TLP **16** (5–6): 817–833, 2016. © Cambridge University Press 2016
doi:10.1017/S1471068416000260

Online learning of event definitions

NIKOS KATZOURIS[1,3], ALEXANDER ARTIKIS[2,3] and GEORGIOS PALIOURAS[3]
[1]*Department of Informatics & Telecommunications, National Kapodistrian University of Athens, Athens, Greece*
[2]*Department of Maritime Studies, University of Piraeus, Piraeus, Greece*
[3]*Institute of Informatics & Telecommunications, National Center for Scientific Research "Demokritos", Athens, Greece*
(*e-mail:* {nkatz,a.artikis,paliourg}@iit.demokritos.gr)

submitted 6 May 2016; revised 8 July 2016; accepted 22 August 2016

Abstract

Systems for symbolic event recognition infer occurrences of events in time using a set of event definitions in the form of first-order rules. The Event Calculus is a temporal logic that has been used as a basis in event recognition applications, providing among others, direct connections to machine learning, via Inductive Logic Programming (ILP). We present an ILP system for online learning of Event Calculus theories. To allow for a single-pass learning strategy, we use the Hoeffding bound for evaluating clauses on a subset of the input stream. We employ a decoupling scheme of the Event Calculus axioms during the learning process, that allows to learn each clause in isolation. Moreover, we use abductive-inductive logic programming techniques to handle unobserved target predicates. We evaluate our approach on an activity recognition application and compare it to a number of batch learning techniques. We obtain results of comparable predicative accuracy with significant speed-ups in training time. We also outperform hand-crafted rules and match the performance of a sound incremental learner that can only operate on noise-free datasets.

KEYWORDS: Inductive Logic Programming, Event Calculus, Online Learning

1 Introduction

Event recognition systems (Etzion and Niblett 2010) process sequences of *simple events*, such as sensor data, and recognize *complex events* of interest, i.e. events that satisfy some pattern. Logic-based event recognition typically uses a knowledge base of first-order rules to represent complex event patterns and a reasoning engine to detect such patterns in the incoming data. Dialects of the Event Calculus (EC) (Kowalski and Sergot 1986) have been used as a language for specifying definitions of complex events (Artikis *et al.* 2015). An advantage of this approach is that is offers direct connections to machine learning, via Inductive Logic Programming (ILP) (De Raedt 2008), alleviating the task of manual authoring of event definitions.

Event recognition applications deal with noisy data streams. Methods that extract insights from such streams need to operate within tight memory and time constraints, building a decision model by a single pass over the training data (Gama and Gaber 2007; Gama 2010). Such a framework is under-explored in ILP, where all data is typically in place when learning begins. Alternatively, some ILP systems are capable of theory revision (Esposito *et al.* 2000). Still, such systems need multiple scans of the data to optimize their theories.

We present OLED (Online Learning of Event Definitions), an ILP system that learns EC theories in a single pass over a data stream. OLED uses the Hoeffding bound (Hoeffding 1963), a statistical tool that allows to build decision models using only a small subset of the data, by relating the size of this subset to a user-defined confidence level on the error margin of not making a (globally) optimal decision (Dhurandhar and Dobra 2012; Domingos and Hulten 2000; Gama *et al.* 2011). OLED learns a clause in a top-down fashion, by gradually adding literals to its body. Instead of evaluating each candidate specialization on the entire input, it accumulates training data from the stream, until the Hoeffding bound allows to select the best specialization. The instances used to make this decision are not stored or reprocessed, but discarded as soon as OLED extracts from them the necessary statistics for clause evaluation.

In the learning problem we address in this work, target clauses are not unrelated, but depend on each other via the axioms of the EC, making it difficult to use common ILP practices that learn clauses in isolation. To handle this issue we use a decoupling scheme of the axioms of the EC during learning, thereby allowing to assess the quality of each clause separately, using a scoring function. Additionally, learning programs in the EC involves *non-Observational Predicate Learning* (non-OPL) (Muggleton 1995), a setting where instances of the target predicates are not directly observable in the data. To handle non-OPL we use abduction (Denecker and Kakas 2002), a framework that may be used for reasoning with incomplete information. We evaluate our approach on an activity recognition application and compare it to a number of batch learning techniques. We obtain results of comparable predicative accuracy with significant speed-ups in training time. We also outperform hand-crafted rules and match the performance of a sound incremental learner that can only operate on noise-free datasets.

The rest of this paper is structured as follows: In Section 2 we discuss related work, while in Section 3 we present some necessary background on the EC, ILP and the Hoeffding bound. In Section 4 we present OLED and in Section 5 we show the results of the empirical analysis. Finally, in Section 6 we discuss some directions for future research and conclude.

2 Related work

The Hoeffding bound has been used for propositional machine learning tasks on data streams, such as learning decision trees (Domingos and Hulten 2000) and decision rules (Gama *et al.* 2011), and clustering (Domingos and Hulten 2001). However, its usage for learning relational models is limited. One reason is that it requires independence of observations, which cannot always be ensured in relational domains, due to dependencies in the data (Jensen 1999; Jensen and Neville 2002;

Hulten *et al.* 2003; Dhurandhar and Dobra 2012). An ILP approach that uses the Hoeffding bound for relational learning is HTILDE (Lopes and Zaverucha 2009), an extension of the TILDE system for learning first-order decision trees (Blockeel and De Raedt 1998). These are decision trees where each internal node consists of a conjunction of literals and each leaf is a propositional predicate representing a class. TILDE constructs trees by testing conjunctions of literals at each node, using an ILP refinement operator to generate the conjunctions and information gain as the guiding heuristic. HTILDE extends TILDE by using the Hoeffding bound to perform these internal tests on a subset of the training data. To ensure independence of observations, HTILDE learns from interpretations (Blockeel *et al.* 1999), a setting, used also by OLED, where each training instance is assumed a disconnected part of the dataset.

Like TILDE, HTILDE learns clauses with a propositional predicate in the head (representing a class). However, the head of a complex event definition is typically a first-order predicate, containing variables that appear in the body of the clause and express relations between entities. Therefore, HTILDE is not general enough for the problem we address in this work. Additionally, HTILDE requires a fully annotated dataset, while in the setting we assume here, annotation for target predicates is missing.

Learning programs in the EC is a challenging task that most ILP learners cannot fully undertake (Ray 2009; Katzouris *et al.* 2015), mainly due to the non-monotonicity of negation as failure (NaF) that the EC uses. XHAIL (Ray 2009) and TAL/ASPAL/RASPAL (Athakravi *et al.* 2013) are systems that can handle the task, by combining ILP with the non-monotonic semantics of abductive logic programming. These approaches ensure soundness of the outcome, which in the presence of NaF requires learning whole theories by jointly optimizing their clauses. This implies an intractable search space, even with relatively small amounts of data. As a result, the aforementioned approaches do not scale to event recognition applications with temporal data streams. In contrast, OLED learns clauses separately using only fragments of the data in an online setting, trading soundness for efficiency.

ILED (Katzouris *et al.* 2015) is a recently proposed scalable extension of the XHAIL system that is able to learn EC theories. It is an incremental learner that revises past hypotheses to fit new observations, and a full-memory system, meaning that revisions should account for a growing historical memory of accumulated data. Using a compressive memory structure to encode the positive examples that each clause entails in the historical memory, ILED requires at most one pass over the past data to revise a hypothesis. One difference from OLED is that the latter learns in an online fashion, thus it does not re-process past examples. Also, ILED is designed to learn sound theories and a key assumption for its scalable strategy is that the training data is noise-free. Other incremental ILP systems, such as INTHELEX (Esposito *et al.* 2000) and FORTE (Richards and Mooney 1995), cannot be applied to the task we address in this work, since they cannot handle negation (FORTE) and non-observable target predicates (INTHELEX, FORTE).

Table 1. *The basic predicates and domain-independent axioms of the* EC *dialect.*

Predicate	Predicate Meaning	Axioms
happensAt(E, T)	Event E occurs at time T	holdsAt$(F, T+1) \leftarrow$ initiatedAt(F, T). (1)
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$(F, T+1) \leftarrow$ holdsAt(F, T), not terminatedAt(F, T). (2)
holdsAt(F, T)	Fluent F holds at time T	

3 Background and Running Example

We assume a logic programming setting, where predicates, terms, atoms, literals, clauses and programs (theories) are defined as in (Gebser *et al.* 2012) and not denotes NaF. Following Prolog's convention, predicates and ground terms in logical formulae start with a lower case letter, while variable terms start with a capital letter.

The Event Calculus (EC) (Kowalski and Sergot 1986) is a temporal logic for reasoning about events and their effects. Its ontology comprises time points, represented by 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. The axioms of the EC incorporate the *common sense law of inertia*, according to which fluents persist over time, unless they are affected by an event. We use a simplified version of the EC that has been shown to suffice for event recognition (Artikis *et al.* 2015). The basic predicates and its *domain-independent* axioms are presented in Table 1. 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 by a set of *domain-specific* axioms. To illustrate our learning approach we use the task of activity recognition, as defined in the CAVIAR project[1]. 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 complex events, consists of activities that involve more than one person, for instance two people *moving together, meeting each other, fighting* and so on. The aim is to recognize complex events by means of combinations of simple events and some additional domain knowledge, such as a person's position and direction.

Table 2(a) presents an example of CAVIAR data, consisting of a narrative of simple events in terms of happensAt/2, expressing the short-term activities of people, and context properties in terms of holdsAt/2, denoting the coordinates and direction

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

Table 2. ***(a)*** *Example data from activity recognition. For instance, at time point 1 person* id_1 *is* walking, *her* (x, y) *coordinates are* (201, 454) *and her direction is* 270°. *The annotation for the same time point states that persons* id_1 *and* id_2 *are not moving together, in contrast to the annotation for time point 2.* ***(b)*** *An example of two domain-specific axioms in the* EC*. The first clause dictates that* moving *of 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 *and their difference in direction is less than* 45°. *The second clause dictates that* moving *of* X *and* Y *is terminated at time* T *if one of them is standing still at time* T *(exhibits an inactive behavior) and their euclidean distance at* T *is greater that 30.*

(a)		**(b)**
Narrative for time 1:	**Narrative for time 2:**	**Two Domain-specific axioms:**
happensAt($walking(id_1), 1$)	happensAt($walking(id_1), 2$)	initiatedAt($moving(X, Y), T$) $\leftarrow$
happensAt($walking(id_2), 1$)	happensAt($walking(id_2), 2$)	happensAt($walking(X), T$),
holdsAt($coords(id_1, 201, 454), 1$)	holdsAt($coords(id_1, 201, 454), 2$)	happensAt($walking(Y), T$),
holdsAt($coords(id_2, 230, 440), 1$)	holdsAt($coords(id_2, 227, 440), 2$)	$distanceLessThan(X, Y, 25, T)$,
holdsAt($direction(id_1, 270), 1$)	holdsAt($direction(id_1, 275), 2$)	$directionLessThan(X, Y, 45, T)$.
holdsAt($direction(id_2, 270), 1$)	holdsAt($direction(id_2, 278), 2$)	
Annotation for time 1:	**Annotation for time 2:**	terminatedAt($moving(X, Y), T$) $\leftarrow$
not holdsAt($moving(id_1, id_2), 1$)	holdsAt($moving(id_1, id_2), 2$)	happensAt($inactive(X), T$),
		$distanceMoreThan(X, Y, 30, T)$.

of the people. Table 2(a) also shows the annotation of complex events (long-term activities) for each time-point in the narrative. The annotation about complex events is obtained via the closed world assumption (we state both positive and negated annotation atoms in Table 2 to avoid confusion). An example of two domain-specific axioms in the EC is presented in Table 2(b).

Our goal is to learn a set of domain-specific axioms specifying complex events. Inductive Logic Programming (ILP) (De Raedt 2008) provides techniques for learning logical theories from examples. In the *Learning from Interpretations* (*LfI*) (Blockeel *et al.* 1999) setting that we use in this work, each training example is an interpretation, i.e. a set of true ground atoms, as in Table 2(a). Given a set of training interpretations $\mathscr{I}$ and some background theory B, which in our case consists of the domain-independent axioms of the EC, the goal in *LfI* is to find a theory H, such that for each interpretation $I \in \mathscr{I}$, $B \cup H$ *covers* I, i.e. I is a model of $B \cup H$. Although different semantics are possible, in this work a "model" is an *answer set* (Gebser *et al.* 2012).

To allow for an online learning setting, we use the Hoeffding bound (Hoeffding 1963), a statistical tool that may be used as a probabilistic estimator of the generalization error of a model (true expected error on the entire input), given its empirical error (observed error on a training subset) (Dhurandhar and Dobra 2012). Given a random variable X with range in $[0, 1]$ and an observed mean $\overline{X}$ of its values after n independent observations, the Hoeffding Bound states that, with probability $1 - \delta$, the true mean $\hat{X}$ of the variable lies in an interval $(\overline{X} - \epsilon, \overline{X} + \epsilon)$, where $\epsilon = \sqrt{\frac{ln(1/\delta)}{2n}}$. In other words, the true average can be approximated by the observed one with probability $1 - \delta$, given an error margin ϵ that decreases with the number of observations n.

4 Online Learning of Event Definitions

ILP learners typically employ a separate-and-conquer strategy: clauses that cover subsets of the examples are constructed one by one recursively, until all examples are covered. Each clause is constructed in a top-down fashion, starting from an overly general clause and gradually specializing it by adding literals to its body. The process is guided by a heuristic function G that assesses the quality of each specialization on the entire training set. At each step, the literal (or set of literals) with the optimal G-score is selected and the process continues until certain criteria are met. To adapt this strategy to an online setting, we use the Hoeffding bound to evaluate candidate specializations on a subset of the training interpretations, instead of evaluating them on the entire input. To do so, we use an argument adapted from (Domingos and Hulten 2000). Let r be a clause and G a clause evaluation function with range in $[0,1]$. The evaluation function that we use in this work will be discussed shortly. Assume also that after n training instances, r_1 is r's specialization with the highest observed mean G-score $\overline{G}$ and r_2 is the second best one, i.e. $\Delta\overline{G} = \overline{G}(r_1) - \overline{G}(r_2) > 0$. Then by the Hoeffding bound we have that for the true mean of the scores' difference $\Delta\hat{G}$ it holds $\Delta\hat{G} > \Delta\overline{G} - \epsilon$, with probability $1-\delta$, where $\epsilon = \sqrt{\frac{ln(1/\delta)}{2n}}$. Hence, if $\Delta\overline{G} > \epsilon$ then $\Delta\hat{G} > 0$, implying that r_1 is indeed the best specialization to select at this point, with probability $1-\delta$. In order to decide which specialization to select, it thus suffices to accumulate observations from the input stream until $\Delta\overline{G} > \epsilon$. Since ϵ decreases with the number of observations, given a desired δ, the number of observations n needed to reach a decision may be traded for a tolerable generalization error ϵ of not selecting the optimal specialization at a certain choice point. The observations need not be stored or reprocessed. We process each observation once to extract the necessary statistics for the computation of the G-score of each candidate specialization. This gives rise to a single-pass clause construction strategy.

In *LfI* each interpretation is independent form others (Blockeel *et al.* 1999). This guarantees the independence of observations that is necessary for using the Hoeffding bound. In our setting, an interpretation consists of ground atoms I known true at two consecutive time points T and $T+1$, as in Table 2(a). In our EC dialect, the initiation/termination of complex events depends only on the simple events and contextual information of the previous time-point, therefore each interpretation is an independent training instance.

4.1 Evaluating Clauses

We relax the *LfI* requirement that a hypothesis H covers every training interpretation to account for noise, and thus seek for a theory with a good fit in the training data. To this end, we define true positive, false positive and false negative atoms as follows:

Definition 1 (*TP, FP, FN atoms*)

Let B consist of the domain-independent EC axioms, r be a clause and I an interpretation. We denote by *narrative*(I) and *annotation*(I) the narrative and the annotation part of I respectively (see also Table 2(a)). We denote by M_I^r an answer set of $B \cup narrative(I) \cup r$. Given an annotation atom α we say that:

- α is a true positive (TP) atom w.r.t. clause r iff $\alpha \in annotation(I) \cap M_I^r$.
- α is a false positive (FP) atom w.r.t. clause r iff $\alpha \in M_I^r$ but $\alpha \notin annotation(I)$.
- α is a false negative (FN) atom w.r.t. clause r , iff $\alpha \in annotation(I)$ but $\alpha \notin M_I^r$.

□

We seek a theory H that maximizes the TP atoms, while minimizing the FP and FN atoms, collectively for all its clauses. To do so, we maintain a count per clause for each such atom. For an initiatedAt clause, its TP (resp. FP) count is increased each time it correctly (resp. incorrectly) initiates a complex event (according to the annotation). For a terminatedAt clause, its TP count is increased each time it correctly allows a complex event to persist, by not terminating it. Its FN count is increased when it incorrectly terminates a complex event.

When learning structure in Horn (negation-free) logic with ILP, a theory H is augmented with new clauses to increase its total TP count, while existing clauses in H are specialized to decrease the FP count. This strategy is not directly applicable to the problem at hand. When learning programs in the EC, the addition of new clauses may be necessary to eliminate FPs, while clause specialization may be necessary to increase TPs, as we explain below. Given a theory H and interpretation I, assume that $B \cup H$ does not cover I. Then one of the following holds:

1. **The *FN* case.** There is at least one FN atom α. This may be due to one of the following:
   (a) No initiatedAt clause in H "fires", failing to initiate the complex event that corresponds to α, when it should. In this case, generating a new initiatedAt clause, eliminates the FN atom, turning it into a TP.
   (b) One or more terminatedAt clauses in H are over-general, terminating the complex event that corresponds to α when they should not. Specializing the over-general clauses, eliminates the FN atom, turning it into a TP.
2. **The *FP* case.** There is at least one FP atom α. This may be due to one of the following:
   (a) No terminatedAt clause in H "fires", failing to terminate the complex event that corresponds to α when it should, so α erroneously persists by inertia. Generating a new terminatedAt clause eliminates the FP.
   (b) One or more initiatedAt clauses are over-general, re-initiating a corresponding complex event when they should not. Specializing the over-general clauses eliminates the FP.

Given the different possible behaviours of initiation and termination clauses in the process of optimizing a theory H, we next define the clause evaluation function.

Definition 2 (*Clause evaluation function*)
Let us denote by TP_r, FP_r and FN_r respectively, the accumulated TP, FP and FN counts of clause r over the input stream. The clause evaluation function G for a clause r is a function with range in $[0, 1]$ defined as follows:

Table 3. *Action dispatching scheme for OLED's initiatedAt (L_{init}) and terminatedAt (L_{term}) parallel processes. The justification refers to the different cases analysed in Section 4.1*

Process	Cause of Failure	Action	Justification
L_{init}	*FP*	Clause expansion	Case 2(b)
L_{init}	*FN*	Theory expansion	Case 1(a)
L_{term}	*FP*	Theory expansion	Case 2(a)
L_{term}	*FN*	Clause expansion	Case 1(b)

$$G(r) = \begin{cases} \frac{TP_r}{TP_r+FP_r} & \text{if } r \text{ is an initiatedAt clause} \\ \frac{TP_r}{TP_r+FN_r} & \text{if } r \text{ is a terminatedAt clause} \end{cases} \qquad \square$$

Both initiatedAt and terminatedAt clauses affect the total *TP* count of a theory *H*, therefore *TP* counts per clause are taken into account for the evaluation of both types of clauses. Additionally, specializing existing clauses further improves the quality of *H* by eliminating *FPs* in the initiatedAt case (case 2(b) above) and *FNs* in favor of *TPs* in the terminatedAt case (case 1(b)). Therefore, *FPs* (resp. *FNs*) should also be taken into account when evaluating initiatedAt (resp. terminatedAt) clauses. On the other hand, the total *FP* (resp. *FN*) count of a theory *H* is not affected by its *existing* terminatedAt (resp. initiatedAt) clauses, but instead requires new clauses to be generated (cases 2(a) and 1(a) respectively). Therefore, *FPs* and *FNs* are irrelevant for the evaluation of existing terminatedAt and initiatedAt clauses respectively. Combining these observations we derive the scoring function of Definition 2, that uses precision and recall for initiatedAt and terminatedAt clauses respectively.

4.2 *The OLED system*

In this section we discuss the main functionality of OLED, presented in Algorithm 1, in detail. Learning begins with an empty hypothesis *H*. On the arrival of new interpretations, OLED either expands *H*, by generating a new clause, or tries to expand (specialize) an existing clause. Clauses of low quality are pruned, after they have been evaluated on a sufficient number of examples. Each incoming interpretation is processed once, to extract the necessary statistics for clause evaluation in the form of *TP*, *FP* and *FN* counts, and is subsequently discarded.

To distinguish between the different cases presented in Section 4.1, initiation and termination clauses are learnt separately in parallel, by two processes L_{init} and L_{term} respectively (each one of these processes runs separately Algorithm 1). The input stream is forwarded to each of these processes simultaneously. Thanks to this decoupling, when either process fails to account for a training interpretation, it is able to infer the causes of failure in terms of *FP* and *FN* atoms. In particular L_{init} detects *FP*/*FN*-failures based on cases 2(b)/1(a) respectively and L_{term} detects *FP*/*FN*-failures based on cases 2(a)/1(b). According to the cause of failure, the process dispatches control either to the theory expansion, or the clause expansion subroutines. The choice among these actions is made by the boolean function

Algorithm 1 OnlineLearning($\mathscr{I}, B, G, \delta, d, N_{min}, S_{min}$)

Input: $\mathscr{I}$: A stream of training interpretations; B: Background knowledge; G: Clause evaluation function; δ : Confidence for the Hoeffding test; d : Specialization depth; S_{min} : Clause G-score quality threshold.

```
1: $H := \emptyset$
2: **for each** $I \in \mathscr{I}$ **do**
3:     Update $TP_r, FP_r, FN_r$ and $N_r$ counts from $I$, for each $r \in H$ and each $r' \in \rho_d(r)$,
       where $N_r$ denotes the number of examples on which $r$ has been evaluated so far.
4:     **if** ExpandTheory($B, H, I$) **then**
5:         $H \leftarrow H \cup$ StartNewClause($B, I$)
6:     **else**
7:         **for each** clause $r \in H$ **do**
8:             $r \leftarrow$ ExpandClause($r, G, \delta$)
9:     $H \leftarrow$ Prune($H, S_{min}$)
10: **return** $H$
11: **function** StartNewClause($B, I$):
12:     Generate a bottom clause $\bot$ from $I$ and $B$
13:     $r := head(\bot) \leftarrow$
14:     $\bot_r := \bot$
15:     $N_r = FP_r = TP_r = FN_r := 0$
16:     **return** $r$
17: **function** ExpandClause($r, G, \delta$):
18:     Compute $\epsilon = \sqrt{\frac{ln(1/\delta)}{2N_r}}$ and let $\overline{G}$ denote the mean value of a clause's $G$-score
19:     Let $r_1$ be the best specialization of $r$, $r_2$ the second best and $\Delta\overline{G} = \overline{G}(r_1) - \overline{G}(r_2)$
20:     Let $\tau$ equal the mean value of $\epsilon$ observed so far
21:     **if** $\overline{G}(r_1) > \overline{G}(r)$ **and** $[\Delta\overline{G} > \epsilon$ **or** $\tau < \epsilon]$:
22:         $\bot_{r_1} := \bot_r$
23:         **return** $r_1$
24:     **else return** $r$
25: **function** prune($H, S_{min}$):
26:     Remove from $H$ each clause $r$ for which $S_{min} - \overline{G}(r) > \epsilon$, where $\epsilon$ is the current
    Hoeffding bound
27:     **return** $H$
```

ExpandTheory in line 4 of Algorithm 1. Action selection is based on the analysis of Section 4.1 and summarised in Table 3. Below we present an example for illustration purposes.

Example Initially, processes L_{init} and L_{term} start with two empty hypotheses, H_{init} and H_{term}. Assume that the annotation in one of the incoming interpretations dictates that the *moving* complex event holds at time 10, while it does not hold at time 9. Since no clause in H_{init} yet exists to initiate *moving* at time 9, L_{init} detects the *moving* instance at time 10 as an *FN* and proceeds to theory expansion (second case in Table 3), generating an initiation clause for *moving*. L_{term} is not concerned with initiation conditions, so it will take no actions in this case. Then, a new interpretation arrives, where the annotation dictates that *moving* holds at time 20, but does not hold at time 21. In this case, since no clause yet exists in H_{term} to terminate *moving* at time 20, L_{term} will detect an *FP* instance at time 21. It will then proceed to theory expansion (third case in Table 3), generating a new termination condition for moving. At the same time, assume that the initiation clause in H_{init} is over-general and erroneously re-initiates *moving* at time 20, generating an FP instance for the L_{init} process at time 21. In response to that, L_{init} will proceed to clause expansion (first case in Table 3), penalizing the over-general initiation clause by increasing its FP count, thus contributing towards its potential replacement by one of its specializations. □

In the remainder of this section, we go into the details of theory and clause expansion, as well as some other aspects of OLED's functionality.

Theory Expansion. The theory expansion process is handled by the `StartNewClause` function in Algorithm 1. A new clause is generated in a data-driven fashion, by constructing a *bottom clause* $\bot$ from a training interpretation (Muggleton 1995). Theory expansion consists of the addition of the clause $r = head(\bot) \leftarrow$ to theory H. From that point on, r is gradually specialized by the addition of literals from $\bot$ to its body. We denote by $\bot_r$ the bottom clause associated to clause r.

In a typical ILP setting, a bottom clause is constructed by selecting a target predicate instance e as a "seed", placing it in the head of a newly generated clause $\bot$ with an empty body. A set of atoms that follow deductively from e and the background knowledge are placed in the body of $\bot$. Constants in $\bot$ are replaced with variables, where appropriate, as indicated by a particular language bias, typically *mode declarations* (Muggleton 1995). To find a clause with a good fit in the data, a *refinement operator* ρ is used to generate candidate clauses that θ-subsume $\bot$.

The aforementioned approach cannot be applied directly to the problem we address here, which falls in the non-Observational Predicate Learning (OPL) class of problems (Muggleton 1995). In non-OPL, instances of target predicates, that are normally used as seeds for the construction of $\bot$, are not directly observable in the training data. In our case, target predicates are $\mathsf{initiatedAt}/2$ and $\mathsf{terminatedAt}/2$, while the annotation in the training interpretations consists of complex event instances in terms of the $\mathsf{holdsAt}/2$ predicate (see Table 2). A workaround is to use abduction to obtain the missing target predicate instances and then construct bottom clauses from them. This approach is followed by the XHAIL system (Ray 2009) and we also adopt it here. Like XHAIL, OLED also uses mode declarations for specifying the language bias.

OLED may output the hypothesis constructed so far at any time during the learning process. We allow a "warm-up" period, in the form of a minimum number of training instances N_{min} on which a clause r must be evaluated before it can be included in an output hypothesis.

Clause Expansion. We use the Hoeffding bound to select among competing specializations of a clause r. These specializations are generated by adding one or more literals from $\bot_r$ to the body of r. An input parameter d for *specialization depth* serves as an upper bound to the number of literals that may be added at each time. We use $\rho_d(r)$ to denote the set of specializations for clause r. Formally, $\rho_d(r) = \{head(r) \leftarrow body(r) \wedge D \mid D \subset body(\bot_r) \text{ and } |D| \leqslant d\}$. E.g. $\rho_1(r)$ consists of all "one-step" specializations of r (i.e. those that result by the addition of a single literal from $\bot_r$), while $\rho_2(r)$ consists of $\rho_1(r)$ plus all "two-step" specializations, and so on.

A clause r is expanded, i.e. replaced by its best-scoring specialization from $\rho_d(r)$, when a sufficient number of interpretations have been seen, such that $\Delta\overline{G} > \epsilon$, where $\Delta\overline{G}$ is the observed difference between the mean G-scores of r's best and second best specializations. To ensure that no clause r is replaced by a specialization of lower quality, r itself is also considered as a potential candidate along with its specializations from $\rho_d(r)$. This ensures that expanding a clause to its best-scoring specialization is better, with probability $1 - \delta$, than not expanding it at all.

Online learners are typically subject to order effects, i.e. they are sensitive to the order in which the examples are presented. Using the Hoeffding bound allows OLED to mitigate such effects, since clause expansion is postponed until sufficient evidence for the quality of the candidate specializations is provided by the data.

Tie-breaking. When the scores of two or more specializations are very similar, a large number of training instances may be required to decide between them. This could be wasteful, since any one of the specializations may be chosen. In such cases, as in (Domingos and Hulten 2000), we break ties as follows: Instead of waiting until $\Delta\overline{G} > \epsilon$, as required by the Hoeffding bound-based heuristic, we expand r to its best-scoring specialization if $\Delta\overline{G} < \epsilon < \tau$, where τ is a tie-breaking threshold (recall that ϵ decreases with the number of training examples, thus it may fall below τ). We follow (Yang and Fong 2011) and use an adaptive tie-breaking threshold, set to the mean value of ϵ that has been observed so far in the training process (see line 20, Algorithm 1). In the case of a tie between r itself and its best-scoring specialization, we follow a conservative approach and do not expand r.

Clause pruning. OLED supports removal of clauses whose score is smaller than a quality threshold S_{min}. To decide when a clause may be removed we also use the Hoeffding bound. If $S_{min} - \overline{G}(r) > \epsilon$, then with probability $1-\delta$, the true mean of r's G-score is lower than the quality threshold S_{min} and therefore r should be removed.

5 Experimental Evaluation

We evaluate OLED's performance on CAVIAR (see Section 3), a benchmark dataset for activity recognition. CAVIAR contains a total of 282067 training interpretations with a mean size of 25 atoms each. The size of the search space (clause subsumption lattice) is determined by the size of bottom clauses, which in these experiments consisted on average of 15 literals each. All experiments were conducted on a Linux machine with a 3.6GHz processor (4 cores and 8 threads) and 16GiB of RAM. The code and data are available online[2].

5.1 Comparison with Manually Constructed Rules and Batch Learning

The purpose of this experiment was to assess whether OLED is able to efficiently learn theories of comparable quality to hand-crafted rules and state-of-the-art batch

[2] `https://github.com/nkatzz/OLED`

Table 4. *Experimental results from the CAVIAR dataset*

		Method	Precision	Recall	F_1-score	Theory size	Time (sec)
(a)	*Moving*	EC_{crisp}	**0.909**	0.634	0.751	28	–
		EC_{MM}	0.844	0.941	**0.890**	28	1692
		XHAIL	0.779	0.914	0.841	**14**	7836
		OLED	0.709	**0.948**	0.812	34	**12**
	Meeting	EC_{crisp}	0.687	0.855	0.762	23	–
		EC_{MM}	0.919	0.813	**0.863**	23	1133
		XHAIL	0.804	**0.927**	0.861	**15**	7248
		OLED	**0.943**	0.750	0.836	29	**23**
(b)	*Moving*	EC_{crisp}	**0.721**	0.639	0.677	28	–
		OLED	0.653	**0.834**	**0.732**	42	124
		EC_{crisp}	0.644	0.855	0.735	23	–
	Meeting	OLED	**0.678**	**0.953**	**0.792**	30	107
(c)	*Moving*	ILED	0.947	**0.981**	**0.963**	55	**34**
		OLED	**0.963**	0.934	0.948	**31**	35
	Meeting	ILED	0.930	**0.976**	0.952	65	**30**
		OLED	**0.975**	0.933	**0.953**	**53**	42

learning approaches. We compare OLED to the following: (i) EC_{crisp}, a hand-crafted set of clauses for the CAVIAR dataset, described in (Artikis *et al.* 2010); (ii) EC_{MM} (Skarlatidis *et al.* 2015), a probabilistic version of EC_{crisp} with weights learnt by the Max-Margin weight learning method for Markov Logic Networks (MLNs) of (Huynh and Mooney 2009); (iii) XHAIL (Ray 2009), a hybrid abductive-inductive learner capable of learning programs in the EC. EC_{MM} was selected because it was shown to achieve good results on CAVIAR (Skarlatidis *et al.* 2015). XHAIL was selected as one of the few ILP systems that is able to learn theories in the EC. OLED and XHAIL were implemented using the Clingo[3] answer set solver as the core reasoning component, while the EC_{MM} approach used in this experiment was implemented in the LoMRF framework[4] for MLNs.

To evaluate EC_{MM}, (Skarlatidis *et al.* 2015) used a fragment of the CAVIAR dataset, which is also the one we use in this experiment. The target complex events in this dataset are related to two persons *meeting each other* or *moving together* and the training data consists of the parts of CAVIAR that involve these complex events. The fragment dataset contains a total of 25738 training interpretations. There are 6272 interpretations in which *moving* occurs and 3722 in which *meeting* occurs. OLED's results were achieved using significance $\delta = 10^{-5}$, a clause pruning threshold S_{min} of 0.7 for *meeting* and 0.5 for *moving* and specialization depth parameter $d = 2$ for *meeting* and $d = 1$ for *moving*. The results reported with this parameter configuration are the best among several other parameter settings that we tried for

[3] http://potassco.sourceforge.net/
[4] https://github.com/anskarl/LoMRF

S_{min} and d. The training time for each run of OLED was the maximum training time of the two parallel processes L_{init} and L_{term}.

Results were obtained using 10-fold cross validation and are presented in Table 4(a) in the form of *precision, recall* and f_1-*score*. These statistics were micro-averaged over the instances of recognized complex events from each fold of the 10-fold cross validation process. Table 4(a) also presents average training time per fold for all approaches except $\mathsf{EC}_{\mathsf{crisp}}$ (where no training is involved), average theory sizes (total number of literals) for OLED and XHAIL, as well as the fixed theory size of $\mathsf{EC}_{\mathsf{crisp}}$ and $\mathsf{EC}_{\mathsf{MM}}$.

$\mathsf{EC}_{\mathsf{MM}}$ achieves the best f_1-score for both complex events, followed closely by XHAIL. OLED achieves a comparable predictive accuracy (particularly for *meeting*), while it outscores the hand-crafted rules. Moreover, OLED achieves speed-ups of several orders of magnitude as compared to $\mathsf{EC}_{\mathsf{MM}}$ and XHAIL, due to its single-pass strategy. The superior accuracy of $\mathsf{EC}_{\mathsf{MM}}$ and XHAIL is due to them being batch learners, optimizing their respective outcomes over the entire training set. This also explains the increased training times for both. Regarding theory size, XHAIL learns significantly more compressed hypotheses than OLED. The reason is that XHAIL learns whole theories, while OLED learns each clause separately to gain in efficiency.

5.2 *Activity Recognition on the Entire CAVIAR Dataset*

We also present experimental results from running OLED on the entire CAVIAR dataset. The target complex events are *meeting* and *moving* as previously. The number of positive interpretations for both complex events is also the same as before, since the data fragment used in the previous experiment contains the parts of CAVIAR where these complex events occur. In contrast, the number of negative training instances is much larger in this experiment.

Due to the high training times of XHAIL and $\mathsf{EC}_{\mathsf{MM}}$, we do not present results with these approaches, and compare OLED only to the set of manually developed clauses $\mathsf{EC}_{\mathsf{crisp}}$. The experimental setting was as follows: We used 10-fold cross validation over the fragment used in the previous experiment, but in each fold, the training and test sets were augmented by a number of negative training sequences. In particular, in each fold, 90% of the negative training sequences from the remaining part of CAVIAR (i.e. the part not contained in the data fragment of the previous experiment) was added to the training set of the fold, while the remaining 10% was added to the test set. The parameter configuration for OLED was the same as in the previous experiment, with the exception of the specialization depth for *meeting*, which was set to $d = 1$. The limited size of the training sets in the experiment of Section 5.1 prevented OLED from sufficiently expanding its clauses when $d = 1$, resulting in over-general theories. Setting $d = 2$, thus trying 2-step specializations as well, made it possible to obtain the results reported in Table 4(a). In contrast, this was not necessary in this experiment, where due to the significantly larger training set size, OLED was able to find good clauses by trying 1-step specializations only.

Table 4(b) shows the results. Both approaches' performance is decreased, as compared to the previous experiment, due to the increased number of false positives, caused by the large number of additional negative instances. OLED still outscores the hand-crafted knowledge base.

5.3 Comparison with an Incremental Learner

We compared OLED to ILED (Katzouris *et al.* 2015), an incremental learner that is able to learn theories in the EC. Recall that ILED cannot learn from noisy data (see also Section 2), therefore, it cannot be used in CAVIAR, which exhibits various types of noise – see (Artikis *et al.* 2010) for details. In order to compare the two systems, we thus generated a noise-free version of CAVIAR with artificial annotation for the *moving* and *meeting* complex events. To produce the annotation, we used the hand-crafted knowledge base $\mathsf{EC}_{\mathsf{crisp}}$ for inference over the CAVIAR narrative. We used 10-fold cross validation to assess the performance of the compared systems. For each fold, the training (resp. test) set consisted of the 90% (resp. 10%) of positive and negative interpretations for each complex event. OLED's parameter setting was as reported in Section 5.2.

The results are presented in Table 4(c). The predictive accuracy for both systems is comparable, with ILED's being slightly better. This was expected, since ILED re-scans the historical memory of past data to revise its theories. Training times are also comparable, with OLED's being slightly higher, as compared to ILED's. ILED is able to avoid certain computations by inferring that they are redundant, based on the assumption that the data is noise-free. Regarding theory size, OLED learns significantly shorter hypotheses that ILED. OLED prunes a number of its learnt clauses, in an effort to avoid fitting potential noise in the data and also follows a conservative clause expansion strategy. In contrast, ILED tries to account for every positive example (and exclude every negative one), since it is designed for learning sound hypotheses.

5.4 Scalability

In this experiment we assess OLED's scalability. When learning from the entire CAVIAR dataset (Section 5.2) the average processing time per training interpretation was 6.7 milliseconds (ms), while the frame rate in CAVIAR, i.e. the rate in which video frames containing new data arrive is 40 ms. As a "stress-test", we evaluated OLED's performance in more demanding learning tasks. We generated 4 different datasets, each of which consisted of a number of copies of CAVIAR. The new datasets differ from the original one in the constants referring to the tracked entities in simple and complex events. We generated datasets consisting of 2, 5, 8 and 10 copies, each of which contained 20, 50, 80 and 100 different entities respectively. Like in the previous experiments, each interpretation includes narrative and annotation atoms from two time points. In this experiment however, the number of atoms in each interpretation grows proportionally to the number of copies of the dataset.

We performed learning with OLED on the original and the enlarged datasets and measured the average processing time per training interpretation. Figure 1 presents the results. For instance, interpretations in the 10 copies of CAVIAR are handled in approximately 2.5 sec in a standard desktop computer. The growth in average

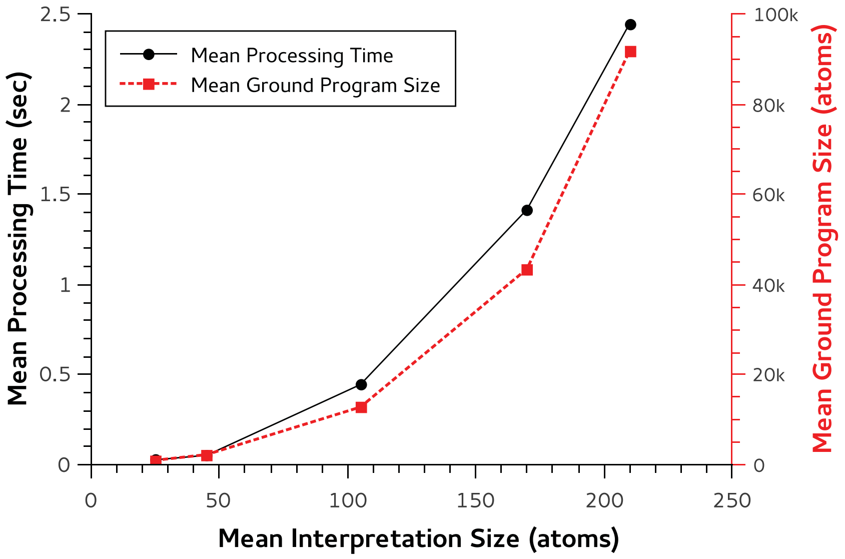

Fig. 1. OLED's mean processing time and mean ground program size per training interpretation, for varying interpretation sizes.

processing time is due to the increased number of annotation atoms in the datasets, as well as the additional domain constants, that result in an exponential increase in the size of the ground program produced during the clause evaluation process (see the dashed line in Figure 1). OLED's performance may be improved by some optimizations, such as taking advantage of domain knowledge about relational dependencies in the data. For instance, in CAVIAR complex events involve two different entities, therefore learning may be split across different processing cores that learn from independent parts of the data. Such optimizations are part of our current work.

6 Conclusions and Further Work

We presented OLED, an ILP system for online learning of complex event definitions in the Event Calculus. OLED is an any-time system that learns by a single-pass over a stream, using the Hoeffding bound to evaluate candidate clauses on a subset of the input. Results of the empirical evaluation indicate that OLED achieves speed-ups of several orders of magnitude, as compared to batch learners, with a comparable predictive accuracy. It also outscores hand-crafted rules and matches the performance of a sound incremental learner that can only operate on noise-free datasets. We intend to improve OLED in several aspects, including scalability and development of adaptive techniques for automated configuration of its parameters. We also plan to experiment with dialects of the EC that allow long-term temporal relations between entities and combine OLED with weight learning techniques towards online statistical relational learning.

Acknowledgements

This work was partly funded by the EU Project REVEAL (FP7 610928).

References

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.

Artikis, A., Skarlatidis, A. and Paliouras, G. 2010. Behaviour recognition from video content: a logic programming approach. *International Journal on Artificial Intelligence Tools 19,* 02, 193–209.

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.

Blockeel, H. and De Raedt, L. 1998. Top-down induction of first-order logical decision trees. *Artificial intelligence 101,* 1, 285–297.

Blockeel, H., De Raedt, L., Jacobs, N. and Demoen, B. 1999. Scaling up inductive logic programming by learning from interpretations. *Data Mining and Knowledge Discovery 3,* 1, 59–93.

De Raedt, L. 2008. *Logical and Relational Learning*. Springer Science & Business Media.

Denecker, M. and Kakas, A. 2002. Abduction in logic programming. In *Computational Logic: Logic Programming and Beyond*. Springer, 402–436.

Dhurandhar, A. and Dobra, A. 2012. Distribution-free bounds for relational classification. *Knowledge and information systems 31,* 1, 55–78.

Domingos, P. and Hulten, G. 2000. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 71–80.

Domingos, P. and Hulten, G. 2001. A general method for scaling up machine learning algorithms and its application to clustering. In *ICML*, Vol. 1. 106–113.

Esposito, F., Semeraro, G., Fanizzi, N. and Ferilli, S. 2000. Multistrategy theory revision: Induction and abduction in inthelex. *Machine Learning 38,* 1-2, 133–156.

Etzion, O. and Niblett, P. 2010. *Event Processing in Action*. Manning Publications Co.

Gama, J. 2010. *Knowledge Discovery from Data Streams*. CRC Press.

Gama, J. and Gaber, M. M. 2007. *Learning from Data Streams*. Springer.

Gama, J., Kosina, P., *et al.* 2011. Learning decision rules from data streams. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, Vol. 22. Citeseer, 1255.

Gebser, M., Kaminski, R., Kaufmann, B. and Schaub, T. 2012. Answer set solving in practice. *Synthesis Lectures on Artificial Intelligence and Machine Learning 6,* 3, 1–238.

Hoeffding, W. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association 58,* 301, 13–30.

Hulten, G., Domingos, P. and Abe, Y. 2003. Mining massive relational databases. In *Proceedings of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*. 53–60.

Huynh, T. N. and Mooney, R. J. 2009. Max-margin weight learning for markov logic networks. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 564–579.

Jensen, D. 1999. Statistical challenges to inductive inference in linked data. In *AISTATS*.

Jensen, D. and Neville, J. 2002. Autocorrelation and linkage cause bias in evaluation of relational learners. In *Inductive Logic Programming*, Springer, 101–116.

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

Kowalski, R. and Sergot, M. 1986. A logic-based calculus of events. *New Generation Computing 4,* 1, 67–95.

Lopes, C. and Zaverucha, G. 2009. Htilde: scaling up relational decision trees for very large databases. In *Proceedings of the 2009 ACM symposium on Applied Computing*. ACM, 1475–1479.

Muggleton, S. 1995. Inverse entailment and Progol. *New generation computing 13,* 3-4, 245–286.

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

Richards, B. L. and Mooney, R. J. 1995. Automated refinement of first-order horn-clause domain theories. *Machine Learning 19,* 2, 95–131.

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.

Yang, H. and Fong, S. 2011. Moderated vfdt in stream mining using adaptive tie threshold and incremental pruning. In *Data Warehousing and Knowledge Discovery*, Springer, 471–483.