TPLP: Page 1–25. © The Author(s), 2021. Published by Cambridge University Press
doi:10.1017/S1471068421000107

Online Learning Probabilistic Event Calculus Theories in Answer Set Programming[*]

NIKOS KATZOURIS and GEORGIOS PALIOURAS
Institute of Informatics and Telecommunications, National Center for Scientific Research (NCSR) "Demokritos", Athens, Greece
(*e-mail:* nkatz@iit.demokritos.gr,paliourg@iit.demokritos.gr)

ALEXANDER ARTIKIS
Institute of Informatics and Telecommunications, National Center for Scientific Research (NCSR) "Demokritos", Athens, Greece
and Department of Maritime Studies, University of Piraeus, Piraeus, Greece
(*e-mail:* a.artikis@iit.demokritos.gr)

submitted 9 November 2020; revised 17 June 2021; accepted 21 June 2021

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 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 number of state-of-the-art batch learning algorithms on CER data sets for activity recognition, maritime surveillance and fleet management. Our results demonstrate the superiority of our novel approach, both in terms of efficiency and predictive performance. This paper is under consideration for publication in Theory and Practice of Logic Programming (TPLP).

KEYWORDS: inductive logic programming and multi-relational data mining, knowledge representation and nonmonotonic reasoning

1 Introduction

Complex Event Recognition (CER) systems (Cugola and Margara 2012) detect occurrences of *complex events* (CEs) in streaming input, defined as spatiotemporal combinations of *simple events* (e.g. sensor data), using a set of CE patterns. Since such patterns are not always known beforehand, machine learning algorithms for discovering them from data are highly useful. Thanks to their efficiency, online learning algorithms are of special interest. Such algorithms should be resilient to noise and uncertainty, which are ubiquitous in temporal data streams (Alevizos *et al*. 2017), while taking into account

[*]This paper is an extended version of Katzouris and Artikis (2020), which has been nominated as a candidate for TPLP's rapid publication track by KR2020's program committee.

commonsense phenomena (Mueller 2014), which often characterize dynamic application domains, such as CER.

Logic-based CER systems (Artikis *et al.* 2012) stand up to these challenges. They combine reasoning under uncertainty with machine learning, via Statistical Relational AI techniques (De Raedt *et al.* 2016), while supporting reasoning with time and change, via action formalisms such as the Event Calculus (Artikis *et al.* 2015).

We advance the state of the art in online learning for CER 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. The proposed algorithm is based entirely on Answer Set Programming (ASP) (Lifschitz 2019), 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.

We compare WOLED's ASP-based implementation to an Markov Logic Networks (MLN)-based one, and to a number of state-of-the art online and batch structure and weight learning algorithms, on three CER data sets for *activity recognition*, *maritime surveillance* and *vehicle fleet management*. Our results demonstrate the superiority of WOLED, both in terms of efficiency and predictive performance.

2 Related work

Event Calculus-based CER (Artikis *et al.* 2015) was combined with MLNs in Skarlatidis *et al.* (2015), in order to deal with the noise and 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 MLNs' first-order logic representations 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 *et al.* (2017). This 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 standpoint and investigate the usefulness of these ideas in challenging domains, such as CER. We also propose a methodology for online structure and the weight learning using ASP tools.

Regarding machine learning, a number of algorithms in non-monotonic Inductive Logic Programming (ILP), such as XHAIL (Ray 2009), TAL (Athakravi *et al.* 2013), and ILASP (Law *et al.* 2018) are capable of learning Event Calculus theories, see Katzouris (2017) for a comprehensive review of such approaches. 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 are either crisp learners (Katzouris *et al.* 2019), or

they rely on MLNs (Katzouris *et al*. 2018; Michelioudakis *et al*. 2016), so they suffer from the same limitations discussed earlier in this section. A recent online learner based on probabilistic theory revision (Guimarães *et al*. 2019) is limited to Horn logic and cannot handle Event Calculus reasoning.

3 Background

Answer Set Programming. In what follows a rule r is an expression of the form $\alpha \leftarrow \delta_1, \ldots, \delta_n$, where α is an atom, called the head of r, $\delta_i's$ are literals (possibly negated atoms), which collectively form the body of r and commas in the bodies of rules denote conjunction. A rule is ground if it contains no variables and a grounding of a rule r is called an instance of r. A (Herbrand) interpretation is a collection of true ground facts. An interpretation I satisfies an atom α iff $\alpha \in I$. I satisfies a ground rule iff satisfying each literal in the body implies that the head atom is also satisfied and it satisfies a non-ground rule r if it satisfies all ground instances of r. An interpretation I is a model of a logic program Π (collection of rules) if it satisfies every rule in Π and it is a minimal model if no strict subset of I has this property. An interpretation I is an answer set of Π if it is a minimal model of the reduct of Π, that is, the negation-free, ground program that results by removing from the ground version of Π all rules with a negated body literal not satisfied by I and removing all negated literals from the bodies of the remaining rules.

A choice rule is an expression of the form $\{\alpha\} \leftarrow \delta_1, \ldots, \delta_n$ 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.

The Event Calculus is a temporal logic for reasoning about events and their effects. Its ontology comprises time points (integers), fluents, that is, properties which have certain values in time, and events, that is, 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 an 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 initiatedAt/2 and terminatedAt/2 rules.

As an example we use the task of activity recognition, as defined in the CAVIAR project[1]. The CAVIAR data set consists of videos of a public space, where actors per-

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

Table 1. ***(a), (b)*** *The basic predicates and the Event Calculus axioms.* ***(c)*** *Example CAVIAR data. At time point 1 person with* id_1 *is* walking*, her* (X, Y) *coordinates are* $(201, 454)$ *and her direction is* $270°$*. The target CE atoms (true state – supervision) for time point 1 state that 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*. For example, 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 than 30*

(a)

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

(b) Axioms of the Event Calculus

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

$\mathsf{holdsAt}(F, T+1) \leftarrow \mathsf{holdsAt}(F, T), \mathsf{not}\ \mathsf{terminatedAt}(F, T).$ (2)

(c)

Observations I_1 ***at time 1:***

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

Target CE instances at time 1:

$\{\mathsf{holdsAt}(move(id_1, id_2), 2),$
$\mathsf{holdsAt}(move(id_2, id_1), 2)\}$

(d) Weighted CE patterns:

1.283 $\mathsf{initiatedAt}(move(X, Y), T) \leftarrow \mathsf{happensAt}(walk(X), T), \mathsf{happensAt}(walk(Y), T),$
$close(X, Y, 25, T), orientation(X, Y, 45, T).$

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

form 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 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 1(c) presents the target CE instances corresponding to the observations in I_1.

In what follows 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:

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

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

4 Structure and weight learning in ASP

The task that WOLED, our proposed algorithm, 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 assumed 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.

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[2], which amounts to

[2] Marginal inference, that is, 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, or

computing a most probable answer set $\mathscr{A}$ of $\Pi = B \cup H_t \cup I_t$. From equations (1) and (2) it follows that

$$\mathscr{A} = \underset{I \in \mathsf{ans}(\Pi)}{\arg\max} P_{\Pi}(I) = \underset{I \in \mathsf{ans}(\Pi)}{\arg\max} W_{\Pi}(I) = \underset{I \in \mathsf{ans}(\Pi)}{\arg\max} \sum_{r \in R_I} w_r \tag{3}$$

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 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 $\mathscr{A}$ of $\Pi = B \cup T(H_t) \cup I_t$, that is, the satisfaction of a ground instance of r_i by $\mathscr{A}$, incurs a cost of $-w_i$ on $\mathscr{A}$, 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 Figure 1, where we assume that the target CE to be recognized is a. (a) presents a CE pattern set H_t, that is, a set of

utilizing techniques for sampling from such answer sets. We are not concerned with marginal inference in this work.

Algorithm 1 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 $\mathscr{A}_{opt}$ of $B \cup T(H_t) \cup I_t$.
9: **return** the target CE instances in $\mathscr{A}_{opt}$.

(a) Current CE pattern set H_t:
$11\ \mathsf{initiatedAt}(a,T) \leftarrow \mathsf{happensAt}(b,T).\quad (rule_1)$
$13\ \mathsf{terminatedAt}(a,T) \leftarrow \mathsf{happensAt}(c,T).\quad (rule_2)$
$-2\ \mathsf{initiatedAt}(a,T) \leftarrow \mathsf{happensAt}(d,T).\quad (rule_3)$

(b) Current data interpretation I_t:
$\{\mathsf{happensAt}(c,1), \mathsf{happensAt}(b,2), \mathsf{happensAt}(c,5), \mathsf{happensAt}(d,8)\}$

(c) Inferred state with crisp logical inference:
$\{\mathsf{holdsAt}(a,3), \mathsf{holdsAt}(a,4), \mathsf{holdsAt}(a,5), \mathsf{holdsAt}(a,9), \mathsf{holdsAt}(a,10)\}$

(d) Program $T(H_t)$ for MAP inference:
$\mathsf{initiatedAt}(a,T) \leftarrow \mathsf{satisfied}(\mathsf{vars}(T), rule_1).$
$\{\mathsf{satisfied}(\mathsf{vars}(T), rule_1)\} \leftarrow \mathsf{happensAt}(b,T).$
$: \sim \mathsf{satisfied}(\mathsf{vars}(T), rule_1).\ [-11, \mathsf{vars}(T), rule_1]$
$\mathsf{terminatedAt}(a,T) \leftarrow \mathsf{satisfied}(\mathsf{vars}(T), rule_2).$
$\{\mathsf{satisfied}(\mathsf{vars}(T), rule_2)\} \leftarrow \mathsf{happensAt}(c,T).$
$: \sim \mathsf{satisfied}(\mathsf{vars}(T), rule_2).\ [-13, \mathsf{vars}(T), rule_2]$
$\mathsf{initiatedAt}(a,T) \leftarrow \mathsf{satisfied}(\mathsf{vars}(T), rule_3).$
$\{\mathsf{satisfied}(\mathsf{vars}(T), rule_3)\} \leftarrow \mathsf{happensAt}(d,T).$
$: \sim \mathsf{satisfied}(\mathsf{vars}(T), rule_3).\ [2, \mathsf{vars}(T), rule_3]$

(e) Inferred state with MAP inference:
$\{\mathsf{holdsAt}(a,3), \mathsf{holdsAt}(a,4), \mathsf{holdsAt}(a,5),$
$\mathsf{satisfied}(\mathsf{vars}(5), rule_2), \mathsf{satisfied}(\mathsf{vars}(2), rule_1)\}$

Fig. 1. ASP-based MAP inference example.

initiation and termination condition for the target CE, a. We assume that the actual real-valued weights of the patterns have been converted into integers; (b) presents the current data interpretation I_t; (c) presents the inferred state obtained with crisp logical inference, that is, 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 ignored. 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

Algorithm 2 LearnNewRules$(B, M, H_t, I_t, I_t^{\mathsf{MAP}}, I_t^{\mathsf{true}})$

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: $\Pi := \emptyset$, $H_{new} := \emptyset$, $H_\perp := \emptyset$, $T(H_\perp) := \emptyset$
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 $\mathscr{A}_{opt}$ of Π.
12: Remove from $H_\perp$ every body literal δ_i^j for which $\mathsf{use}(i, j) \notin \mathscr{A}_{opt}$ and each rule r_i for which $\mathsf{use}(i, 0) \notin \mathscr{A}_{opt}$.
13: $H_{new} \leftarrow H_\perp$.
14: **return** H_{new}.

the occurrence of $\mathsf{happensAt}(d, 8) \in I_t$ reinitiates a, via $rule_3 \in H_t$, so a holds at times 9 & 10.

(d) in Figure 1 presents the program $T(H_t)$ obtained from H_t, via the transformation in Algorithm 1, to allow for MAP inference; Finally, (e) presents the MAP-inferred state, that is, 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 $\mathsf{satisfied}/2$ instances in the MAP-inferred state correspond to the ground atoms $\mathsf{terminatedAt}(a, 5)$ and $\mathsf{initiatedAt}(a, 2)$, which, along with inertia, are responsible for the target CE inferences.

4.2 Learning new rules

Right after making a prediction via MAP inference on the current interpretation the true state is revealed to the learner, compared against the MAP-inferred state and the erroneous predictions are identified. The existing CE pattern set H_t is expanded with the addition of new rules, generated in response to these mistakes, via the addition of new initiatedAt/2 (resp. terminatedAt/2) patterns, generated from false-negative (FN) (resp. false-positive (FP)) mistakes, which have the potential to prevent similar mistakes in the future. For instance, an FN mistake at time t, that is, 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, many of which are redundant, or generated from noisy data points. To avoid that, WOLED instantiates an optimization process that seeks a good trade-off between theory complexity and accuracy. This is done by combining structure induction techniques from non-monotonic ILP with probabilistic reasoning with the existing CE pattern set H_t, in order to learn a new set of CE patterns R, which is as compressive as possible, while at the same time, the result of reasoning with $B \cup H_t \cup R$ on the current interpetation approximates the corresponding true state as close possible. Viewing the current input interpretation as a small training set, such a process corresponds to the standard machine learning practice of jointly minimizing training error and model complexity.

To realize this process, WOLED employs the strategy for new rule generation, presented in Algorithm 2: First, a set of bottom rules (BRs) is created (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), 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} with the property 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 try/3 atom, which, via the try/3 definitions provided, may be satisfied either by satisfying δ_i^j and an additional $\mathsf{use}(i, j)$ atom, or by "assuming" 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 prescrip-

Input interpretation:

$I_t = \{\text{happensAt}(c,1), \text{happensAt}(b,2), \text{happensAt}(c,5), \text{happensAt}(d,8)\}$

MAP-inferred state using the weighted theory from Table 1:

$I_t^{\text{MAP}} = \{\text{holdsAt}(a,3), \text{holdsAt}(a,4), \text{holdsAt}(a,5)\}$

True state:

$I_t^{\text{true}} = \{\text{holdsAt}(a,3), \text{holdsAt}(a,4), \text{holdsAt}(a,5), \text{holdsAt}(a',6), \text{holdsAt}(a',7), \text{holdsAt}(a',8), \text{holdsAt}(a',9), \text{holdsAt}(a',10)\}$

Mode Declarations:

$head(\text{initiatedAt}(a',time))$ $head(\text{terminatedAt}(a',time))$
$head(\text{initiatedAt}(a,time))$ $head(\text{terminatedAt}(a,time))$
$body(\text{happensAt}(c,time))$ $body(\text{happensAt}(b,time))$
$body(\text{not happensAt}(e,time))$ $body(\text{holdsAt}(a,time))$

Bottom rule $\perp$ generated from time point 5:

Ground: $\text{initiatedAt}(a',5) \leftarrow \text{happensAt}(c,5), \text{not happensAt}(e,5), \text{holdsAt}(a,5).$

Lifted: $\text{initiatedAt}(a',T) \leftarrow \text{happensAt}(c,T), \text{not happensAt}(e,T), \text{holdsAt}(a,T).$

Syntactically transformed program $H_\perp$ for new rule induction:

$\text{initiatedAt}(a',T) \leftarrow use(1,0), try(1,1,T), try(1,2,T), try(1,3,T).$
$try(1,1,T) \leftarrow use(1,1), \text{happensAt}(c,T).$
$try(1,1,T) \leftarrow \text{not } use(1,1), time(T).$
$try(1,2,T) \leftarrow use(1,2), \text{not happensAt}(e,T).$
$try(1,2,T) \leftarrow \text{not } use(1,2), time(T).$
$try(1,3,T) \leftarrow use(1,3), \text{holdsAt}(a,T).$
$try(1,3,T) \leftarrow \text{not } use(1,3), time(T).$

Fig. 2. Example of new rule induction in response to prediction mistakes.

tions 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 structure induction technique in previous works (Ray 2009; Katzouris *et al*. 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, given the already existing weighted rules. This technique allows to generalize from the data in the current interpretation, while taking into account previously discovered patterns and their relative quality, as reflected by their weights.

Example 2

We illustrate the rule induction technique in Algorithm 2 via an example. Recall Example 1 and assume that after making a prediction and generating the MAP-inferred state we receive the true state presented in Figure 2. Assume also that we have an additional target CE here, a'. The true state in Figure 2 differs from the MAP-inferred one as it contains holdsAt/2 atoms concerning a'. These atoms are missing from the MAP-inferred state; therefore, they are false-negative (FN) predictions, that is, true instances of a target complex event, which are not recognized by the weighted theory of Figure 1. In order to revise the current theory toward eliminating the FN's we employ the strategy of Algorithm 2. First, a set of ground atoms that eliminate the erroneous predictions if added to the current theory are generated via abductive reasoning. These atoms will serve as heads for new rules and their signatures are predefined via mode declarations

(see Figure 2). In our example one such atom, $\mathsf{initiatedAt}(a', 5)$, suffices (note that a' in the true state holds from time 6 onwards, therefore, it must have been initiated in the previous time point). A bottom rule $\perp$ having this atom in the head is generated from the data in the current interpretation, as shown in Figure 2. The signatures of literals in the body of $\perp$ are specified via the mode declarations and the actual atoms are groundings of such literals, generated using constants in the head, that are true in the data. The "lifted" version of $\perp$ shown in Figure 2 is then transformed into program $H_\perp$ in Figure 2, forming a search space for learning a new rule.

Approximating the true state is then realized via finding an optimal answer set of a program Π consisting of the following parts: (i) the axioms of the Event Calculus (background knowledge); (ii) program $T(H_t)$ from Figure 1, used for probabilistic inference with the weighted theory H_t; (iii) program $H_\perp$ from Figure 2, used for new rule induction; (iv) a choice rule for *use/2* atoms and weak constraints that minimize the *use/2* atoms included in the optimal answer set, thus encoding a preference for simpler rules in the rule induction process (line 9, Algorithm 2); (v) weak constraints that minimize the "disagreement" between the optimal answer set and the true state w.r.t. the target CE instances included in the optimal answer set (line 10, Algorithm 2).

An optimal answer set of program Π contains the atoms $use(1, 0), use(1, 1)$, and $use(1, 3)$, which correspond to the rule $r^\star$: $\mathsf{initiatedAt}(a', T) \leftarrow \mathsf{happensAt}(c, T), \mathsf{holdsAt}(a, T)$. This is the simplest rule r with the property that $H_t \cup r$ correctly accounts for the true state.

Remark. Note that reasoning with the weighted theory H_t is necessary in order to learn the new rule $r^\star$ in this example. Indeed, $rule_1$ in Figure 1 initiates a at time 2, thus causing it to hold at time 5 and allowing the $\mathsf{holdsAt}(a, T)$ atom to be added to the body of $r^\star$. Observe that without this atom, the more general rule, $\mathsf{initiatedAt}(a', T) \leftarrow \mathsf{happensAt}(c, T)$ would yield a number of false- positive predictions, since it would initiate a' at time 1, due to the $\mathsf{happensAt}(c, 1)$ atom in the input data, thus causing a' to (erroneously) hold in the interval $[2, 5]$. It can be seen by comparing costs in the optimization process that "settling" for the initial false-negative predictions for a' in the interval $[6, 10]$ (by not learning a new rule at all) is more cost efficient than learning the rule $\mathsf{initiatedAt}(a', T) \leftarrow \mathsf{happensAt}(c, T)$, which does retrieve the false negatives, but causes the false positives for a' in the interval $[2, 5]$. Therefore, had we not used H_t as background knowledge, we would not have learned a new rule from the data in this example. It is worth mentioning that the same would have happened if we used H_t as a crisp theory, that is, without the rules' weights. In that case $rule_3$ from Figure 1 would be responsible for a number of false positives in the inferred state. This example, therefore, demonstrates the merit of combining probabilistic inference with existing rules with the new rule induction process.

4.3 Weight learning

The CE patterns' weights (which are initialized to a close-to-zero value) are updated by comparing their true groundings in the inferred and the true state, respectively. For a target CE α and an $\mathsf{initiatedAt}/2$ (resp. $\mathsf{terminatedAt}/2$) 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 toward correct predictions are promoted, while those that contribute to erroneous predictions are downweighted.

As in Katzouris *et al.* (2018), we use the AdaGrad algorithm (Duchi *et al.* 2011) for weight updates, a version of Gradient Descent that dynamically adapts the learning rate, that is, 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, called the *prediction-based loss*, is a simple variant of the max-margin loss for structured prediction for MLNs (Huynh and Mooney 2011), whose subgradient is the vector with Δg_i in its ith coordinate, where $\Delta g_i = g_{r_i}^{MAP} - g_{r_i}^{true}$ denotes the difference in the true groundings of the ith pattern, r_i, in the MAP-inferred and the true state, respectively. Essentially, Δg_i counts prediction mistakes which are relevant to r_i, either false- positive mistakes ($\Delta g_i > 0$), for which r_i is responsible, or false-negative predictions ($\Delta g_i < 0$), committed by the entire theory, which r_i could have helped prevent. The weight update rule for the ith CE pattern r_i is then:

$$w_i^{t+1} = sign\left(w_i^t - \frac{\eta}{C_i^t}\Delta g_i^t\right)\ max\left\{0, |w_i^t - \frac{\eta}{C_t^i}\Delta g_i^t| - \lambda\frac{\eta}{C_t^i}\right\} \tag{4}$$

where 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 proportional to the sum of r_i's past subgradients (the Δg_i's) (plus a $\delta \geq 0$ to avoid division by zero in η/C_i^t). Note that according to equation (4), a $\Delta g_i > 0$, that is, a case where r_i is responsible for false-positive predictions, leads to a weight demotion for r_i, while a $\Delta g_i < 0$ leads to a promotion of its weight, that can help the theory retrieve the false-negative misses.

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 utility, expressed by Δg_i^t. Smaller values for C_i^t correspond to "rare", but potentially highly informative features, and therefore lead to weight updates of larger magnitude. The "informativeness" of these features is reflected in the magnitude of the current subgradient Δg_i, since, regardless of the value of C_i^t, zero, or very small values of the current subgradient (corresponding to noninformative features) have very small effect to the weight.

The regularization term in equation (4), $\lambda\frac{\eta}{C_t^i}$, is the amount by which the ith CE pattern's weight is discounted when $\Delta g_i^t = 0$. As usual, the role of regularization is to introduce a bias toward simpler models, in this case by eventually (over time) pushing to zero the weights of irrelevant rules, that play no significant role in helping the theory make correct predictions.

4.4 Revising existing CE patterns' structure

Although the rules induced with the process described in Section 4.2 are useful locally, that is, w.r.t. the current input interpretation I_t, they may be proven inadequate w.r.t. a more "global" view of the data. A case where this typically occurs is inducing the

simplest rule that helps approximate the true state in I_t, which turns out to be over-general once larger regions of the data are taken into account. Weight learning may help suppress the effects of such rules in the predictive performance of the model, but this is not enough: Since the data are processed incrementally in small batches, it may be the case that a "data view" large-enough for learning a high-quality target rule may never occur, causing the learning process to fail.

A remedy is to revise the rules' structure over time, as larger portions of the data are revealed. Similarly to OLED (Katzouris *et al.* 2016), WOLED does so via a classical in ILP, hill-climbing search process, searching for a high-quality CE pattern into a *subsumption lattice* defined by a bottom rule. Such bottom rules are generated during the rule induction process of Section 4.2. Each induced rule r is associated with a bottom rule $\perp_r$ (such that r θ-subsumes $\perp_r$), which serves as a pool to draw literals from, in order to specialize r over time. This process is online, using the data in the incoming interpretations 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.

In particular, at each point in time a parent rule and its specializations are evaluated 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:

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

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 observations seen so far, we refer to Katzouris *et al.* (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 3 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 grayed out part of the search space in Figure 3 represents the portion that has already been searched, while the non grayed 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, 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.

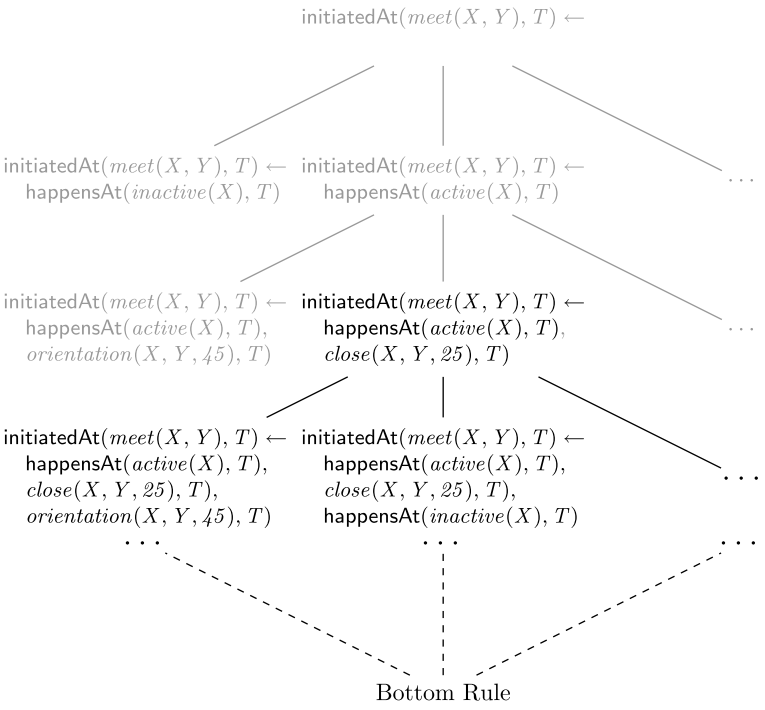

Fig. 3. A subsumtion lattice.

5 Discussion on implementations

We highlight the differences between 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 Clingo[3] 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[4] solver.

Another important difference between WOLED-ASP and WOLED-MLN, from an algorithmic perspective, lies in the new CE pattern generation process. As discussed in Section 4.2, WOLED-ASP is able to perform the search for new structure, while taking into account the contribution of the weights of existing patterns in approximating the true state. 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

[3] https://potassco.org/
[4] https://sourceforge.net/projects/lpsolve

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.

6 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*.

Data sets. *CAVIAR*[5] is a benchmark data set 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 6272 video frames in CAVIAR where *moving* occurs and 3722 frames where *meeting* occurs. A fragment of a CE definition for *moving* is presented in Table 1(d).

Our second data set is a publicly available data set from the field of maritime monitoring[6]. It consists of Automatic Identification System (AIS) position signals collected from vessels sailing around the area of Brest, France, for a period of six months, between October and March 2015. The data have been preprocessed using trajectory compression techniques (Patroumpas *et al*. 2017) that identify "critical points" in a trajectory, that is, mobility features, such as vessel stops, turns, slow motion movements, etc. The critical points maritime data set has been further 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 are 16,152,631 critical points in the maritime data set, involving 4961 vessels and 6894 areas, for a total size of approximately 1,3GB.

The maritime data set 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. 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. Since such behavior is often related to illegal activities, tracking it is of special interest for maritime surveillance.

Our third data set is provided by Vodafone Innovus[7], a commercial vehicle fleet management provider and our partner in the Track & Know[8] EU-funded funded project. The data consist of time-stamped vehicle positions (GPS), 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 data set with contextual information, such as *icy road*. We refer to Tsilionis *et al*. (2019) for a detailed account of this data set.

[5] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/
[6] https://zenodo.org/record/1167595#.WzOOGJ99LJ9
[7] https://www.vodafoneinnovus.com
[8] https://trackandknowproject.eu/

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

In order to provide some insight into the different domains used for these experiments, Table 2 presents the language bias used for all target CEs, in addition to some indicative complex event definitions from each domain (note that for the *moving* complex event, an indicative fragment of its specification was shown in Table 1).

All experiments were carried out on a machine with a 3.6 GHz processor (4 cores, 8 threads) and 16 GB of RAM. Clingo (v. 5.4.0) was used with the –opt-strategy=usc option, which significantly speeds-up the optimization process. The hyperparameters for the different algorithms compared in these experiments were set as follows: For AdaGrad, $\eta = 1.0, \lambda = 0.01, \delta = 1.0$. The significance parameter for Hoeffding tests was set to $\delta = 10^{-2}$. The code for all algorithms used in these experiments is available online[9].

6.1 Scalability of inference

The purpose of our first experiment was to compare the scalability of the ASP-based MAP inference process, which lies at WOLED-ASP's core, to that of WOLED-MLN's. To that end 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 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, and 1000 time points. 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.

Figure 4 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.

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

Table 2. *Language bias (mode declarations) used in the experiments and some indicative complex event definitions from each domain*

Meeting/Moving:	***Rendezvous:***
head(initiatedAt(move(+person, +person), +time))	head(initiatedAt(rendezVous(+vessel, +vessel), +time))
head(terminatedAt(move(+person, +person), +time))	head(terminatedAt(rendezVous(+vessel, +vessel), +time))
head(initiatedAt(meet(+person, +person), +time))	body(happensAt(slow_motion_start(+vessel), +time))
head(terminatedAt(meet(+person, +person), +time))	body(happensAt(slow_motion_end(+vessel), +time))
body(happensAt(walking(+person), +time))	body(happensAt(stop_start(+vessel), +time))
body(happensAt(running(+person), +time))	body(happensAt(stop_end(+vessel), +time))
body(happensAt(abrupt(+person), +time))	body(proximity(+vessel, +vessel, +time))
body(happensAt(active(+person), +time))	body(not proximity(+vessel, +vessel, +time))
body(happensAt(inactive(+person), +time))	body(happensAt(stopped(+vessel), +time))
body(happensAt(appears(+person), +time))	body(happensAt(slow_motion(+vessel), +time))
body(happensAt(disappears(+person), +time))	body(happensAt(heading_change(+vessel), +time))
body(close(+person, +person, #distance, +time))	body(happensAt(communication_gap(+vessel), +time))
body(not close(+person, +person, #distance, +time))	body(happensAt(sailing_speed(+vessel), +time))
body(orientation(+person, +person, #diff, +time))	body(farFromPorts(+vessel, +time))
body(visible(+person, +time))	
body(not visible(+person, +time))	
Dangerous Driving:	
head(initiatedAt(dangDrive(+vehicle), +time))	head(terminatedAt(dangDrive(+vehicle), +time))
body(happensAt(acceleration(+vehicle), +time))	body(happensAt(cornering(+vehicle), +time))
body(happensAt(braking(+vehicle), +time))	body(happensAt(stop(+vehicle), +time))
body(happensAt(icy_road(+vehicle), +time))	body(happensAt(overspeeding_start(+vehicle), +time))
body(happensAt(overspeeding_end(+vehicle), +time))	body(happensAt(overspeeding(+vehicle), +time))

Table 2. *Continued*

Some indicative domain rules that must be learnt:

$\texttt{initiatedAt}(meet(X,Y),T) \leftarrow$
$\quad \texttt{happensAt}(walking(X),T),$
$\quad \texttt{happensAt}(walking(Y),T),$
$\quad close(X,Y,34,T).$
$\texttt{terminatedAt}(meet(X,Y),T) \leftarrow$
$\quad \texttt{happensAt}(walking(X),T),$
$\quad not\ close(X,Y,30,T).$

$\texttt{initiatedAt}(rendezVous(X,Y),T) \leftarrow$
$\quad \texttt{happensAt}(stopped(X),T),$
$\quad \texttt{happensAt}(slow\_motion(Y),T),$
$\quad proximity(X,Y,T).$
$\quad farFromPorts(X,T),$
$\quad farFromPorts(Y,T),$
$\texttt{terminatedAt}(rendezVous(X,Y),T) \leftarrow$
$\quad \texttt{happensAt}(sailing\_speed(X),T),$
$\quad not\ proximity(X,Y,T).$

$\texttt{initiatedAt}(dangDrive(X,Y),T) \leftarrow$
$\quad \texttt{happensAt}(acceleration(X),T),$
$\quad \texttt{happensAt}(icy\_road(X),T).$
$\texttt{terminatedAt}(dangDrive(X),T) \leftarrow$
$\quad \texttt{happensAt}(overspeeding\_end(X),T).$
$\texttt{terminatedAt}(dangDrive(X),T) \leftarrow$
$\quad \texttt{happensAt}(stop(X),T).$

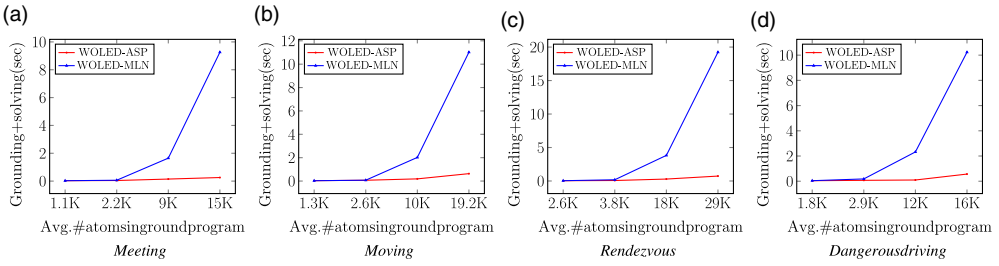

Fig. 4. Scalability of MAP inference.

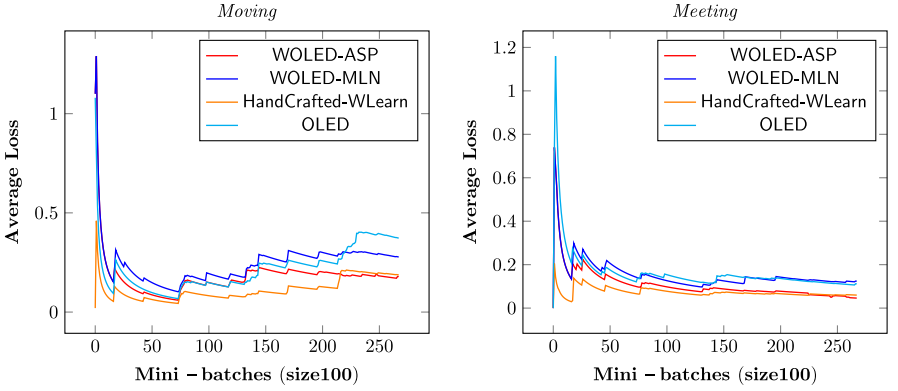

Fig. 5. Prequential Evaluation on CAVIAR.

6.2 Online structure & weight learning performance

In our next experiment we assess WOLED-ASP's predictive performance and efficiency in the task of online structure and weight learning and we compare it to (i) WOLED-MLN; (ii) OLED (Katzouris *et al*. 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* evaluation (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 information 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 from prequential evaluation for *meeting* & *moving* are presented in Figure 5, while Table 3 reports several statistics for the systems being compared: (i) F_1-scores 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 data sets, whose size makes tenfold cross-validation impractical, we used half the data set for training and half for testing, so the reported F_1-scores are obtained from the latter half; (ii) CE pattern set sizes (total number of literals) at the end of a single-pass

Table 3. *Online structure and weight learning results*

	Method	F_1-score (test set)	Theory size	Inference Time (s)	Pred. Compl. Time (s)	Total Time (s)
Moving	WOLED-ASP	**0.821**	26	15	–	112
	WOLED-MLN	0.801	47	187	28	478
	OLED	0.730	**24**	**13**	–	74
	HandCrafted	0.637	28	–	–	–
	HandCrafted-WL	0.702	28	16	–	**52**
Meeting	WOLED-ASP	**0.887**	34	**12**	–	82
	WOLED-MLN	0.841	56	134	12	145
	OLED	0.782	42	10	–	36
	HandCrafted	0.735	**23**	–	–	–
	HandCrafted-WL	0.753	**23**	13	–	**31**
Rendezvous	WOLED-ASP	0.98	18	647	–	4856
	WOLED-MLN	0.98	18	2923	434	6218
	OLED	0.98	18	**623**	–	**4688**
Dang.Drive	WOLED-ASP	0.99	**21**	341	–	2465
	WOLED-MLN	0.99	28	926	287	3882
	OLED	0.99	**21**	**312**	–	**2435**

over a data set; (iii) total inference time at the end of a single-pass over a data set (MAP inference for WOLED-ASP, WOLED-MLN & HandCrafted-WL, crisp logical inference for OLED); (iv) for WOLED-MLN, total time spent on predicate completion; (v) total training time at the end of a single-pass over a data set, which includes time spent on CE pattern generation, computing θ-subsumption, etc., that is, the dominant costs involved in learning CE patterns structure. Note that we report on (iii), (iv), (v) 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 data sets, since in these data sets hand-crafted CE patterns were used to generate the ground truth in the first place. We also omit prequential learning curves for *rendezVous* & *dangerous driving* in Figure 5, since, due to the synthetic ground truth in the maritime & the fleet management data sets, the learning curves for these CEs are very similar for all algorithms being compared and are not informative.

The results in Figure 5 and in Table 3 seem to validate the claim of Section 5 on the differences in predictive performance between WOLED-ASP and WOLED-MLN. Indeed, WOLED-ASP clearly outperforms WOLED-MLN, both in prequential error and in cross-validation F_1-scores, indicating better generalization abilities. Moreover, WOLED-ASP learns simpler CE patterns sets, as shown by the theory size statistic, which seems to validate the claim made right before Section 5. HandCrafted-WL outperforms WOLED-ASP in the prequential task for the most part of the training process. This was expected, since HandCrafted-WL has the advantage of operating on a good set of rules provided beforehand and, therefore, is less prone to erroneous predictions. On the other hand, its inability to learn new rules explains its inferior test-set F_1-scores for *meeting* & *moving*.

OLED aims at quickly discovering a good set of rules. It is not concerned with optimizing their joint predictive performance and does not learn weights. It is the most efficient of all learners, but it is also outperformed by all in terms of prequential error and test set F_1-scores.

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 predicate completion.

6.3 Comparison to batch learners

In our last experiment we compare WOLED-ASP to a number of batch learning algorithms for learning structure and weights. To that end we used smaller data sets whose size makes batch learning practical. In particular, we used a fragment of the CAVIAR data set that has been used in batch learning experiments in previous work (Skarlatidis *et al*. 2015) and a small excerpt of the maritime data set. The fragment CAVIAR data set consists of the regions of the original data set where the two target CEs (*meeting* and *moving*) occur and contains a total of 25,738 training interpretations. The maritime fragment data set contains 11,930 training interpretations corresponding to six data sequences, extracted from the original maritime data set, where *rendezvous* between pairs of vessels occurs.

We compare WOLED-ASP to the following batch learning algorithms: (i) XHAIL (Ray 2009), a non-monotonic ILP learner whose rule induction strategy WOLED-ASP combines with probabilistic reasoning; (ii) ILED (Katzouris *et al*. 2015), an algorithm that combines XHAIL's learning machinery with theory revision, in order to learn incrementally, and has been shown to achieve performance comparable to that of XHAIL's, while being much more efficient; (iii) ILASP (Law *et al*. 2015), a state-of-the art ILP system that learns answer set programs from examples and has been used for inducing complex event patterns (Law *et al*. 2018); (iv) MaxMargin, a batch weight learning algorithm for MLN, introduced in Huynh and Mooney (2009), which has been used with the CAVIAR fragment data set in the past and as been shown to achieve very good results. XHAIL, ILED, and ILASP are crisp learners (i.e. there is no weight learning involved). MaxMargin was used with hand-crafted rule sets and the task was to learn weights for these rules.

WOLED-ASP, XHAIL, ILED, and ILASP are based on ASP and rely on Clingo. The implementation of XHAIL and ILED is available online[10] and so is the most recent version of ILASP (ILASP4)[11]. MaxMargin is available from the LoMRF platform.

The original ILED algorithm is designed for soundness and cannot tolerate noise. To account for that in this experiment we used a noise-tolerant version (denoted by ILED-HC) that learns theories in an iterative hill-climbing process: It first constructs a bottom

[10] https://github.com/nkatzz/ORL
[11] http://www.ilasp.com/download

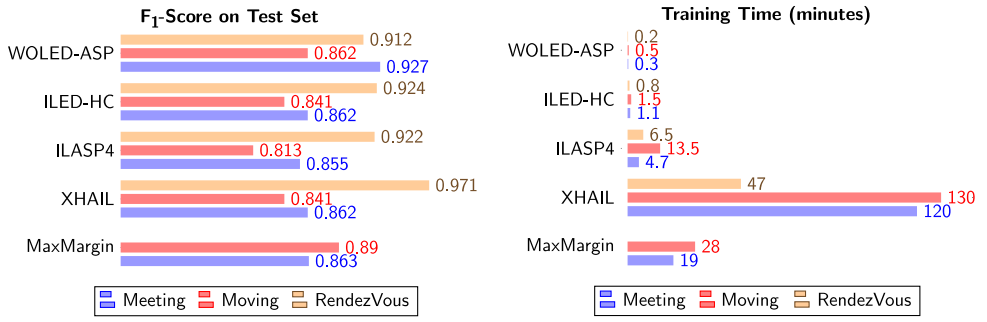

Fig. 6. Comparison with batch learners.

theory (collection of bottom rules) from the mode declarations and then passes once over the data, which are presented in mini-batches, to generate a number of different theories, by generalizing the bottom theory w.r.t. each mini-batch. The theory with the best performance on the training set is retained and is subsequently further revised from each mini-batch in an additional pass over the data. The process continues, keeping the best revision at each iteration, until no improvement in performance is observed, or a max-iterations threshold is reached. In these experiments we used batch size of 100 time points with ILED-HC, from which the algorithm converged in approximately 5–7 iterations over the data for *meeting* and *moving* and 3 iterations for *rendezvous*.

We used tenfold cross-validation process for *meeting* and *moving* and sixfold cross-validation for *rendezVous*. The results are presented in Figure 6 in the form of (micro-averaged) F_1-scores from the testing sets and average training times for each algorithm. We omit results for MaxMargin on *rendezVous*, since the hand-crafted rules that MaxMargin would learn weights for are those that were used to generate the ground truth.

WOLED-ASP achieves the best F_1-score for *meeting*, with a significant distance from the other algorithms. It also achieves the second-best F_1-score for *moving*. In *rendezVous*, XHAIL is a clear winner w.r.t. predictive performance, while all other algorithms achieve comparable F_1-scores. WOLED-ASP is significantly faster than ILASP, XHAIL and MaxMargin, while its efficiency is comparable to that of ILED-HC's, which, however, is outperformed by WOLED-ASP.

To appreciate the differences in performance between the algorithms being compared, it is helpful to take a look into their inner workings. XHAIL generalizes a bottom theory from the entirety of the training data in one go, which explains its increased training times. On the other hand, it is thanks to this strategy that XHAIL is capable to learn better theories for *rendezVous* than algorithms that process the training examples individually, thus failing to discover fragments of the *rendezVous* definition, whose utility is revealed only when "looking" at the training data as a whole. In contrast to the *rendezVous* case where the ground truth is synthetic, XHAIL's learning strategy seems less useful in CAVIAR, where XHAIL marginally outperforms ILASP at the cost of much higher training times, it achieves identical performance to ILED and is outperformed by WOLED-ASP on both *meeting* and *moving* and by MaxMargin on *moving*. This latter observation is an indication for the merit of weight learning.

In contrast to XHAIL, ILED-HC, and WOLED-ASP, which generate rules on demand in a data-driven fashion, ILASP4 explicitly enumerates its search space. Even by taking into

account what is referred to as "commonsense constraints" in Law *et al.* (2018), that is, forbidding useless rules from being generated (e.g. rules stating that two persons/vessels are close and far at the same time), This yields large search spaces, of approximately 3400 rules for *meeting*, 10,000 rules for *moving,* and 12,000 rules for *rendezVous*, which explains ILASP's increased training times, as compared to those of WOLED-ASP and ILED-HC's. On the other hand, given a search space, ILASP4 is guaranteed to find an optimal hypothesis therein, that is, a theory that minimizes training error and model complexity, while processing the data incrementally. XHAIL also comes with optimality guarantees, which, however, are subject to the quality of the bottom theory that is used to structure its search space, while WOLED-ASP and ILED-HC use heuristic search procedures with no guarantees on the quality of the learnt hypotheses. ILASP4's inferior predictive performance is attributed to the learning setting, which for ILASP4 follows closely the one reported in Law *et al.* (2018) and aims at keeping learning tractable: first, to limit the number of irrelevant answer sets generated during learning, additional constraints on ILASP4's search space dictate that the learnt event patterns should only account for the "turning points" in a fluent's truth value. That is, initiation rules for a fluent f are learnt from data points t, such that f does not hold at t and holds at $t+1$. Similarly, for termination rules, f should hold at t and not hold at $t+1$. Second, since such turning points in fluents' truth values are too few in the data sets, as compared to points where fluents hold/do not hold continuously, the turning point examples are weighted[12], reflecting their increased importance and simulating the effect of "over-sampling" such examples. This allows for efficient learning with ILASP4 in this domain. The downside is that learning becomes very susceptible to the even the slightest noise in the turning point examples, which explains ILASP4's inferior performance, as compared to the other algorithms.

We conclude this section by pointing out that XHAIL and ILASP4 are more general-purpose learners than the event-based algorithms they were compared to. Therefore, they may be outperformed by the latter on the particular task of learning event definitions, but on the other hand, they may be used for learning tasks that the event-based algorithms cannot.

7 Conclusions and future work

We presented an online algorithm for learning weighted Event Calculus rules. Our system is entirely implemented in ASP 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 data sets indicates that it compares favorably to state-of-the art online and batch learners. Future work involves combination with semi-supervised learning, toward handling the scarcity of labeled data in streaming settings.

Acknowledgments

This work is supported by the project entitled "INFORE: Interactive Extreme-Scale Analytics and Forecasting", funded by the EUs Horizon 2020 research and innovation pro-

[12] The weight is translated to a cost for hypotheses that do not correctly account for these turning point examples.

gramme under grant agreement No 825070 and by the project SYNTELESIS "Innovative Technologies and Applications based on the Internet of Things and Cloud Computing" (MIS 5002521), which is funded by the Operational Programme "Competitiveness, Entrepreneurship and Innovation" (NSRF 2014-2020) and co-financed by Greece and the EU. We would like to thank Mark Law for his assistance in running ILASP.

References

Alevizos, E., Skarlatidis, A., Artikis, A. and Paliouras, G. 2017. Probabilistic complex event recognition: A survey. *ACM Computing Surveys 50,* 5, 71:1–71:31.

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

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

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. 2008. *Logical and Relational Learning*. Springer Science & Business Media.

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.

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.

Guimarães, V., Paes, A. and Zaverucha, G. 2019. Online probabilistic theory revision from examples with proppr. *Machine Learning 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*. SIAM, 642–651.

Katzouris, N. 2017. Scalable relational learning for event recognition. *PhD Thesis, University of Athens.* http://users.iit.demokritos.gr/ nkatz/papers/nkatz-phd.pdf

Katzouris, N. and Artikis, A. 2020. WOLED: A tool for online learning weighted answer set rules for temporal reasoning under uncertainty. In *KR 2020*.

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., Artikis, A. and Paliouras, G. 2019. Parallel online event calculus learning for complex event recognition. *Future Generation Computer Systems 94*, 468–478.

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. 2015. The ILASP system for learning answer set programs. www.ilasp.com.

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

Lee, J., Talsania, S. and Wang, Y. 2017. Computing LPMLN using ASP and MLN solvers. *Theory and Practice of Logic Programming 17,* 5–6, 942–960.

Lee, J. and Wang, Y. 2016. Weighted rules under the stable model semantics. In *KR, 2016*.

Lifschitz, V. 2019. *Answer Set Programming*. Springer.

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

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.