumtion lattice.

In particular, at each point in time WOLED evaluates a parent rule and its specializations on incoming data, via an information gain scoring function, assessing the cumulative merit of a specialization over the parent rule, across the portion of the stream seen so far:

$$G(r, r') = P_r \cdot (log\frac{P_r}{P_r + N_r} - log\frac{P_{r'}}{P_{r'} + N_{r'}})$$

where r' is r's parent rule and for each rule r, P_r (resp. N_r) denotes the sum of true (resp. false) groundings of r in the MAP-inferred states generated so far. The information gain function is normalized in $[0, 1]$ by taking 0 as the minimum (as we are interested in positive gain only) and dividing a G-value by its maximum, $G_{max}(r, r') = P_{r'} \cdot (-log\frac{P_{r'}}{P_{r'}+N_{r'}})$. When the range of G is $[0, 1]$, a Hoeffding test succeeds, allowing to select r_1 as the best of a parent rule r's specializations, when $G(r_1, r) - G(r_2, r) > \epsilon = \sqrt{\frac{log 1/\delta}{2N}}$, where r_1, r_2 are respectively r's best and second-best specializations, δ is a confidence parameter and N is the number of

Algorithm 2 $\mathsf{LearnNewCEPatterns}(B, M, H_t, I_t, I_t^{\mathsf{MAP}}, I_t^{\mathsf{true}})$
Input: background knowledge B; mode declarations M, the current CE pattern set H_t; the current data interpretation I_t; the MAP-inferred state I_t^{MAP}; the true state I_t^{true}
Output: A set H_{new} of new CE patterns.

1: $\mathit{\Pi} := \emptyset$, $H_{new} := \emptyset$, $H_\perp := \emptyset$, $T(H_\perp) := \emptyset$
2: $Mistakes := I_t^{\mathsf{true}} \setminus I_t^{\mathsf{MAP}}$.
3: **for each** $m \in Mistakes$ **do**
4: $\quad H_\perp \leftarrow \mathsf{generateBottomRule}(m, I_t, M)$
5: $H_\perp \leftarrow \mathsf{compressBottomRules}(H_\perp)$
6: **for each** bottom rule $r_i = \alpha_i \leftarrow \delta_i^1, \ldots, \delta_i^n$ in $H_\perp$ **do**
7: $\quad$ Add to $T(H_\perp)$ the following rules:
$\alpha_i \leftarrow \mathsf{use}(i, 0), \mathsf{try}(i, 1, \mathsf{v}(\delta_i^1)), \ldots, \mathsf{try}(i, n, \mathsf{v}(\delta_i^n))$.
$\mathsf{try}(i, 1, \mathsf{v}(\delta_i^1)) \leftarrow \mathsf{use}(i, 1), \delta_i^1$.
$\mathsf{try}(i, 1, \mathsf{v}(\delta_i^1)) \leftarrow \mathsf{not}\ \mathsf{use}(i, 1)$.
$\ldots$
$\mathsf{try}(i, n, \mathsf{v}(\delta_i^n)) \leftarrow \mathsf{use}(i, n), \delta_i^n$.
$\mathsf{try}(i, n, \mathsf{v}(\delta_i^n)) \leftarrow \mathsf{not}\ \mathsf{use}(i, n)$.
8: $\Pi \leftarrow B \cup I_t \cup T(H_t) \cup T(H_\perp)$,
where $T(H_t)$ is the MAP inference-related transformation of Algorithm 1 applied to the current CE pattern set H_t.
9: Add to Π the following rules:
$\{\mathsf{use}(I, J)\} \leftarrow \mathsf{ruleId}(I), \mathsf{literalId}(J)$.
$: \sim \mathsf{use}(I, J).\ [1, I, J]$
10: Add to Π one weak constraint of the form $: \sim \mathsf{not}\ \alpha.\ [1]$ (resp. $: \sim \alpha.\ [1]$) for each target CE instance α included (resp. not included – closed world assumption) in I_t^{true}.
11: Find an optimal answer set $\mathcal{A}_{opt}$ of Π.
12: Remove from $H_\perp$ every body literal δ_i^j for which $\mathsf{use}(i, j) \notin \mathcal{A}_{opt}$ and each rule r_i for which $\mathsf{use}(i, 0) \notin \mathcal{A}_{opt}$.
13: $H_{new} \leftarrow H_\perp$.
14: **return** H_{new}.

observations seen so far, we refer to (Katzouris, Artikis, and Paliouras 2016) for further details.

A successful Hoeffding test results in replacing the parent rule r with its best specialization r_1 and moving one level down in the subsumption lattice, via generating r_1's specializations and subsequently evaluating them on new data.

Figure 1 illustrates the process for an initiation CE pattern. The rules at each level of the lattice represent the specializations of a corresponding rule at the preceding level. The greyed-out part of the search space in Figure 1 represents the portion that has already been searched, while the non greyed-out rule at the third level represents the best-so-far rule that has resulted from a sequence of Hoeffding tests.

The specializations' weights are learnt simultaneously to those of their parent rules as described in Section 4.2, by comparing the specializations' true groundings over time in the MAP-inferred states (generated from "top theories", consisting of parent rules only) and the true states respectively.

Algorithm 3 $\mathsf{WOLED}(B, M, \mathcal{I})$
Input: background knowledge B; mode declarations M; a stream of interpretations $\mathcal{I}$

1: $H_t := \emptyset$.
2: **for each** interpretation $I_t \in \mathcal{I}$ **do**
3: $\quad I_t^{\mathsf{MAP}} := \mathsf{MAPInference}(B, H_t, I_t)$.
4: $\quad$ Receive I_t^{true}.
5: $\quad Mistakes := I_t^{\mathsf{true}} \setminus I_t^{\mathsf{MAP}}$.
6: $\quad H_t \leftarrow \mathsf{SpecializePatterns}(H_t)$.
7: $\quad H_{\mathsf{new}} := \mathsf{LearnNewPatterns}(B, M, I_t, I_t^{\mathsf{MAP}}, I_t^{\mathsf{true}})$.
8: $\quad H_{\mathsf{new}} \leftarrow \mathsf{UpdateWeights}(H_t \cup H_{\mathsf{new}}, mistakes)$.
9: $\quad H_t \leftarrow H_t \cup H_{\mathsf{new}}$.

4.4 Learning New CE patterns

If necessary, the existing CE pattern set H_t is expanded with the addition of new CE patterns, generated in response to erroneous predictions. New initiatedAt/2 (resp. terminatedAt/2) patterns, generated from false negative (*FN*) (resp. false positive (*FP*)) mistakes, have the potential to prevent similar mistakes in the future. For instance, an *FN* mistake at time t, i.e. a target CE instance predicted as false, while actually being true at t, could have been prevented via a pattern that initiates the target CE at some time prior to t.

Generating new CE patterns from the entirety of mistakes may result in a very large number of rules, most of which are redundant. To avoid that, WOLED uses the following strategy for new CE pattern generation, presented in Algorithm 2: First, a set of bottom rules (BRs) is generated (line 4), using the constants in the erroneously predicted atoms to generate ground initiatedAt/2 and terminatedAt/2 atoms, which are placed in the head of a set of initially empty-bodied rules. The bodies of these rules are then populated with literals, grounded with constants that appear in the head, that are true in the current data interpretation I_t. The signatures of allowed body literals are specified via *mode declarations* (De Raedt 2008).

Next, constants in the BRs are replaced by variables and the BR set is "compressed" (line 5) to a bottom theory $H_\perp$, which consists of unique, w.r.t. θ-subsumption, variabilized BRs. The new CE patterns are chosen among those that θ-subsume $H_\perp$. To this end, the generalization technique of (Ray 2009; Katzouris, Artikis, and Paliouras 2015), which allows to search into the space of theories that θ-subsume $H_\perp$, is combined with inference with the existing weighted CE pattern set H_t, yielding a concise set of CE patterns H_{new}, such that an optimal answer set of $B \cup H_t \cup H_{new} \cup I_t$ best-approximates the true state associated with I_t.

To this end, each BR $r_i \in H_\perp$ is "decomposed" in the way shown in line 7 of Algorithm 2, where the head of r_i corresponds to an atom $\mathsf{use}(i, 0)$ and each of its body literals, δ_i^j, to a $\mathsf{try}/3$ atom, which, via the $\mathsf{try}/3$ definitions provided, may be satisfied either by satisfying δ_i^j and an additional $\mathsf{use}(i, j)$ atom, or by "assuming" $\mathsf{not}\ \mathsf{use}(i, j)$. Choosing between these two options is done via ASP optimization in line 9 of Algorithm 2, where the choice rule generates

use/2 atoms that correspond to head atoms/body literals for $H_\perp$, and the subsequent weak constraint minimizes the generated instances to those necessary to approximate the true state, as encoded via the additional weak constraints in line 10. New rules are "assembled" from the bottom rules in $H_\perp$, by following the prescriptions encoded in the use/2 atoms of an optimal answer set of the resulting program, as in line 12.

This is essentially the XHAIL algorithm (Ray 2009) in an ASP context. The difference of our approach from usages of this technique in previous works (Ray 2009; Katzouris, Artikis, and Paliouras 2015), is that here the search into the space of $H'_\perp s$ subsumers is combined with MAP inference with the existing set of weighted CE patterns (line 8, Algorithm 2). Therefore, new patterns are generated only insofar they indeed help to better approximate the true state. This technique allows to generalize from the data in the current interpretation via avoiding to over-fit that data, which may be potentially corrupted by noise.

Once the new CE patterns are generated, their weights (initially set to a near-zero value) are updated based on their groundings in I_t and the true state. Moreover, each new pattern r is associated with the bottom rules from $H_\perp$, which are θ-subsumed by r_i. These bottom rules are used as a pool of literals for further specializing r over time, as described in Section 4.3.

WOLED's learning strategy, mentioned at the beginning of Section 4, is summarized in Algorithm 3.

5 Experimental Evaluation

We present an experimental evaluation of our approach on three CER data sets from the domains of *activity recognition, maritime monitoring* and *vehicle fleet management*.

5.1 Datasets Used

CAVIAR[4] is a benchmark dataset for activity recognition, described in Section 3, consisting of 28 videos with 26,419 video frames in total. We experimented with learning CE patterns for two CEs from CAVIAR, related to two people *meeting each other* and *moving together*, which we henceforth denote by *meeting* and *moving* respectively. There are 6,272 video frames in CAVIAR where *moving* occurs and 3,722 frames where *meeting* occurs. A fragment of a CE definition for *moving* is presented in Table 1(d).

Our second dataset is a publicly available dataset from the field of maritime monitoring[5]. It consists of Automatic Identification System (AIS) position signals collected from vessels sailing in the area of Brest, France, for a period of six months, between October and March 2015. The data have been pre-processed using trajectory compression techniques (Patroumpas et al. 2017), whereby major changes along each vessel's movement are tracked. This process allows to identify critical points along each trajectory, such as a vessel stop, a turn, or a slow motion movement. Using the retained movement features, i.e. the critical points, the trajectory of a vessel may be reconstructed with small deviations from the original one. The maritime dataset has been additionally pre-processed, in order to extract spatial relations between vessels (e.g. vessels being close to each other) and areas of interest, such as protected areas, areas near coast, open-sea areas etc. There 16,152,631 critical points in the maritime dataset, involving 4,961 vessels and 6,894 areas, for a total size of approximately 1,3GB.

The maritime dataset is not labeled in terms of occurring CE instances, we therefore used hand-crafted CE patterns to perform CER on the critical points, thus generating the annotation, and the purpose of learning was to reconstruct the hand-crafted CE patterns. We experimented with learning CE patterns for a CE related to vessels involved in potentially suspicious rendezvous (henceforth denoted by *rendezVous*), which holds when two vessels are stopped, or move with very low speed in proximity to each other in the open sea.

Our third dataset is provided by Vodafone Innovus[6], a commercial vehicle fleet management provider and our partner in the Track & Know[7] EU-funded funded project. The dataset consists of real-world vehicle positional signals (GPS) with a geographical coverage that practically includes the entirety of Greece and a number of neighbouring countries, and a temporal duration of one month. The data consist of time-stamped vehicle positions, in addition to mobility-related events, such as *abrupt acceleration, abrupt deceleration, harsh cornering*, provided by an accelerometer device installed in each commercial vehicle. Moreover, map-matched weather attributes were used to enrich the dataset with contextual information, such as *icy road*. We refer to (Tsilionis et al. 2019) for a detailed account of this dataset.

Similarly to the maritime dataset, due to the lack of CE-related ground truth in the fleet management dataset, we used hand-crafted patterns, developed in collaboration with domain experts in Track & Know, to generate the ground truth CE instances, and the purpose of learning was to reconstruct these patterns from the data. We experimented with learning CE patterns for one target CE related to *dangerous driving*, which holds in a number of occasions, such as abruptly accelerating/decelerating on an icy road, or while over-speeding, etc. The fleet management dataset consists of 4M positional records for a total size of 527 MB.

All experiments were carried-out on a 3.6GHz processor (4 cores, 8 threads) and 16GB of RAM. The code for all algorithms used in these experiments in available online[8].

5.2 Scalability of Inference

The purpose of our first experiment was to assess the scalability of the ASP-based MAP inference process, which lies at WOLED's core. In this respect, we compare the ASP-based version of WOLED, which we henceforth denote by WOLED-ASP, with the version of (Katzouris et al. 2018), which relies on MLN libraries, and which we henceforth denote by WOLED-MLN.

Contrary to WOLED-ASP, which is based entirely on the

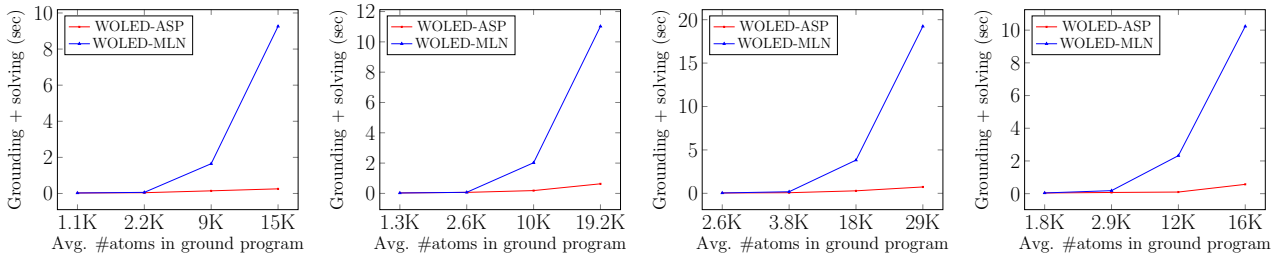
[4]http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

[5]5https://zenodo.org/record/1167595#.WzOOGJ99LJ9

[6]https://www.vodafoneinnovus.com

[7]https://trackandknowproject.eu/

[8]https://github.com/nkatzz/ORL



(a) *Meeting* (a) *Moving* (c) *Rendezvous* (d) *Dangerous driving*

Figure 2: Scalability of MAP inference.

Clingo[9] answer set solver, WOLED-MLN is based on a number of different software tools. It uses the LoMRF library for Markov Logic Networks (Skarlatidis and Michelioudakis 2014), for grounding MLN theories and performing circumscription via predicate completion (Skarlatidis et al. 2015), in order to convert them into a form that supports the non-monotonic semantics of the Event Calculus for reasoning, something that WOLED-ASP has out of the box. MAP inference in WOLED-MLN is performed via a state-of-the-art in MLNs, Integer Linear Programming-based approach, which is introduced in (Huynh and Mooney 2009) and is implemented using the lpsolve[10] solver.

To compare the MAP inference scalability of the two implementations, we used the task of online weight learning with hand-crafted CE patterns, where the learner is required to first perform MAP inference on the incoming interpretations with a fixed-structure CE pattern set, and then update the CE patterns' weights based on their contribution to erroneous inferences in the MAP-inferred state. Given that the weight update cost is negligible and the CE pattern set is fixed, the MAP inference cost is the dominant one in this task, and in turn, it depends on the cost of grounding the current CE pattern set, plus the cost of solving the corresponding weighted MaxSat problem for each incoming interpretation. Note that since the CE pattern sets for each CE are fixed in this experiment, predicate completion in WOLED-MLN is performed only once at the beginning of a run, therefore its cost is negligible.

The data were consumed by the learners in mini-batches, where each mini-batch is an interpretation consisting of data in a particular time interval. We performed weight learning with different mini-batch sizes of 50, 100, 500 & 1000 time points. The size, in total number of literals in a CE pattern set, of the hand-crafted theories used in this experiment was as follows: *meeting* 23 literals, *moving* 28 literals, *rendezVous* 18 literals and *dangerousDriving* 16 literals.

We measured the average MAP inference time (grounding plus solving time) for WOLED-ASP and WOLED-MLN respectively, throughout a single-pass over the data, for different mini-batch sizes. Note that as the mini-batch size grows, so does the size of the corresponding ground program from which the MAP-inferred state is extracted.

[9]https://potassco.org/

[10]https://sourceforge.net/projects/lpsolve

Figure 2 presents the results, which indicate that the growth in the size of the ground program, as the mini-batch size increases, entails an exponential growth to the MAP inference cost for WOLED-MLN. In contrast, thanks to Clingo's highly optimized grounding and solving abilities, MAP inference with WOLED-ASP takes near-constant time.

5.3 Online Structure & Weight Learning Performance

In our next experiment we assess WOLED-ASP's predictive performance and efficiency in the task of online structure & weight learning and we compare it to (i) WOLED-MLN; (ii) OLED (Katzouris, Artikis, and Paliouras 2016), the crisp version of the algorithm that learns unweighted CE patterns; (iii) HandCrafted, a set of predefined rules for each CE and (iv) HandCrafted-WL, the rules in HandCrafted with weights learnt by WOLED-ASP.

To assess the predictive performance of the systems compared we used two methods: *Prequential* (predictive sequential) evaluation, or *interleaved test-then-train* (Bifet et al. 2018), where each incoming data interpretation is first used to evaluate the current CE pattern set and then to update its structure and weights, and standard cross-validation. In prequential evaluation we typically measure the average prediction loss over time, which is an indication of a learner's ability to incorporate new knowledge that arrives over time into the current model. With cross-validation we assess a learner's generalization abilities, by evaluating the predictive performance of a learnt model on a test set.

The results are presented in Table 3, where the following statistics are reported for each one of the systems being compared: (i) the average prequential loss, which, for the n-th mini-batch in the learning process is defined as S/n, where S is the cumulative sum of false positive and false negative predictions up to that time. The value reported in 3 is the final value of S/n in a prequential run; (ii) F_1-score on a test set. For CAVIAR we used tenfold cross-validation and the reported F_1-scores are micro-averages obtained from ten different test sets. For the maritime and the fleet management datasets, whose size makes tenfold cross-validation impractical, we used half the dataset for training and half for testing, so the reported F_1-scores are obtained for the latter half; (iii) CE pattern set sizes (total number of literals) at the end of a prequential run (i.e., after a single-pass over

	Method	Prequential Loss	F_1-score (test set)	Theory size	Inference Time (sec)	Pred. Compl. Time (sec)	Total Time (sec)
Moving	WOLED-ASP	**1.723**	**0.821**	26	15	–	112
	WOLED-MLN	2.817	0.801	47	187	28	478
	OLED	3.755	0.730	**24**	**13**	–	74
	HandCrafted	6.342	0.637	28	–	–	–
	HandCrafted-WL	4.343	0.702	28	16	–	**52**
Meeting	WOLED-ASP	**1.212**	**0.887**	34	**12**	–	82
	WOLED-MLN	2.554	0.841	56	134	12	145
	OLED	3.224	0.782	42	10	–	36
	HandCrafted	5.734	0.735	**23**	–	–	–
	HandCrafted-WL	4.024	0.753	**23**	13	–	**31**
Rendezvous	WOLED-ASP	**0.023**	0.98	18	647	–	4,856
	WOLED-MLN	0.088	0.98	18	2,923	434	6,218
	OLED	0.092	0.98	18	**623**	–	**4,688**
Dang.Drive	WOLED-ASP	**0.045**	0.99	**21**	341	–	2,465
	WOLED-MLN	1.234	0.99	28	926	287	3,882
	OLED	1.756	0.99	**21**	**312**	–	**2,435**

Table 3: Online structure & weight learning results.

a dataset); (iv) Total inference time at the end of a prequential run (MAP inference for WOLED-ASP, WOLED-MLN & HandCrafted-WL, crisp logical inference for OLED); (v) For WOLED-MLN, total time spent on predicate completion at the end of a prequential run; (vi) Total training time at the end of a prequential run, which includes time spent on CE pattern generation, computing θ-subsumption etc, i.e. the dominant costs involved in learning CE patterns structure. Note that we report on (iv), (v), (vi) only for approaches that require training (i.e., not for HandCrafted). Also, we did not experiment with hand-crafted CE patterns in the maritime and the fleet management datasets, since in these datasets hand-crafted CE patterns were used to generate the ground truth in the first place.

In addition to the different implementations of probabilistic inference, an important difference between WOLED-ASP and WOLED-MLN from an algorithmic perspective, lies in the new CE pattern generation process of Section 4.4. Thanks to its ASP-based implementation and the underlying optimization tools, WOLED-ASP is able to perform the search for new CE patterns, while taking into account the contribution of the weights of existing ones in approximating the true state of an interpretation I_t. As a result, it generates new patterns only when this does indeed result in a better approximation of the true state, given the existing weighted patterns. In contrast, WOLED-MLN, lacks this ability. It generates a bottom theory $H_\perp$ from the erroneously predicted atoms, and then attempts to gradually learn a high-quality CE pattern from the rules therein, regardless of their quality. In comparison, WOLED-ASP's strategy may lead, in principle, to simpler theories of more meaningful rules and lower online error (i.e. better prequential performance). The results in Table 3 seem to validate this claim. WOLED-ASP achieves the best prequential performance among all compared approaches. It also achieves superior cross-validation performance, as compared to WOLED-MLN (test set F_1-scores), which indicates that its new CE pattern generation strategy affects its ability to generalize. Moreover, WOLED-ASP learns simpler CE patterns sets, as shown by the theory size statistic. OLED lacks the ability for weight learning, while HandCrafted-WL does not update the CE patterns' structure, which explains their inferior prequential and cross-validation performance. The trade-off is their lower training times.

Regarding efficiency, it may be seen by comparing inference times to total training times, that the dominant cost is related to structure learning tasks (recall that total training times factor-in such costs). Yet, in comparison to WOLED-MLN, WOLED-ASP achieves significantly lower costs for MAP inference, which approximate the cost of OLED's crisp logical inference. In addition to its more sophisticated CE pattern creation strategy, which tends to generate fewer CE patterns of high quality, this results in WOLED-ASP being significantly more efficient than WOLED-MLN. Note also, that an additional, not negligible cost for WOLED-MLN stems from the necessity of predicate completion.

6 Conclusions & Future Work

We presented an online algorithm for learning complex event patterns in the form of weighted Event Calculus rules. Our system is entirely implemented in Answer Set Programming and it is capable of combining temporal reasoning under uncertainty via probabilistic logical inference, with online structure and weight learning techniques. Our empirical evaluation on three CE datasets indicates that it is significantly more efficient than an implementation based on Markov Logic Networks, while achieving superior online predictive performance and having better generalization abilities. Future work involves combination of semi-supervised/active learning strategies, capable of handling the scarcity of labeled data in streaming settings.

Acknowledgements

This work is supported by the project entitled "Track & Know: Big Data for Mobility Tracking Knowledge Extraction in Urban Areas", which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780754, and by the project entitled "INFORE: Interactive Extreme-Scale Analytics and Forecasting", which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825070.

References

Alevizos, E.; Skarlatidis, A.; Artikis, A.; and Paliouras, G. 2017. Probabilistic complex event recognition: A survey. *ACM Comput. Surv.* 50(5):71:1–71:31.

Artikis, A.; Skarlatidis, A.; Portet, F.; and Paliouras, G. 2012. Logic-based event recognition. *The Knowledge Engineering Review* 27(04):469–506.

Artikis, A.; Sergot, M.; and Paliouras, G. 2015. An event calculus for event recognition. *Knowledge and Data Engineering, IEEE Transactions on* 27(4):895–908.

Athakravi, D.; Corapi, D.; Broda, K.; and Russo, A. 2013. Learning through hypothesis refinement using answer set programming. In *Inductive Logic Programming*. Springer. 31–46.

Bifet, A.; Gavaldà, R.; Holmes, G.; and Pfahringer, B. 2018. *Machine learning for data streams: with practical examples in MOA*. MIT Press.

Cugola, G., and Margara, A. 2012. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)* 44(3):15.

De Raedt, L.; Kersting, K.; Natarajan, S.; and Poole, D. 2016. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 10(2):1–189.

De Raedt, L. 2008. *Logical and relational learning*. Springer Science & Business Media.

Domingos, P. M., and Hulten, G. 2000. Mining high-speed data streams. In *ACM SIGKDD*, 71–80.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Guimarães, V.; Paes, A.; and Zaverucha, G. 2019. Online probabilistic theory revision from examples with proppr. *Mach. Learn.* 108(7):1165–1189.

Huynh, T. N., and Mooney, R. J. 2009. Max-margin weight learning for markov logic networks. In *ECML-2009*. Springer. 564–579.

Huynh, T. N., and Mooney, R. J. 2011. Online max-margin weight learning for markov logic networks. In *SDM*, 642–651. SIAM.

Katzouris, N.; Artikis, A.; and Paliouras, G. 2015. Incremental learning of event definitions with inductive logic programming. *Machine Learning* 100(2-3):555–585.

Katzouris, N.; Artikis, A.; and Paliouras, G. 2016. Online learning of event definitions. *TPLP* 16(5-6):817–833.

Katzouris, N.; Michelioudakis, E.; Artikis, A.; and Paliouras, G. 2018. Online learning of weighted relational rules for complex event recognition. In *ECML-PKDD 2018*, 396–413.

Law, M.; Russo, A.; and Broda, K. 2018. Inductive learning of answer set programs from noisy examples. *Advances in Cognitive Systems*.

Lee, J., and Wang, Y. 2016. Weighted rules under the stable model semantics. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR*, 145–154. AAAI Press.

Lee, J.; Talsania, S.; and Wang, Y. 2017. Computing LPMLN using ASP and MLN solvers. *Theory Pract. Log. Program.* 17(5-6):942–960.

Michelioudakis, E.; Skarlatidis, A.; Paliouras, G.; and Artikis, A. 2016. Osla: Online structure learning using background knowledge axiomatization. In *ECML*, 232–247. Springer.

Mueller, E. T. 2014. *Commonsense reasoning: an event calculus based approach*. Morgan Kaufmann.

Patroumpas, K.; Alevizos, E.; Artikis, A.; Vodas, M.; Pelekis, N.; and Theodoridis, Y. 2017. Online event recognition from moving vessel trajectories. *GeoInformatica* 21(2):389–427.

Ray, O. 2009. Nonmonotonic abductive inductive learning. *Journal of Applied Logic* 7(3):329–340.

Skarlatidis, A., and Michelioudakis, E. 2014. Logical Markov Random Fields (LoMRF): an open-source implementation of Markov Logic Networks.

Skarlatidis, A.; Paliouras, G.; Artikis, A.; and Vouros, G. A. 2015. Probabilistic event calculus for event recognition. *ACM Transactions on Computational Logic (TOCL)* 16(2):11.

Tsilionis, E.; Koutroumanis, N.; Nikitopoulos, P.; Doulkeridis, C.; and Artikis, A. 2019. Online event recognition from moving vehicles: Application paper. *Theory Pract. Log. Program.* 19(5-6):841–856.