Scalable Data Analytics Scalable Algorithms, Software Frameworks and Visualisation ICT-2013.4.2a

Project **FP7-619435 / SPEEDD**

Deliverable **D3.1**

Distribution **Public**

http://speedd-project.eu/

Event Recognition and Forecasting Technology (Part 2)

Anastasios Skarlatidis, Evangelos Michelioudakis, Nikos Katzouris, Alexandros Artikis, Georgios Paliouras, Elias Alevizos, Ioannis Vetsikas, Christos Vlassopoulos

Status: Final (Version 1.0)

December 2014

Project

Project ref.no.	FP7-619435
Project acronym	SPEEDD
Project full title	Scalable ProactivE Event-Driven Decision making
Porject site	http://speedd-project.eu/
Project start	February 2014
Project duration	3 years
EC Project Officer	Aleksandra Wesolowska

Deliverable

Deliverabe type	Report
Distribution level	Public
Deliverable Number	D3.1
Deliverable title	Event Recognition and Forecasting Technology (Part 2)
Contractual date of delivery	M1 (February 2014)
Actual date of delivery	December 2014
Relevant Task(s)	WP3/Task 3.1
Partner Responsible	NCSR “D”
Other contributors	
Number of pages	36
Author(s)	Anastasios Skarlatidis, Evangelos Michelioudakis, Nikos Katzouris, Alexandros Artikis, Georgios Paliouras, Elias Alevizos, Ioannis Vetsikas, Christos Vlassopoulos
Internal Reviewers	
Status & version	Final
Keywords	Event Recognition, Uncertainty, Markov Logic Networks, Event Calculus, Parameter Estimation, Answer Set Programming, Inductive Logic Programming, Abductive Logic Programming

Contents

1 Introduction **3**
1.1 History of the Document 3
1.2 Purpose and Scope of the Document 3
1.3 Relationship with Other Documents 4
1.4 Terminology Alignment 4

2 Machine Learning **5**
2.1 Introduction 5
2.2 Background 6
2.2.1 Event Calculus 7
2.2.2 Statistical Relational Learning 8
2.2.3 Markov Logic Networks 9
2.3 Probabilistic Inference 11
2.3.1 Marginal Inference 11
2.3.2 MAP Inference 12
2.4 Parameter Estimation 12
2.5 Structure Learning 13
2.5.1 Top-Down 14
2.5.2 Bottom-Up 14
2.5.3 Hypergraph Lifting 15
2.5.4 Structural Motifs 15
2.5.5 Online Structure Learning 16
2.6 Incremental Learning of Event Definitions 16
2.6.1 Running example: Activity recognition 18
2.6.2 The XHAIL System 19
2.6.3 ILED: Incremental Learning of Event Definitions 20
2.7 Experimental evaluation of ILED 22
2.7.1 Activity Recognition 22
2.7.2 City Transport Management 25

3 Conclusions and Future Work **29**

List of Figures

2.1 The research domain of Statistical Relational Learning combines logicbased representation with probabilistic modelling and machine learning. 9
2.2 Revision of a hypothesis H_n in response to a new example window w_n. $\mathcal{E}$ represents the historical memory of examples. 21
2.3 Average times needed for ILED to revise an initial hypothesis in the face of new evidence presented in windows of size 10, 50 and 100 examples. The initial hypothesis was obtained from a training set of varying size (1K, 10K, 50K and 100K examples) which subsequently served as the historical memory. 25

List of Tables

1.1 Terminology alignment between the two parts of the Deliverable 3.1. 4

2.1 The Event Calculus predicates. 7
2.2 An annotated stream of SDEs 18
2.3 Comparison of ILED and XHAIL. G is the window granularity. 24
2.4 Comparative performance of ILED and XHAIL on selected subsets of the CTM dataset each containing 20 examples. G is the granularity of the windows. 26
2.5 ILED with hierarchical bias. 28

Executive Summary

This document is the second part of the Deliverable 3.1 and presents the advancements made in event recognition and forecasting technology of the SPEEDD project, in order to reason about events and learn event definitions over large amounts of data, as well as under situations of uncertainty.

SPEEDD develops a system for proactive event-driven decision-making. Decisions are triggered by forecasting events — whether they correspond to problems or opportunities — instead of reacting to them once they happen. The decisions are made in real-time and require on-the-fly processing of Big Data, i.e., extremely large amounts of noisy data streaming from different geographical locations, as well as historical data. The effectiveness of the SPEEDD project will be evaluated in two use cases. First, proactive traffic management, aiming to forecast traffic congestions and thus to attenuate them. Second, proactive credit card fraud management, aiming to significantly improve fraud detection accuracy, without compromising efficiency, and forecast various types of fraudulent activity.

SPEEDD implements event recognition methods (also known as event pattern matching or event pattern detection systems), in order to extract useful information, in the form of events, by processing time-evolving data that comes from various sources (e.g., various types of sensor, network activity logs, ATMs, transactions, etc.). The extracted information — recognised and/or forecasted events — can be exploited by other systems or human experts, in order to monitor an environment and respond to the occurrence of significant events. Event recognition methods employ rich representation that can naturally and compactly represent events with complex relational structure, e.g., events that are related to other events with temporal constraints. Unfortunately, uncertainty is an unavoidable aspect of real-world event recognition applications. Consider for example, noisy or incomplete observations from road sensors, as well as imperfect definitions of fraudulent activity. Under situations of uncertainty, the performance of an event recognition system may be seriously compromised.

To address the above requirements, we combine probabilistic and logic-based modelling for representing and reasoning about events and their effects under uncertainty. Specifically, we take advantage of the succinct, structured and declarative representation of the Event Calculus formalism, in order to formally express events and their effects. To handle the uncertainty, we employ the state-of-the-art probabilistic and relational framework of Markov Logic Networks. The combination of probabilistic and logical modelling has also the advantage of expressing event definitions with well defined probabilistic and logical schematics and thus, we can employ state-of-the-art probabilistic reasoning and machine learning techniques.

Another important characteristic of the SPEEDD project, is that machine learning algorithms must deal with large amounts of data that continuously evolves. As a result, the current knowledge base of

event definitions may need to be refined or enhanced with new definitions. Therefore, the traditional approach of non-incremental batch machine learning algorithms cannot be applied in SPEEDD. To address this issue, we present our current advancements in incremental learning of event definitions from large amounts of noise-free data. We are planning to extend the probabilistic modelling methods with incremental event definition learning techniques.

1 Introduction

1.1 History of the Document

Version	Date	Author	Change Description
0.1	3/11/2014	Evangelos Michelioudakis (NCSR)	Set up of the document
0.2	4/11/2014	All Deliverable Authors (NCSR)	Structure of the document
0.3	10/11/2014	Anastasios Skarlatidis (NCSR)	Content adjusted: Event Calculus and MLN
0.4	17/11/2014	Evangelos Michelioudakis (NCSR)	Content adjusted: Learning for MLN
0.5	24/11/2014	Nikos Katzouris (NCSR)	Content adjusted: ILED
0.6	1/12/2014	Anastasios Skarlatidis (NCSR)	Content adjusted
1.0	16/12/2014	Anastasios Skarlatidis (NCSR)	Content adjusted in response to internal review

1.2 Purpose and Scope of the Document

This document presents the progress of the SPEEDD project with respect to event recognition and forecasting under uncertainty, as well as the current advancements to probabilistic inference and machine learning for event definitions. Furthermore, the presented work identifies the research directions that will be pursued in the second year of the project.

The reader is expected to be familiar with Complex Event Processing, Artificial Intelligence and Machine Learning techniques, as well as the general intent and concept of the SPEEDD project. The target relationship is:

- SPEEDD researchers
- SPEEDD audit

SPEEDD emphasises to scalable event recognition, forecasting and machine learning of event definitions for Big Data, under situations where uncertainty holds. This document presents the current advancements and discusses the scientific and technological issues that are being investigated in Work-Package 3.

1.3 Relationship with Other Documents

This document is related to project deliverable D6.1 “The Architecture Design of the SPEEDD prototype” that presents the on-line (including event recognition and forecasting) and off-line (machine learning) architecture of the SPEEDD prototype. Furthermore, deliverables D7.1 and D8.1 outline the requirements and the characteristics of the “Proactive Credit Card Fraud Management” and “Proactive Traffic Management” project use cases, respectively.

1.4 Terminology Alignment

Table 1.1 shows the terminology alignment between the two parts of the Deliverable 3.1.

D3.1, part 1 (IBM)	**D3.1, part 2 (NCSR)**
Raw Event	Simple, Derived Event (SDE)
Derived Event	Composite Event (CE)
Situation	

Table 1.1: Terminology alignment between the two parts of the Deliverable 3.1.

2

Machine Learning

2.1 Introduction

Fraud is an increasing business with increasing profits. Despite the advancing in terms of anti-fraud technologies, the larger number of transactions and the improvement of fraudsters skills make the fraud detection more challenging. Models for fraud detection have accompanied the evolution of both machine learning theory and the fraud itself. The general tendency is for the use of supervised classification (where the label differentiating genuine from fraudulent transactions is required) instead of unsupervised approaches (where fraud detection follows the detection of outliers in the data).

Moreover, fraud is constantly changing and therefore, it is fundamental to use adaptive approaches to follow this evolution. Advanced fraud detection machine learning techniques must also scale to handle billions of payments in history, hundreds of payments per second, and millisecond-level latencies. The number of transactions (and consequently, fraud) does not stop increasing and it is crucial to efficiently process large amounts of information.

Transportation and traffic congestion are crucial aspects due to the rapid increase in the number of vehicles. Traffic congestion results in excess delays, reduced safety, and increased environmental pollution. Traffic analysis and forecasting, necessary for a good management of transportation systems, require the analysis of massive data streams storming from various sensors, and this brings further difficult tasks (mainly about real-time processing of big quantities, geographically distributed and noisy data).

Both fraud detection and traffic management depend upon multiple layered and distributed information systems. As time evolves these systems share, collect and process data in various structured and unstructured digital formats. When such information is aggregated and correlated, it might become a source of significant knowledge and represent activities of importance for an organisation. These pieces of information, together with their temporal occurrence, can be represented by *events*.

An event is simply something that happens, or contemplated as happening (Luckham and Schulte, 2011). It provides the fundamental abstraction for representing time-evolving pieces of information. It can be anything, such as a sensor signal, a financial transaction, as well as the result of some intelligent processing, e.g., traffic flow measurements, etc. Regardless of the type of information that it carries, the important property of an event is that it occurs for some period of time. Temporally, the occurrence of an event may come in all sizes. It may happen instantaneously at some point in time or during some interval of time. Although an event as an entity represents a single piece of information, it might be related to other events in various ways, e.g., temporally, spatially, causally, etc. Furthermore, related events tend to occur in patterns, possibly mixed with other unrelated events. For example, in traffic management and monitoring the events representing that some vehicles at the same time (temporal relation) are moving slowly in close distance (spatial relations), may indicate the situation of an ongoing traffic congestion (event pattern/definition).

Automatic event recognition and forecasting of significant events can be performed by systems that employ Complex Event Processing (CEP) techniques. The aim of a CEP system is to recognise *composite events* (CEs) of interest, based on input streams of time-stamped symbols, that is *simple, derived events* (SDEs). A SDE is the result of applying a computational derivation process to some other event, such as an event coming from a sensor (Luckham and Schulte, 2011). CEs are defined as relational structures over other sub-events, either CEs or SDEs. Such CE definitions have the form of rules, usually expressed in a formal language, that capture the knowledge of domain experts. Due to the dynamic nature of the aforementioned use cases, the CE definitions may require to be refined or it may need to enhance the current knowledge base with new definitions. Manual creation of event definitions is a tedious and cumbersome process and thus machine learning techniques are required.

Unfortunately, uncertainty is an unavoidable aspect of real-world event recognition applications and it appears to be a consequence of several factors (Shet et al. 2007; Artikis et al. 2010a; Etzion and Niblett 2010, Section 11.2; Gal et al. 2011; Skarlatidis 2014). Under situations of uncertainty, the performance of an event recognition system may be seriously compromised. Below we outline the types of uncertainty that might appear in an event recognition application:

Erroneous input SDEs. Noisy observations result to streams containing erroneous input events. For example, noise in the signal transmission may distort the observed values.

Incomplete SDE streams. Partial observations result in incomplete input streams. For example, a sensor may fail for some period of time and stop sending information, interrupting the detection of a SDE.

Imperfect event definitions. Low-level detection systems often cannot detect all SDEs required for CE recognition, e.g. due to a limited number of sensing sources. Logical definitions of CEs, therefore, have to be constructed upon a limited and often insufficient dictionary of SDEs. Furthermore, when Machine Learning algorithms are used, similar patterns of SDEs may be inconsistently annotated. As a result, CE definitions and background knowledge, either learnt from data or derived by domain experts cannot strictly follow the annotation.

To overcome the issues of uncertainty and meet the aforementioned requirements, we are developing probabilistic inference and structure learning algorithms that operate under noisy environments and Big Data. Specifically, we combine a well-defined temporal logic formalism with statistical relational modelling and learning methods. Furthermore, we present the state-of-the-art approaches of supervised structure learning methods that can learn definitions from data, as well as we present our current work for scalable incremental learning of event definitions over large amounts of data.

2.2 Background

In order to develop such a mechanism for symbolic event recognition and learning under situations where various forms of uncertainty hold, a framework is required that combines a logic-based representation formalism (in our case the Event Calculus) with the field of Statistical Relational Learning.

2.2.1 Event Calculus

The Event Calculus, originally introduced by Kowalski and Sergot (1986), is a many-sorted first-order predicate calculus for reasoning about events and their effects. A number of different dialects have been proposed using either logic programming or classical logic — see Shanahan (1999), Miller and Shanahan (2002) and Mueller (2008) for surveys. Most Event Calculus dialects share the same ontology and core domain-independent axioms. The ontology consists of *time-points*, *events* and *fluents*. The underlying time model is often linear and may represent time-points as real or integer numbers. A *fluent* is a property whose value may change over time. When an *event* occurs it may change the value of a fluent. The core domain-independent axioms define whether a fluent holds or not at a specific time-point. Moreover, the axioms incorporate the common sense *law of inertia*, according to which fluents persist over time, unless they are affected by the occurrence of some event.

We base our model on an axiomatisation of a discrete version of the Event Calculus in first-order logic (Skarlatidis, 2014). The discrete dialect of the Event Calculus has been proven to be logically equivalent to the Event Calculus when the domain of time-points is limited to integers (Mueller, 2008). For the task of event recognition, we focus only on the domain-independent axioms that determine the influence of events to fluents and the inertia of fluents. Furthermore, similar to Artikis et al. (2010a), predicates stating the initiation and termination of fluents are only defined in terms of fluents and time-points. Table 2.1 summarises the main elements of the Event Calculus dialect. Variables (starting with an upper-case letter) are assumed to be universally quantified unless otherwise indicated. Predicates, functions and constants start with a lower-case letter.

Predicate	**Meaning**
$\texttt{happensAt}(E, T)$	Event E occurs at time-point T
$\texttt{holdsAt}(F, T)$	Fluent F holds at time-point T
$\texttt{initiatedAt}(F, T)$	Fluent F is initiated at time-point T
$\texttt{terminatedAt}(F, T)$	Fluent F is terminated at time-point T

Table 2.1: The Event Calculus predicates.

The Event Calculus axioms that determine when a fluent holds are defined as follows:

$$\begin{aligned}&\texttt{holdsAt}(F, T{+}1) \Leftarrow \\ &\qquad \texttt{initiatedAt}(F, T)\end{aligned} \tag{2.1}$$

$$\begin{aligned}&\texttt{holdsAt}(F, T{+}1) \Leftarrow \\ &\qquad \texttt{holdsAt}(F, T) \wedge \\ &\qquad \neg\texttt{terminatedAt}(F, T)\end{aligned} \tag{2.2}$$

Axiom (2.1) defines that if a fluent F is initiated at time T, then it holds at the next time-point. Axiom (2.2) specifies that a fluent continues to hold unless it is terminated.

The axioms that determine when a fluent does not hold are defined similarly:

$$\begin{aligned}&\neg\texttt{holdsAt}(F, T{+}1) \Leftarrow \\ &\qquad \texttt{terminatedAt}(F, T)\end{aligned} \tag{2.3}$$

$$\begin{aligned}&\neg\texttt{holdsAt}(F, T{+}1) \Leftarrow \\ &\qquad \neg\texttt{holdsAt}(F, T) \wedge \\ &\qquad \neg\texttt{initiatedAt}(F, T)\end{aligned} \tag{2.4}$$

According to Axiom (2.3), if a fluent F is terminated at time T then it does not hold at the next time-point. Axiom (2.4) states that a fluent continues not to hold unless it is initiated.

The predicates `happensAt`, `initiatedAt` and `terminatedAt` are defined only in a domain-dependent manner. `happensAt` expresses the input evidence, determining the occurrence of a simple, derived events (SDE) at a specific time-point. The input stream of observed SDEs, therefore, is represented in the Event Calculus as a *narrative* of ground `happensAt` predicates. `initiatedAt` and `terminatedAt` specify under which circumstances a fluent — representing a composite event (CE) — is to be initiated or terminated at a specific time-point. The domain-dependent rules of the Event Calculus, i.e., the initiation and/or termination of some $\texttt{fluent}_1$ over some domain-specific entities X and Y take the following general form:

$$\begin{aligned}
&\texttt{initiatedAt}(\texttt{fluent}_1(X,\,Y),\,T) \Leftarrow\\
&\qquad \texttt{happensAt}(\texttt{event}_\texttt{i}(X),\,T)\ \wedge\ \ldots\ \wedge\\
&\qquad \texttt{holdsAt}(\texttt{fluent}_\texttt{j}(X),\,T)\ \wedge\ \ldots\ \wedge\\
&\qquad \texttt{Conditions}[X,Y,T]\\
&\texttt{terminatedAt}(\texttt{fluent}_1(X,\,Y),\,T) \Leftarrow\\
&\qquad \texttt{happensAt}(\texttt{event}_\texttt{k}(X),\,T)\ \wedge\ \ldots\ \wedge\\
&\qquad \texttt{holdsAt}(\texttt{fluent}_\texttt{l}(X),\,T)\ \wedge\ \ldots\ \wedge\\
&\qquad \texttt{Conditions}[X,Y,T]
\end{aligned} \tag{2.5}$$

The domains of time-points, events and fluents, are represented by the finite sets $\mathcal{T}$, $\mathcal{E}$ and $\mathcal{F}$, respectively. All individual entities that appear in a particular event recognition task, e.g., persons, objects, etc., are represented by the constants of the finite set $\mathcal{O}$. $\texttt{Conditions}[X,Y,T]$ in (2.5) is a set of predicates, joined by conjunctions, that introduce further constraints in the definition, referring to time $T \in \mathcal{T}$ and entities $X,\,Y \in \mathcal{O}$. The predicates `happensAt` and `holdsAt`, as well as those appearing in $\texttt{Conditions}[X,Y,T]$, may also be negated. The initiation and termination of a fluent can be defined by more than one rule, each capturing a different initiation and termination case. With the use of `happensAt` predicates, we can define a CE over SDE observations. Similarly, with the `holdsAt` predicate we can define a CE over other CE, in order to create hierarchies of CE definitions. In both `initiatedAt` and `terminatedAt` rules, the use of `happensAt`, `holdsAt` and $\texttt{Conditions}[X,Y,T]$ is optional and varies according to the requirements of the target event recognition application.

2.2.2 Statistical Relational Learning

Similar to any pure logic-based formalism, the Event Calculus can compactly represent complex event relations. A knowledge base of Event Calculus axioms and CE definitions is defined by a set of first-order logic formulas. Each formula is composed of predicates that associate variables or constants, representing SDEs, CEs, time-points, etc. One of the strong motivations for using such a relational representation is its ability to directly express dependencies between related instances — e.g., events. While this type of representation is highly expressive, it cannot handle uncertainty. Each formula imposes a (hard) constraint over the set of possible worlds, that is, Herbrand interpretations. A missed or an erroneous SDE detection can have a significant effect on the event recognition results. For example, an initiation may be based on an erroneously detected SDE, causing the recognition of a CE with absolute certainty.

Statistical machine learning systems, e.g., methods that are based on probabilistic graphical models (Rabiner and Juang, 1986; Murphy, 2002; Lafferty et al., 2001), adopt a probabilistic approach to handle uncertainty. Such probabilistic models have been successfully used in real-word applications that involve various forms of uncertainty, such as speech recognition, natural language processing, activity recognition, etc. By employing statistical learning techniques, the parameters of such models are estimated automatically from training example sets. Compared to logic-based methods, probabilistic methods are less flexible for applications containing several entities and relations among them. By relying on propositional representations, it is difficult to represent complex relational data or assimilate prior domain knowledge (e.g., knowledge from experts or common sense knowledge). When the target application requires more expressive modelling capabilities, these methods typically are extended specifically for the application in an ad-hoc fashion — see for example the methods of Brand et al. (1997); Gong and Xiang (2003); Wu et al. (2007); Vail et al. (2007) and Liao et al. (2005).

Statistical Relational Learning (SRL) aims to develop methods that can effectively represent, reason and learn in domains with uncertainty and complex relational structure (e.g., relations among instances of SDEs and CEs). As shown in Figure 2.1, SRL combines a logic-based representation with probabilistic modelling and machine learning. In the domain of event recognition, the logic-based representation allows one to naturally define the relations between events and incorporate existing domain knowledge. This powerful representation is combined with probabilistic modelling, in order to naturally handle uncertainty. Using machine learning techniques, the model can automatically be estimated or refined according to the given set of example data.

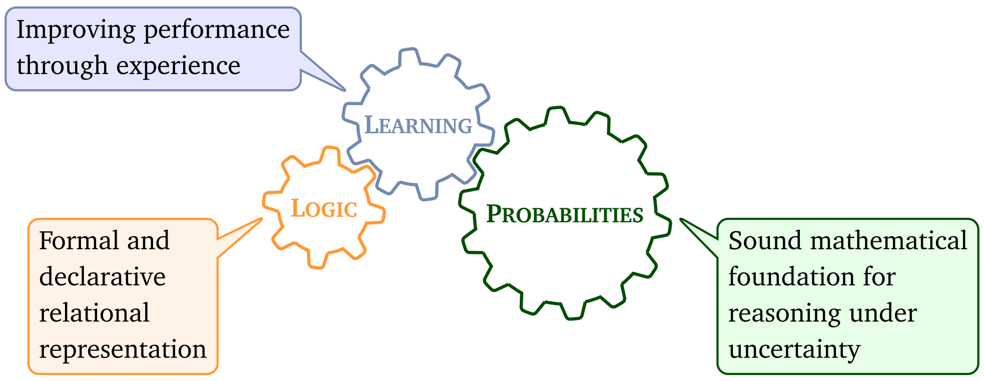
Improving performance through experience
LEARNING
LOGIC
PROBABILITIES
Formal and declarative relational representation
Sound mathematical foundation for reasoning under uncertainty

Figure 2.1: The research domain of Statistical Relational Learning combines logicbased representation with probabilistic modelling and machine learning.

Another advantage of SRL is that its probabilistic and logic-based representation naturally enables parameter sharing (also known as parameter tying). Specifically, the logic-based representation defines declaratively templates in the form of rules. Given some input evidence (e.g., observed SDEs), all instantiations of a particular rule share an identical structure and the same parameter (e.g., a weight value or some probability). Therefore, the number of parameters is reduced and the probability distribution is simplified, resulting to more efficient inference and learning. This parameter sharing is essentially similar to the plates notation (Buntine, 1994) or the transition probability distribution sharing between all states over time in Hidden Markov Models (Rabiner and Juang, 1986), but more generic as it is based on a logical representation.

2.2.3 Markov Logic Networks

Markov Logic Networks (MLN) (Domingos and Lowd, 2009) is a state-of-the-art SRL method that provides a framework which combines first-order logic representation with Markov Network modelling. Specifically, MLNs soften the constraints that are imposed by the formulas of a knowledge base and perform probabilistic inference. In MLNs, each formula F_i is represented in first-order logic and is associated with a weight value $w_i \in \mathbb{R}$. The higher the value of weight w_i, the stronger the constraint represented by formula F_i. In contrast to classical logic, all worlds in MLNs are possible with a certain probability. The main idea behind this is that the probability of a world increases as the number of formulas it violates decreases.

A knowledge base in MLNs may contain both hard and soft-constrained formulas. Hard-constrained formulas are associated with an infinite weight value and capture the knowledge which is assumed to be certain. Therefore, an acceptable world must at least satisfy the hard constraints. Soft constraints capture imperfect knowledge in the domain, allowing for the existence of worlds in which this knowledge is violated.

Formally, a knowledge base L of weighted formulas, together with a finite domain of constants $\mathcal{C}$, is transformed into a ground Markov network $M_{L,\mathcal{C}}$. All formulas are converted into *clausal form* and each clause is ground according to the domain of its distinct variables. The nodes in $M_{L,\mathcal{C}}$ are Boolean random variables, each one corresponding to a possible grounding of a predicate that appears in L. The predicates of a ground clause form a clique in $M_{L,\mathcal{C}}$. Each clique is associated with a corresponding weight w_i and a Boolean *feature*, taking the value 1 when the ground clause is true and 0 otherwise. The ground $M_{L,\mathcal{C}}$ defines a probability distribution over possible worlds and is represented as a log-linear model.

In event recognition we aim to recognise CEs of interest given the observed streams of SDEs. For this reason we focus on discriminative MLNs (Singla and Domingos, 2005), that are akin to Conditional Random Fields (Lafferty et al., 2001; Sutton and McCallum, 2007). Specifically, the set of random variables in $M_{L,\mathcal{C}}$ can be partitioned into two subsets. The former is the set of evidence random variables X, e.g., formed by a narrative of input ground `happens` predicates, representing the occurred SDEs. The latter is the set of random variables Y that correspond to groundings of query `holdsAt` predicates, as well as groundings of any other hidden/unobserved predicates, i.e., `initiates` and `terminates`. The joint probability distribution of a possible assignment of $Y{=}\mathbf{y}$, conditioned over a given assignment of $X{=}\mathbf{x}$, is defined as follows:

$$P(Y{=}\mathbf{y} \mid X{=}\mathbf{x}) = \frac{1}{Z(\mathbf{x})} exp\left(\sum_{i=1}^{|F_c|} w_i n_i(\mathbf{x}, \mathbf{y})\right) \tag{2.6}$$

The vectors $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$ represent a possible assignment of evidence X and query/hidden variables Y, respectively. $\mathcal{X}$ and $\mathcal{Y}$ are the sets of possible assignments that the evidence X and query/hidden variables Y can take. F_c is the set of clauses produced from the knowledge base L and the domain of constants $\mathcal{C}$. The scalar value w_i is the weight of the *i*-th clause and $n_i(\mathbf{x}, \mathbf{y})$ is the number of satisfied groundings of the *i*-th clause in $\mathbf{x}$ and $\mathbf{y}$. $Z(\mathbf{x})$ is the partition function, that normalises over all possible assignments $\mathbf{y}' \in \mathcal{Y}$ of query/hidden variables given the assignment $\mathbf{x}$, that is, $Z(\mathbf{x}) = \sum_{\mathbf{y}' \in \mathcal{Y}} exp(\sum_{i}^{|F_c|} w_i n_i(\mathbf{x}, \mathbf{y}'))$.

Equation (2.6) represents a single exponential model for the joint probability of the entire set of query variables that is globally conditioned on a set of observables. Such a conditional model can have a much simpler structure than a full joint model, e.g., a Bayesian Network. By modelling the conditional distribution directly, the model is not affected by potential dependencies between the variables in X and can ignore them. The model also makes independence assumptions among the random variables Y, and defines by its structure the dependencies of Y on X. Therefore, conditioning on a specific assignment $\mathbf{x}$, given by the observed SDEs in event recognition, reduces significantly the number of possible worlds and inference becomes much more efficient (Singla and Domingos, 2005; Minka, 2005; Sutton and McCallum, 2007).

2.3 Probabilistic Inference

When performing event recognition in noisy enviroments using the Markov Logic Networks framework, efficient methods for probabilistic inference are required. Directly computing the Equation (2.6) is intractable, because the value of $Z(\mathbf{x})$ depends on the relationship among all clauses in the knowledge base. For this reason, a variety of efficient inference algorithms have been proposed in the literature, based on local search and sampling (Poon and Domingos, 2006; Singla and Domingos, 2006; Biba et al., 2011), variants of Belief Propagation (Singla and Domingos, 2008; Kersting et al., 2009; Gonzalez et al., 2009; Kersting, 2012), Integer Linear Programming (Riedel, 2008; Huynh and Mooney, 2009; Noessner et al., 2013), lifted model counting (Gogate and Domingos, 2011; den Broeck et al., 2011; Apsel and Brafman, 2012), etc. Below we present the two types of inference that can be performed in MLNs — i.e., marginal inference and maximum a-posteriori inference (MAP).

2.3.1 Marginal Inference

In event recognition, marginal inference computes the conditional probability that CEs hold given an input of observed SDEs:

$$P(\texttt{holdsAt}(\texttt{CE},\texttt{T}) = \mathit{True} \mid \mathit{SDEs})$$

In other words, this probability value measures the confidence that the CE is recognised. Since it is #P-complete to compute this probability, we can employ Markov Chain Monte Carlo (MCMC) sampling algorithms to approximate it.

Due to the combination of logic with probabilistic modelling, inference in MLN must handle both deterministic and probabilistic dependencies. Deterministic or near-deterministic dependencies are formed from formulas with infinite and strong weights respectively. Being a purely statistical method, MCMC can only handle probabilistic dependencies. In the presence of deterministic dependencies, two important properties of Markov Chains, *ergodicity* and *detailed balance*, are violated and the sampling algorithms give poor results (Poon and Domingos, 2006). Ergodicity is satisfied if all states are aperiodically reachable from each other, while detailed balance is satisfied if the probability of moving from state y to state y' is the same as the probability of moving from y' to y. Ergodicity and detailed balance are violated in the presence of deterministic dependencies because these dependencies create isolated regions in the state space by introducing zero-probability (impossible) states. Even near-deterministic dependencies create regions that are difficult to cross — i.e., contain states with near zero-probability. As a result, typical MCMC methods, such as Gibbs sampling (Casella and George, 1992), get trapped in local regions. Thus, they are unsound for deterministic dependencies and they find it difficult to converge in the presence of near-deterministic ones.

To overcome these issues and deal with both deterministic and probabilistic dependencies, we employ the state-of-the-art MC-SAT algorithm (Poon and Domingos, 2006), which is a MCMC method that combines satisfiability testing with *slice-sampling* (Damlen et al., 1999). Initially, a satisfiability solver is used to find those assignments that satisfy all hard-constrained clauses (i.e., clauses with infinite weights). At each subsequent sampling step, MC-SAT chooses from the set of ground clauses satisfied by the current state the clauses that must be satisfied at the next step. Each clause is chosen with probability proportional to its weight value. Clauses with infinite or strong weights, that represent, deterministic and near-deterministic dependencies will always be chosen with absolute certainty and high probability, respectively. Then, instead of taking a sample from the space of all possible states, slice-sampling restricts sampling to the states that satisfy at least all chosen clauses. In this manner, MCMC cannot get trapped in local regions, as satisfiability testing helps to collect samples from all isolated and difficult-to-cross regions.

2.3.2 MAP Inference

MAP inference, on the other hand, identifies the most probable assignment among all `holdsAt` instantiations that are consistent with the given input of observed SDEs:

$$\underset{\texttt{holdsAt}}{\operatorname{argmax}}\, P(\texttt{holdsAt(CE, T)} \mid \mathit{SDEs})$$

This task reduces to finding the truth assignment of all `holdsAt` instantiations that maximises the sum of weights of satisfied ground clauses. This is equivalent to the weighted maximum satisfiability problem. The problem is NP-hard in general and there has been significant work on finding an approximate solution efficiently using local search algorithms (e.g., MaxWalkSAT (Kautz et al., 1997), see Hoos and Stützle (2004) for in depth analysis) or using linear programming methods (e.g., Riedel (2008); Huynh and Mooney (2009); Noessner et al. (2013)).

We employ both approaches for approximate MAP inference by using the classic local search MaxWalkSAT solver proposed by Kautz et al. (1997) and the LP-relaxed Integer Linear Programming method proposed by Huynh and Mooney (2009). In particular, for the latter algorithm, the ground Markov network is translated into a set of linear constraints and solved using standard linear optimisation algorithms. Due to the NP-hardness of the problem, the linear programming solver usually returns non-integral solutions — i.e., the assignment of some ground $\texttt{holdsAt}(\texttt{CE}, \texttt{T})$ is not Boolean, but within the open interval $(0, 1)$. For that reason, the method uses a rounding procedure called ROUNDUP (Boros and Hammer, 2002). Specifically, the procedure iteratively assigns the truth value of non-integral ground atoms, by satisfying the clauses that appear with respect to their cost (i.e., summation of their weights). Compared to the local search MaxWalkSAT algorithm, the linear programming approach typically achieves higher accuracy (Huynh and Mooney, 2009).

2.4 Parameter Estimation

The weights of the soft-constrained clauses in MLNs can be estimated from training data, using supervised learning techniques. When the goal is to learn a model that recognises CEs with some confidence (i.e., probability), then the most widely adopted learning approach is to minimise the negative conditional log-likelihood (CLL) function that is derived from Equation (2.6). Given training data that are composed of a set X of evidence predicates (e.g., ground `happens` predicates) and their corresponding query predicates Y (e.g., ground `holdsAt` predicates), the negative CLL has the following form:

$$-\log P_w(Y{=}\mathbf{y} \mid X{=}\mathbf{x}) = \log Z(\mathbf{x}) - \sum_{i=1}^{|F_c|} w_i n_i(\mathbf{x}, \mathbf{y})$$

The vector $\mathbf{x}$ is the assignment of truth values to the evidence random variables X, according to the training data. Similarly, $\mathbf{y}$ represents a possible assignment of truth values to the query random variables Y that are provided as annotation. CLL is used to evaluate how well the model fits the given training data.

The parameters of the model are the weight values w_i that are associated to the soft-constrained clauses in F_c and can be estimated by using either *first-order* or *second-order* optimisation methods (Singla and Domingos, 2005; Lowd and Domingos, 2007). First-order methods apply standard gradient descent optimisation techniques, e.g., the Voted Perceptron algorithm (Collins, 2002; Singla and Domingos, 2005), while second-order methods pick a search direction based on the quadratic approximation of the target function. As stated by Lowd and Domingos (2007), second-order methods are more appropriate for MLN training, as they do not suffer from the problem of *ill-conditioning*. In a training set some clauses may have a significantly greater number of satisfied groundings than others, causing the variance of their counts to be correspondingly larger. This situation makes the convergence of the standard gradient descent methods very slow, since there is no single appropriate learning rate for all soft-constrained clauses.

If the goal is to predict accurate target-predicate probabilities, these approaches are well motivated. However, in many applications, the actual goal is to maximise an alternative performance metric such as classification accuracy or F-measure. Max-margin methods are a competing approach to discriminative training and they also have the advantage that can be adapted to maximise a variety of performance metrics in addition to classification accuracy (Joachims, 2005).

Thus, an alternative approach to CLL optimisation is max-margin training, which is better suited to problems where the goal is to maximise the classification accuracy (Huynh and Mooney, 2009, 2011a). Instead of optimising the CLL, max-margin maximise the following ratio:

$$\frac{P(Y{=}\mathbf{y}|X{=}\mathbf{x},\mathbf{w})}{P(Y{=}\hat{\mathbf{y}}|X{=}\mathbf{x},\mathbf{w})}$$

The above equation measures the ratio between the probability of correct truth assignment $\mathbf{y}$ of CEs and the closest competing incorrect truth assignment $\hat{\mathbf{y}}{=}\text{argmax}_{\bar{\mathbf{y}}\in\mathbf{Y}\setminus\mathbf{y}}P(Y{=}\bar{\mathbf{y}}|X{=}\mathbf{x})$ which is also known as separation oracle. We employ the method of Huynh and Mooney (2009) which formulates the max-margin problem as 1-slack structural support vector machine (SVM) using a cutting-plane algorithm proposed by Joachims et al. (2009). Specifically, structural SVMs are predicting structured outputs instead of simple labels or real values. In particular, they describe the problem of learning a function $h : \mathcal{X} \mapsto \mathcal{Y}$, where $\mathcal{X}$ is the space of inputs examples, and $\mathcal{Y}$ is the space of multivariate and structured outputs from the set of training examples $S = ((\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)) \in (\mathcal{X} \times \mathcal{Y})^n$.

The goal is to find a function h that has low prediction error. This can be accomplished by learning a discriminant function $f : \mathcal{X} \times \mathcal{Y} \mapsto \mathcal{R}$ and maximise f over all $\mathbf{y} \in \mathcal{Y}$ for a given input $\mathbf{x}$ to get a classifier of the form:

$$h_w(\mathbf{x}) = \underset{\mathbf{y}\in\mathcal{Y}}{\text{argmax}} f_w(\mathbf{x}, \mathbf{y})$$

The discriminant function $f_w(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ is linear in the space of features, where $\mathbf{w} \in \mathcal{R}^N$ is a parameter vector and $\mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ is a feature vector relating an input $\mathbf{x}$ and output $\mathbf{y}$. The features need to be designed for a given problem so that they capture the dependency structure of $\mathbf{y}$ and $\mathbf{x}$ and the relations among the outputs $\mathbf{y}$. In our case $\mathbf{\Psi}(\mathbf{x}, \mathbf{y}) = \mathbf{n}(\mathbf{x}, \mathbf{y})$ which is the number of satisfied groundings in $\mathbf{x}$ and $\mathbf{y}$ (see Equation (2.6)). Therefore, the goal is to find a parameter vector $\mathbf{w}$ that maximises the margin by employing linear programming techniques such as the Integer Linear Programming inference method described in Section 2.3.2.

2.5 Structure Learning

In order to develop a method for learning event definitions from historical data, we are researching existing algorithms and study various approaches. Below we briefly present structure learning methods for Markov Logic Networks.

2.5.1 Top-Down

The top-down structure learning (TDSL) approach (Kok and Domingos, 2005), learn or revise an MLN one clause at a time, one literal at a time. The initial structure can either be an empty network or existing KB. Either way, it is useful to start by adding all unit clauses (single atoms) to the MLN. The weights of these capture the marginal distributions of the predicates, allowing the longer clauses to focus on modelling predicate dependencies. To extend this initial model, TDSL either repeatedly finds the best clause using beam search (Furcy and Koenig, 2005) and adds it to the MLN, or adds all "good" clauses of length l before trying clauses of length $l + 1$. Candidate clauses are formed by adding each predicate (negated or otherwise) to each current clause, with all possible combinations of variables, subject to the constraint that at least one variable in the new predicate must appear in the current clause. Weighted pseudo-log-likelihood (WPLL) is used as an evaluation measure:

$$\log P^{\bullet}(X = x) = \sum_{r \in R} c_r \sum_{k=1}^{g_r} \log P_w(X_{r,k} = x_{r,k} | MB_x(X_{r,k})) \tag{2.7}$$

where R is the set of first-order atoms, g_r is the number of groundings of first-order atom r, and $x_{r,k}$ is the truth value of the k-th grounding of r. By default, atom weights are simply set $c_r = 1/g_r$, which has the effect of weighting all first-order predicates equally. To combat overfitting, TDSL penalises the WPLL using a structure prior of $e^{-\alpha \sum_{i=1} F d_i}$, where d_i is the number of literals that differ between the current version of the clause and the original one.

TDSL follows a blind generate and test strategy in which many potential changes to an existing model are systematically generated independent of the training data, and then tested for empirical adequacy. For complex models such as MLNs, the space of potential revisions is combinatorially explosive and such a search can become difficult to control, resulting in convergence to suboptimal local maxima.

2.5.2 Bottom-Up

Bottom-up learning methods attempt to use the training data to directly construct promising structural changes or additions to the model, avoiding many of the local maxima and plateaus in a large search space. The Bottom-Up Structure Learning algorithm (BUSL), introduced by Mihalkova and Mooney (2007), applies this bottom-up approach to the task of learning MLN structure. Empirically, BUSL often yields more accurate models much faster than purely top-down structure learning.

BUSL uses a propositional Markov network structure learner to construct template networks that then guide the construction of candidate clauses. The template networks are composed of *template nodes*, conjunctions of one or more literals that serve as building blocks for creating clauses. Template nodes are constructed by looking for groups of constant-sharing ground literals that are true in the data and abstracting them by substituting variables for the constants. Thus, these template nodes could also be viewed as portions of clauses that have true groundings in the data. To understand why conjunctions of literals with true groundings are good candidates for clause components, consider the special case of a definite clause: $L_1 \wedge \cdots \wedge L_n \Rightarrow P$. If the conjoined literals in the body have no true groundings, then the clause is always trivially satisfied. Therefore, true conjunctions will be most useful for building effective clauses.To search for these, BUSL generates clause candidates by focusing on each maximal clique in turn and producing all possible clauses consistent with it. The candidates are then evaluated using the WPLL score Equation (2.7), as in top-down structure learning.

BUSL restricts its structure search for clauses only to those candidates whose literals correspond to template nodes that form a clique in the template. It also makes a number of additional restrictions on the search in order to decrease the number of free variables in a clause, thus decreasing the size of the ground MLN during inference, and further reducing the search space. Therefore BUSL typically restrict the search to very short paths, creating short clauses from them and greedily joining them into longer ones. Although is faster than the top-down approach, still has the problem of convergence to suboptimal local maxima.

2.5.3 Hypergraph Lifting

Learning via Hypergraph Lifting (LHL), introduced by Kok and Domingos (2009), is an approach which directly utilises the data in constructing candidates using relational pathfinding to a fuller extent than previous ones. It mitigates the exponential search problem by first inducing a more compact representation of data, in the form of a hypergraph over clusters of constants. Pathfinding on this lifted hypergraph is typically at least an order of magnitude faster than on the ground training data, and produces MLNs that are more accurate than previous approaches. A relational database can be viewed as a hypergraph with constants as nodes and relations as hyperedges. LHL finds paths of true ground atoms in the hypergraph that are connected via their arguments. To make this tractable (there are exponentially many paths in the hypergraph), the hypergraph is lifted by jointly clustering the constants to form higher-level concepts, and find paths in it. Then, the ground atoms are variabilised in each path, and they are used to form clauses, which are evaluated using a pseudo-likelihood measure. Finally, LHL iterates over the clauses from shortest to longest and for each clause, compares its score against those of its sub-clauses. If a clause scores higher than all the sub-clauses, it is retained, otherwise is discarded because is unlikely to be useful. Finally the retained clauses are added to an MLN, relearn the weights, and keep the clauses in the MLN which improve the overall WPLL Equation (2.7).

2.5.4 Structural Motifs

Learning using Structural Motifs (LSM), introduced by Kok and Domingos (2010), is an approach that can find long formulas. Its key insight is that relational data usually contains recurring patterns, which we term structural motifs. These motifs confer three benefits. First, by confining its search to occur within motifs, LSM need not waste time following spurious paths between motifs. Second, LSM only searches in each unique motif once, rather than in all its occurrences in the data. Third, by creating various motifs over a set of objects, LSM can capture different interactions among them. Structural motif is frequently characterised by objects that are densely connected via many paths, and can be identified by using the concept of truncated hitting time in random walks. LHL, described above, represents a database as a hypergraph containing fewer nodes and therefore fewer paths, ameliorating the cost of finding paths in the next component. In LHL, two nodes are clustered together if they are related to many common nodes. Thus, intuitively, LHL is making use of length-2 paths to determine the similarity of nodes. In contrast, LSM uses longer paths, and thus more information, to find clusterings of nodes (motifs). In addition, LSM finds various clusterings rather than just a single one.

A structural motif is a set of literals, which defines a set of clauses that can be created by forming disjunctions over the negations/non-negations of one or more of the literals. Thus, it defines a subspace within the space of all clauses. LSM discovers subspaces where literals are densely connected and groups them into a motif. Each hyperedge is labeled with a predicate symbol. LSM groups nodes that are densely connected by many paths and the hyperedges connecting the nodes into a motif. Then it compresses the motif by clustering the nodes into high-level concepts, reducing the search space of clauses in the motif. Finally, LSM runs relational pathfinding on each motif to find candidate rules, and retains the good ones in an MLN using the same algorithms as LHL.

Both LHL and LSM are data-driven structure learning algorithms and they cannot exploit background knowledge of axioms as well as mode declarations for learning rules which are key methods on relational structure learning approaches as Inductive Logic Programming. This inability makes them inappropriate for capturing complex relation and learning qualitative meaningful rules.

2.5.5 Online Structure Learning

All previous methods for learning the structure are batch algorithms that are effectively designed for training data with relatively few mega-examples. A mega-example is a large set of connected facts which are disconnected and independent from each other. Moreover, most existing weight learning methods for MLNs employ batch training where the learner must repeatedly run inference over all training examples in each iteration, which becomes computationally expensive for large datasets.

Online Structure Learning (OSL) (Huynh and Mooney, 2011b) updates both parameters and structure using incremental structure and parameter learning. In particular, OSL consider the predicted possible worlds, measure the difference from the ground-truth ones and searches for clauses that differentiate them. This is related to the idea of Inductive Logic Programming (ILP). In this case each ground-truth possible world plays the role of a positive example and any predicted possible world that differs from the ground-truth possible world is incorrect and can be considered as a negative example.

Therefore, at each step t, OSL receives an example, produces the predicted possible world and finds the atoms that differentiate it from the ground-truth one. Then searches the ground-truth possible world for clauses specific to this set of atoms. In order to find useful clauses specific to a set of atoms (true ground atoms), relational pathfinding is used combined with mode declarations to speed up the process. Subsequently, these paths must be generalised to form first-order clauses. A standard way to generalise paths is to replace each constant in a conjunction with a variable. However, for many tasks, it is critical to have clauses that are specific to a particular constant. Consequently, OSL introduce mode declarations for creating clauses and variabilises all constants in a conjunction according to these mode declarations. Finally, the resulting set of clauses is added to an empty MLN and learn their parameters using an online variation of the max-margin learner introduced above. Therefore, OSL shows much potential for structure learning in MLNs and particularly when incremental properties and speed are desired.

2.6 Incremental Learning of Event Definitions

In this section we present a machine learning technique for the automatic construction of event definitions, in the language of the Event Calculus. Our approach relies on Inductive Logic Programming (ILP) (Muggleton and Raedt, 1994; Lavrac and Dzeroski, 1993), a sub-field of machine learning that uses logic programming as a unifying representation language for the background knowledge, the training examples and the constructed hypothesis. Although ILP has been successfully applied in a variety of problems, learning Event Calculus theories from large amounts of data has several challenges that make most ILP systems inappropriate.

The first difficulty is related to Negation as Failure (NaF) (Clark, 1977), which the Event Calculus uses to model inertia. Most ILP learners cannot handle NaF at all, or lack a robust NaF semantics (Sakama, 2000; Ray, 2009). Another problem that often arises when dealing with events, is the need to infer implicit or missing knowledge, such as possible causes of observed events. As an example, consider the task of learning a set of domain-specific axioms, i.e., a set of initiatedAt and terminatedAt rules, from examples in the form of holdsAt literals. Since the target predicates are missing from the supervision, one must "discover" proper instances of them in order to construct a hypothesis. In ILP, such a problem is called *non-Observational Predicate Learning (non-OPL)* (Muggleton, 1995) and is a task that most ILP systems have difficulty to handle. One way to address this problem is through the combination of ILP with Abductive Logic Programming (ALP) (Denecker and Kakas, 2002; Kakas and Mancarella, 1990; Kakas et al., 1993). Abduction in logic programming is usually given a non-monotonic semantics (Eshghi and Kowalski, 1989) and in addition, it is by nature an appropriate framework for reasoning with incomplete knowledge. Although it has a long history in the literature (Ade and Denecker, 1995), only recently has this combination brought about systems such as XHAIL (Ray, 2009), TAL (Corapi et al., 2010) and ASPAL (Corapi et al., 2011; Athakravi et al., 2013) that may be used for the induction of event-based knowledge.

The majority of ILP systems are *batch learners*, in the sense that all training data must be in place prior to the initiation of the learning process. This is not always suitable for event-oriented learning tasks, where data is often collected at different times and under various circumstances, or arrives in streams. In order to account for new training examples, a batch learner has no alternative but to re-learn a hypothesis from scratch. The cost is poor scalability when "learning in the large" (Dietterich et al., 2008) from a growing set of data. This is particularly true in the case of temporal data, which usually come in large volumes. Consider for instance data which span a large period of time, or sensor data transmitted at a very high frequency. ILP algorithms do not scale well to large volumes of data, given that the expressive power of logic programming comes at the cost of increased complexity in ILP.

An alternative approach is learning incrementally, that is, processing training instances when they become available, and altering previously inferred knowledge to fit new observations, instead of discarding it and starting from scratch. This process, also known as *Theory Revision* (Wrobel, 1996), exploits previous computations to speed-up the learning, since revising a hypothesis is generally considered more efficient than learning it from scratch (Biba et al., 2008; Esposito et al., 2000; Cattafi et al., 2010). However, scaling to the large volumes of today's datasets or handling streaming data remains an open issue, and the development of scalable algorithms for theory revision has been identified as an important research direction (Muggleton et al., 2012). As historical data grow over time, it becomes progressively harder to revise knowledge, so that it accounts both for new evidence and past experience. One direction towards scaling theory revision systems is the development of techniques for reducing the need for reconsulting the whole history of accumulated experience, while updating existing knowledge.

This is the direction we take in this project. We build on the ideas of non-monotonic ILP and use XHAIL as the basis for a scalable, incremental learner for the induction of event definitions in the form of Event Calculus theories. We describe a compressive "memory" structure, incorporated in the learning process, which reduces the need for reconsulting past experience in response to a revision. Using this structure, we propose a method which, given a stream of examples, a theory which accounts for them and a new training instance, requires at most one pass over the examples in order to revise the initial theory, so that it accounts for both past and new evidence. We evaluate empirically our approach on real and synthetic data from an activity recognition application and a transport management application. Our results indicate that our approach is significantly more efficient than XHAIL, without compromising predictive accuracy, and scales adequately to large data volumes.

The rest of this section is structured as follows. Section 2.6.1 presents the domain of activity recognition, which we use as a running example. Section 2.6.2 presents the XHAIL system. Section 2.6.3 presents our incremental learning approach. Experimental evaluation in presented in Section 2.7.

Narrative	**Annotation**
...	...
happensAt(inactive(id_1), 999)	not holdsAt(moving(id_1, id_2), 999)
happensAt(active(id_2), 999)	
holdsAt(coords(id_1, 201, 432), 999)	
holdsAt(coords(id_2, 230, 460), 999)	
holdsAt(direction(id_1, 270), 999)	
holdsAt(direction(id_2, 270), 999)	
happensAt(walking(id_1), 1000)	not holdsAt(moving(id_1, id_2), 1000)
happensAt(walking(id_2), 1000)	
holdsAt(coords(id_1, 201, 454, 1000)	
holdsAt(coords(id_2, 230, 440, 1000)	
holdsAt(direction(id_1, 270, 1000)	
holdsAt(direction(id_2, 270, 1000)	
happensAt(walking(id_1), 1001)	holdsAt(moving(id_1, id_2), 1001)
happensAt(walking(id_2), 1001)	
holdsAt(coords(id_1, 201, 454), 1001)	
holdsAt(coords(id_2, 227, 440), 1001)	
holdsAt(direction(id_1, 275), 1001)	
holdsAt(direction(id_2, 278), 1001)	
...	...

Table 2.2: An annotated stream of SDEs

2.6.1 Running example: Activity recognition

We use the task of activity recognition, as defined in the CAVIAR[1] project, as a running example. The CAVIAR dataset consists of videos of a public space, where actors walk around, meet each other, browse information displays, fight and so on. These videos have been manually annotated by the CAVIAR team to provide the ground truth for two types of activity. The first type corresponds to SDEs, that is, knowledge about a person's activities at a certain time point (for instance *walking*, *running*, *standing still* and so on). The second type corresponds to CEs, activities that involve more than one person, for instance two people *moving together*, *fighting*, *meeting* and so on. The aim is to recognise CEs by means of combinations of SDEs and some additional domain knowledge, such as a person's position and direction at a certain time point.

Low-level events are represented in the Event Calculus by streams of ground happensAt/2 atoms (see Table 2.2), while CEs and other domain knowledge are represented by ground holdsAt/2 atoms. Streams of SDEs together with domain-specific knowledge will henceforth constitute the *narrative*, in ILP terminology, while knowledge about CEs is the *annotation*. Table 2.2 presents an annotated stream of SDEs. We can see for instance that the person id_1 is $inactive$ at time 999, her (x, y) coordinates are $(201, 432)$ and her direction is 270°. The annotation for the same time point informs us that id_1 and id_2

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

are not moving together. Fluents express both CEs and input information, such as the coordinates of a person. We discriminate between *inertial* and *statically defined* fluents. The former should be inferred by the Event Calculus axioms, while the latter are provided with the input.

Given such a domain description in the language of the Event Calculus, the aim of machine learning addressed in this work is to automatically derive the *Domain-Specific Axioms*, that is, the axioms that specify how the occurrence of SDEs affects the truth values of the fluents that represent CEs, by initiating or terminating them. Thus, we wish to learn initiatedAt and terminatedAt definitions from positive and negative examples from the narrative and the annotation.

Henceforth, we use the term "example" to encompass anything known true at a specific time point. We assume a closed world, thus anything that is not explicitly given is considered false (to avoid confusion, in the tables throughout the paper we state both negative and positive examples). An example's time point will also serve as reference. For instance, three different examples e_{999}, e_{1000} and e_{1001} are presented in Table 2.2. According to the annotation, an example is either positive or negative w.r.t. a particular CE. For instance, e_{1000} in Table 2.2 is a negative example for the *moving* CE, while e_{1001} is a positive example.

2.6.2 The XHAIL System

XHAIL constructs hypotheses in a three-phase process. Given an ILP task $ILP(B, E, M)$, the first two phases return a ground program K, called *Kernel Set of* E, such that $B \cup K \vDash E$. The first phase generates the heads of K's clauses by abductively deriving from B a set Δ of instances of head mode declaration atoms, such that $B \cup \Delta \vDash E$. The second phase generates K, by saturating each previously abduced atom with instances of body declaration atoms that deductively follow from $B \cup \Delta$.

The Kernel Set is a multi-clause version of the *Bottom Clause*, a concept widely used by inverse entailment systems like PROGOL and ALEPH. These systems construct hypotheses one clause at a time, using a positive example as a "seed", from which a most-specific Bottom Clause is generated by inverse entailment (Muggleton, 1995). A "good", in terms of some heuristic function, hypothesis clause is then constructed by a search in the space of clauses that subsume the Bottom Clause. In contrast, the Kernel Set is generated from all positive examples at once, and XHAIL performs a search in the space of theories that subsume it, in order to arrive at a "good" hypothesis. This is necessary due to difficulties related to the non-monotonicity of NaF, which are typical of systems that learn one clause at a time. Another important difference between the Kernel Set and the Bottom Clause is that the latter is constructed by a seed example that must be provided by the supervision, while the former can also utilise atoms that are derived abductively from the background knowledge, allowing to successfully address non-OPL problems.

In order to utilise the Kernel Set as a search space, it first needs to be *variabilised*. To do so, each term in a Kernel Set clause that corresponds to a variable, as indicated by the mode declarations, is replaced by an actual variable, while each term that corresponds to a ground term is retained intact.

The third phase of XHAIL functionality concerns the actual search for a hypothesis. Contrary to other inverse entailment systems like PROGOL and ALEPH, which rely on a heuristic search, XHAIL performs a complete search in the space of theories that subsume K_v in order to ensure soundness of the generated hypothesis. This search is biased by *minimality*, i.e., preference towards hypotheses with fewer literals. A hypothesis is thus constructed by dropping as many literals and clauses from K_v as possible, while correctly accounting for all the examples.

To sum up, XHAIL provides an appropriate framework for learning event definitions in the form of Event Calculus programs. However, a serious obstacle that prevents XHAIL from being widely applicable as a machine learning system for event recognition is scalability. XHAIL scales poorly, partly because of the increased computational complexity of abduction, which lies at the core of its functionality, and partly because of the combinatorial complexity of learning whole theories, which may result in an intractable search space. In what follows, we use the XHAIL machinery to develop an incremental algorithm that scales to large volumes of sequential data, typical of event-based applications.

2.6.3 ILED: Incremental Learning of Event Definitions

We assume a database $\mathcal{E}$ of examples, called historical memory, storing examples presented over time. Initially $\mathcal{E} = \emptyset$. At time n, ILED (Incremental Learning of Event Definitions) is presented with a hypothesis H_n such that $\mathsf{EC} \cup H_n \vDash \mathcal{E}$, where EC denotes the axioms of the Event Calculus. At time n ILED is also presented with a new set of examples w_n. The goal is to revise H_n to a hypothesis H_{n+1}, so that $\mathsf{EC} \cup H_{n+1} \vDash \mathcal{E} \cup w_n$.

A main challenge of adopting a full memory approach like the one described above is to scale it up to a growing size of experience. This is in line with a key requirement of incremental learning where "the incorporation of experience into memory during learning should be computationally efficient, that is, theory revision must be efficient in fitting new incoming observations" (Langley, 1995; Mauro et al., 2005). In the stream processing literature, the number of passes over a stream of data is often used as a measure of the efficiency of algorithms (Li et al., 2004; Li and Lee, 2009). In this spirit, the main contribution of ILED, in addition to scaling up XHAIL, is that it adopts a "single-pass" theory revision strategy, that is, a strategy that requires at most one pass over $\mathcal{E}$ in order to compute H_{n+1} from H_n.

A single-pass revision strategy is far from trivial. For instance, the addition of a new clause C in response to a set of new examples w_n implies that H_n must be checked throughout $\mathcal{E}$. In case C covers some negative examples in $\mathcal{E}$ it should be specialised, which in turn may affect the initial coverage of C in w_n. If the specialisation results in the rejection of positive examples in w_n, extra clauses must be generated and added to H_n, in order to retrieve the lost positives, and these clauses should be again checked for correctness in $\mathcal{E}$. This process continues until a hypothesis H_{n+1} is found, that accounts for all the examples in $\mathcal{E} \cup w_n$. In general, this requires several passes over the historical memory.

Since experience may grow over time to an extent that is impossible to maintain in the working memory, we follow an external memory approach (Biba et al., 2008). This implies that the learner does not have access to all past experience as a whole, but to independent sets of training data, in the form of *sliding windows*.

Revisions are implemented by means of revision operators that act upon the theory at hand and later the examples it accounts for. Revision operators may be generalisation or specialisation operators. The former increase the example coverage of the revised theory, in order to account for uncovered positive examples, while the latter decrease example coverage, in order to reject negative examples. ILED does not utilise revision operators that retract knowledge, such as the deletion of clauses or antecedents are excluded, due to the exponential cost of backtracking in the historical memory (Badea, 2001). The supported revision operators are thus:

- Addition of new clauses.
- Refinement of existing clauses, i.e., replacement of an existing clause with one or more specialisations of that clause.

To cover more positive examples, ILED adds `initiatedAt` clauses and refines `terminatedAt` clauses, while to reject negative examples, it adds `terminatedAt` clauses and refines `initiatedAt` clauses.
Figure 2.2 illustrates ILED's theory revision process with a simple example. New clauses are generated by generalising a Kernel Set of the incoming window, as shown in Figure 2.2, where a `terminatedAt`

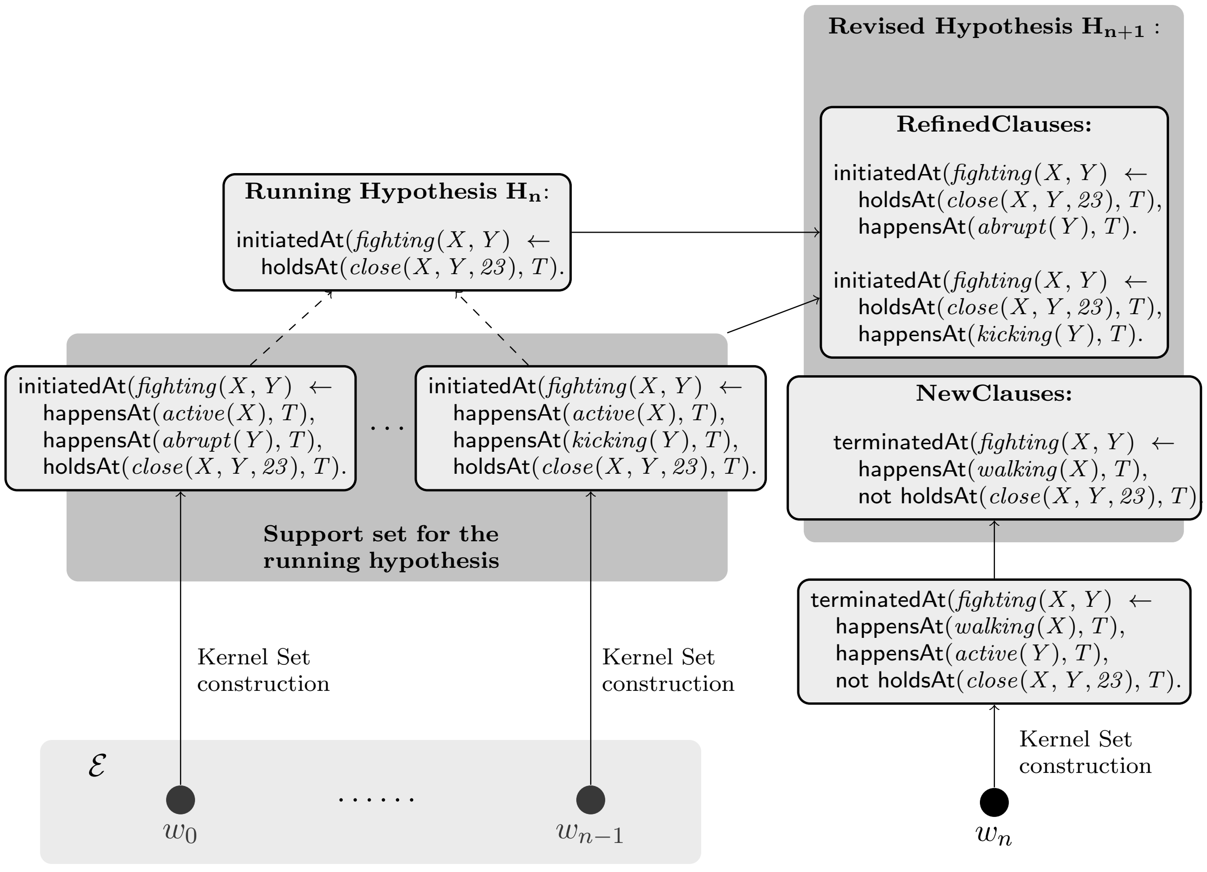
Revised Hypothesis $\mathbf{H_{n+1}}$:
RefinedClauses:
initiatedAt(fighting(X, Y) ← holdsAt(close(X, Y, 23), T), happensAt(abrupt(Y), T).
initiatedAt(fighting(X, Y) ← holdsAt(close(X, Y, 23), T), happensAt(kicking(Y), T).
Running Hypothesis $\mathbf{H_n}$:
initiatedAt(fighting(X, Y) ← holdsAt(close(X, Y, 23), T).
initiatedAt(fighting(X, Y) ← happensAt(active(X), T), happensAt(abrupt(Y), T), holdsAt(close(X, Y, 23), T).
initiatedAt(fighting(X, Y) ← happensAt(active(X), T), happensAt(kicking(Y), T), holdsAt(close(X, Y, 23), T).
NewClauses:
terminatedAt(fighting(X, Y) ← happensAt(walking(X), T), not holdsAt(close(X, Y, 23), T)
Support set for the running hypothesis
terminatedAt(fighting(X, Y) ← happensAt(walking(X), T), happensAt(active(Y), T), not holdsAt(close(X, Y, 23), T).
Kernel Set construction
Kernel Set construction
Kernel Set construction
$\mathcal{E}$
w_0
w_{n-1}
w_n

Figure 2.2: Revision of a hypothesis H_n in response to a new example window w_n. $\mathcal{E}$ represents the historical memory of examples.

clause is generated from the new window w_n. Moreover, to facilitate refinement of existing clauses, each clause in the running hypothesis is associated with a memory of the examples it covers throughout $\mathcal{E}$, in the form of a "bottom program", which we call *support set*. The support set is constructed gradually, as new example windows arrive. It serves as a refinement search space, as shown in Figure 2.2, where the single clause in the running hypothesis H_n is refined w.r.t. the incoming window w_n into two specialisations. Each such specialisation results by adding to the initial clause one antecedent from the two support set clauses which are presented in Figure 2.2. The revised hypothesis H_{n+1} is constructed from the refined clauses and the new ones, along with the preserved clauses of H_n, if any.

The intuition behind the support set stems from the XHAIL methodology. Given a set of examples E, XHAIL learns a hypothesis by generalising a Kernel Set K of these examples. E may be too large to process in one go and a possible solution is to partition E in smaller example sets $E_1, \dots, E_n$ and try to learn a hypothesis that accounts for the whole of E, by gradually revising an initial hypothesis H_1 acquired from E_1. In this process of progressive revisions, a compressive memory of "small" Kernel sets of $E_1, \dots, E_n$ may be used as a surrogate for the fact that one is not able to reason with the whole Kernel Set K. This is the role of the support set.

By means of this memory, and as far as clause refinement is concerned, ILED is able to repair problems locally, i.e., in a single example window, without affecting coverage in the parts of the historical memory where the clause under refinement has been previously checked. In more detail, given a hypothesis clause C and a window w where C must be refined, and denoting by $\mathcal{E}_{tested}(C)$, the part of $\mathcal{E}$

where C has already been tested, ILED refines C so that its refinement covers all positive examples that C covers in $\mathcal{E}_{tested}(C)$, making the task of checking $\mathcal{E}_{tested}(C)$ in response to the refinement redundant.

There are two key features of ILED that contribute towards its scalability, which are both due to the support set: First, the re-processing of past experience is necessary only in the case where new clauses are generated and is redundant in the case where a revision consists of refinements of existing clauses. Second, re-processing of past experience requires a single pass over the historical memory, meaning that it suffices to "re-see" each past window exactly once to ensure that the output revised hypothesis H_{n+1} is complete & consistent w.r.t. the entire historical memory.

2.7 Experimental evaluation of ILED

In this section, we present experimental results from two real-world applications: Activity recognition, using real data from the benchmark CAVIAR video surveillance dataset[2], as well as large volumes of synthetic CAVIAR data; and City Transport Management (CTM) using data from the PRONTO[3] project.

Part of our experimental evaluation aims to compare ILED with XHAIL. To achieve this aim we had to implement XHAIL, because the original implementation was not publicly available until recently (Bragaglia and Ray, 2014). All experiments were conducted on a 3.4 GHz Linux machine with 12 GB of RAM. The algorithms were implemented in Python, using the Clingo[4] Answer Set Solver (Gebser et al., 2012) as the main reasoning component, and a Mongodb[5] NoSQL database for the historical memory of the examples.

2.7.1 Activity Recognition

In activity recognition, our goal is to learn definitions of CEs, such as *fighting, moving* and *meeting*, from streams of SDEs like *walking, standing, active* and *abrupt*, as well as spatio-temporal knowledge. We use the benchmark CAVIAR dataset for experimentation. Details on the CAVIAR dataset and more information about activity recognition applications may be found in (Artikis et al., 2010b). Consider for instance the following definition of the *fighting* CE:

$$\begin{aligned}&\mathtt{initiatedAt}(\mathtt{fighting}(X,Y),T) \leftarrow\\ &\quad \mathtt{happens}(\mathtt{active}(X),T),\\ &\quad \mathtt{not}\ \mathtt{happens}(\mathtt{inactive}(Y),T),\\ &\quad \mathtt{holdsAt}(\mathtt{close}(X,Y,\mathit{23}),T).\end{aligned} \tag{2.8}$$

$$\begin{aligned}&\mathtt{initiatedAt}(\mathtt{fighting}(X,Y),T) \leftarrow\\ &\quad \mathtt{happens}(\mathtt{abrupt}(X),T),\\ &\quad \mathtt{not}\ \mathtt{happens}(\mathtt{inactive}(Y),T),\\ &\quad \mathtt{holdsAt}(\mathtt{close}(X,Y,\mathit{23}),T).\end{aligned} \tag{2.9}$$

$$\begin{aligned}&\mathtt{terminatedAt}(\mathtt{fighting}(X,Y),T) \leftarrow\\ &\quad \mathtt{happens}(\mathtt{walking}(X),T),\\ &\quad \mathtt{not}\ \mathtt{holdsAt}(\mathtt{close}(X,Y,\mathit{23}),T).\end{aligned} \tag{2.10}$$

$$\begin{aligned}&\mathtt{terminatedAt}(\mathtt{fighting}(X,Y),T) \leftarrow\\ &\quad \mathtt{happens}(\mathtt{running}(X),T),\\ &\quad \mathtt{not}\ \mathtt{holdsAt}(\mathtt{close}(X,Y,\mathit{23}),T).\end{aligned} \tag{2.11}$$

Clause (2.8) dictates that a period of time for which two persons X and Y are assumed to be fighting is initiated at time T if one of these persons is *active*, the other one is not *inactive* and their distance is

[2] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/
[3] http://www.ict-pronto.org/
[4] http://potassco.sourceforge.net/
[5] http://www.mongodb.org/

smaller than 23 pixel positions. Clause (2.9) states that *fighting* is initiated between two people when one of them moves *abruptly*, the other is not *inactive*, and the two persons are sufficiently close. Clauses (2.10) and (2.11) state that *fighting* is terminated between two people when one of them walks or runs away from the other.

CAVIAR contains noisy data mainly due to human errors in the annotation (List et al., 2005; Artikis et al., 2010b). Thus, for the experiments we manually selected a noise-free subset of CAVIAR. The resulting dataset consists of 1000 examples (that is, data for 1000 distinct time points) concerning the CEs *moving*, *meeting* and *fighting*. These data, selected from different parts of the CAVIAR dataset, were combined into a continuous annotated stream of narrative atoms, with time ranging from 0 to 1000.

In addition to the real data, we generated synthetic data on the basis of the manually-developed CAVIAR event definitions described in (Artikis et al., 2010b). In particular, streams of SDEs concerning four different persons were created randomly and were then classified using the rules of (Artikis et al., 2010b). The final dataset was obtained by generating negative supervision via the closed world assumption and appropriately pairing the supervision with the narrative. The generated data consists of approximately 10^5 examples, which amounts to 100 MB of data.

The synthetic data is much more complex than the real CAVIAR data. This is due to two main reasons: First, the synthetic data includes significantly more initiations and terminations of a CE, thus much larger learning effort is required to explain it. Second, in the synthetic dataset more than one CE may be initiated or terminated at the same time point. This results in Kernel Sets with more clauses, which are hard to generalise simultaneously.

ILED vs XHAIL

The purpose of this experiment was to assess whether ILED can efficiently generate hypotheses comparable in size and predictive quality to those of XHAIL. To this end, we compared both systems on real and synthetic data using 10-fold cross validation with replacement. For the real data, 90% of randomly selected examples, from the total of 1000 were used for training, while the remaining 10% was retained for testing. At each run, the training data were presented to ILED in example windows of sizes 10, 50, 100. The data were presented in one batch to XHAIL. For the synthetic data, 1000 examples were randomly sampled at each run from the dataset for training, while the remaining data were retained for testing. Similar to the real data experiments, ILED operated on windows of sizes of 10, 50, 100 examples and XHAIL on a single batch.

Table 2.3 presents the experimental results. Training times are significantly higher for XHAIL, due to the increased complexity of generalising Kernel Sets that account for the whole set of the presented examples at once. These Kernel Sets consisted, on average, of 30 to 35 16-literal clauses, in the case of the real data, and 60 to 70 16-literal clauses in the case of the synthetic data. In contrast, ILED had to deal with much smaller Kernel Sets. The complexity of abductive search affects ILED as well, as the size of the input windows grows. ILED handles the learning task relatively well (in approximately 30 seconds) when the examples are presented in windows of 50 examples, but the training time increases almost 15 times if the window size is doubled.

Concerning the size of the produced hypothesis, the results show that in the case of real CAVIAR data, the hypotheses constructed by ILED are comparable in size with a hypothesis constructed by XHAIL. In the case of synthetic data, the hypotheses returned by both XHAIL and ILED were significantly more complex. Note that for ILED the hypothesis size decreases as the window size increases. This is reflected in the number of revisions that ILED performs, which is significantly smaller when the input comes in larger batches of examples. In principle, the richer the input, the better the hypothesis

	ILED			XHAIL
Real CAVIAR data	$G = 10$	$G = 50$	$G = 100$	$G = 900$
Training Time (sec)	34.15 (± 6.87)	23.04 (± 13.50)	286.74 (±98.87)	1560.88 (±4.24)
Revisions	11.2 (± 3.05)	9.1 (± 0.32)	5.2 (±2.1)	–
Hypothesis size	17.82 (± 2.18)	17.54 (± 1.5)	17.5 (±1.43)	15 (±0.067)
Precision	98.713 (± 0.052)	99.767 (± 0.038)	99.971 (±0.041)	99.973 (±0.028)
Recall	99.789 (± 0.083)	99.845 (± 0.32)	99.988 (±0.021)	99.992 (±0.305)
Synthetic CAVIAR data	$G = 10$	$G = 50$	$G = 100$	$G = 1000$
Training Time (sec)	38.92 (± 9.15)	33.87 (± 9.74)	468 (±102.62)	21429 (±342.87)
Revisions	28.7 (± 9.34)	15.4 (± 7.5)	12.2 (±6.23)	–
Hypothesis size	143.52 (± 19.14)	138.46 (± 22.7)	126.43 (±15.8)	118.18 (±14.48)
Precision	55.713 (± 0.781)	57.613 (± 0.883)	63.236 (±0.536)	63.822 (±0.733)
Recall	68.213 (± 0.873)	71.813 (± 0.756)	71.997 (±0.518)	71.918 (±0.918)

Table 2.3: Comparison of ILED and XHAIL. G is the window granularity.

that is initially acquired, and consequently, the less the need for revisions in response to new training instances. There is a trade-off between the window size (thus the complexity of the abductive search) and the number of revisions. A small number of revisions on complex data (i.e., larger windows) may have a greater total cost in terms of training time, as compared to a greater number of revisions on simpler data (i.e., smaller windows). For example, in the case of window size 100 for the real CAVIAR data, ILED performs 5 revisions on average and requires significantly more time than in the case of a window size 50, where it performs 9 revisions on average. On the other hand, training times for windows of size 50 are slightly better than those obtained when the examples are presented in smaller windows of size 10. In this case, the "unit cost" of performing revisions w.r.t. a single window are comparable between windows of size 10 and 50. Thus the overall cost in terms of training time is determined by the total number of revisions, which is greater in the case of window size 10.

Concerning predictive quality, the results indicate that ILED's precision and recall scores are comparable to those of XHAIL. For larger input windows, precision and recall are almost the same as those of XHAIL. This is because ILED produces better hypotheses from larger input windows. Precision and recall are smaller in the case of synthetic data for both systems, because the testing set in this case is much larger and complex than in the case of real data.

ILED Scalability

The purpose of this experiment was to assess the scalability of ILED. The experimental setting was as follows: Sets of examples of varying sizes were randomly sampled from the synthetic dataset. Each such example set was used as a training set in order to acquire an initial hypothesis using ILED. Then a new window which did not satisfy the hypothesis at hand was randomly selected and presented to ILED, which subsequently revised the initial hypothesis in order to account for both the historical memory (the initial training set) and the new evidence. For historical memories ranging from 10^3 to 10^5 examples, a new training window of size 10, 50 and 100 was selected from the whole dataset. The process was repeated ten times for each different combination of historical memory and new window size. Figure 2.3 presents the average revision times. The revision times for new window sizes of 10 and 50 examples are very close and therefore omitted to avoid clutter. The results indicate that revision time grows polynomially in the size of the historical memory.

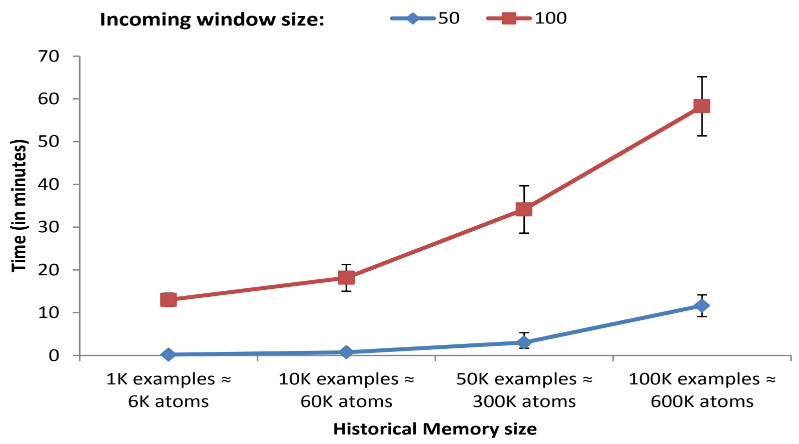

Figure 2.3: Average times needed for ILED to revise an initial hypothesis in the face of new evidence presented in windows of size 10, 50 and 100 examples. The initial hypothesis was obtained from a training set of varying size (1K, 10K, 50K and 100K examples) which subsequently served as the historical memory.

2.7.2 City Transport Management

In this section we present experimental results from the domain of City Transport Management (CTM). We use data from the PRONTO[6] project. In PRONTO, the goal was to inform the decision-making of transport officials by recognising CEs related to the punctuality of a public transport vehicle (bus or tram), passenger/driver comfort and safety. These CEs were requested by the public transport control centre of Helsinki, Finland, in order to support resource management. Low-level events were provided by sensors installed in buses and trams, reporting on changes in position, acceleration/deceleration, in-vehicle temperature, noise level and passenger density. At the time of the project, the available datasets included only a subset of the anticipated SDE types as some SDE detection components were not functional. For the needs of the project, therefore, a synthetic dataset was generated. The synthetic PRONTO data has proven to be considerably more challenging for event recognition than the real data, and therefore we chose the former for evaluating ILED (Artikis et al., 2014). The CTM dataset contains $5 \cdot 10^4$ examples, which amount approximately to 70 MB of data.

In contrast to the activity recognition application, the manually developed CE definitions of CTM that were used to produce the annotation for learning, form a hierarchy. In these hierarchical event definitions, it is possible to define a function level that maps all CEs to non-negative integers as follows: A level-1 event is defined in terms of SDEs (input data) only. An level-n event is defined in terms of at least one level-$n-1$ event and a possibly empty set of SDEs and CEs of level below $n-1$. Hierarchical definitions are significantly more complex to learn as compared to non-hierarchical ones. This is because initiations and terminations of events in the lower levels of the hierarchy appear in the bodies of event definitions in the higher levels of the hierarchy, hence all target definitions must be learnt simultaneously. As we show in the experiments, this has a striking effect on the required learning effort. A solution for simplifying the learning task is to utilise knowledge about the domain (the hierarchy), learn event definitions separately, and use the acquired theories from lower levels of the event hierarchy as non-revisable background knowledge when learning event definitions for the higher levels. Below is a fragment of the CTM event hierarchy:

[6]http://www.ict-pronto.org/

	ILED		XHAIL
	$G = 5$	$G = 10$	$G = 20$
Training Time (**hours**)	1.35 (± 0.17)	1.88 (± 0.13)	4.35 (±0.25)
Hypothesis size	28.32 (± 1.19)	24.13 (± 2.54)	24.02 (±0.23)
Revisions	14.78 (± 2.24)	13.42 (± 2.08)	–
Precision	63.344 (± 5.24)	64.644 (± 3.45)	66.245 (± 3.83)
Recall	59.832 (± 7.13)	61.423 (± 5.34)	62.567 (± 4.65)

Table 2.4: Comparative performance of ILED and XHAIL on selected subsets of the CTM dataset each containing 20 examples. G is the granularity of the windows.

$$\begin{aligned}&\texttt{initiatedAt}(\texttt{punctuality}(Id, nonPunctual), T) \leftarrow\\&\quad \texttt{happens}(\texttt{stopEnter}(Id, StopId, late), T).\end{aligned} \tag{2.12}$$

$$\begin{aligned}&\texttt{initiatedAt}(\texttt{punctuality}(Id, nonPunctual), T) \leftarrow\\&\quad \texttt{happens}(\texttt{stopLeave}(Id, StopId, early), T).\end{aligned} \tag{2.13}$$

$$\begin{aligned}&\texttt{terminatedAt}(\texttt{punctuality}(Id, nonPunctual), T) \leftarrow\\&\quad \texttt{happens}(\texttt{stopEnter}(Id, StopId, early), T).\end{aligned} \tag{2.14}$$

$$\begin{aligned}&\texttt{terminatedAt}(\texttt{punctuality}(Id, nonPunctual), T) \leftarrow\\&\quad \texttt{happens}(\texttt{stopEnter}(Id, StopId, scheduled), T).\end{aligned} \tag{2.15}$$

$$\begin{aligned}&\texttt{initiatedAt}(\texttt{drivingQuality}(Id, low), T) \leftarrow\\&\quad \texttt{initiatedAt}(\texttt{punctuality}(Id, nonPunctual), T),\\&\quad \texttt{holdsAt}(\texttt{drivingStyle}(Id, unsafe), T).\end{aligned} \tag{2.16}$$

$$\begin{aligned}&\texttt{initiatedAt}(\texttt{drivingQuality}(Id, low), T) \leftarrow\\&\quad \texttt{initiatedAt}(\texttt{drivingStyle}(Id, unsafe), T),\\&\quad \texttt{holdsAt}(\texttt{punctuality}(Id, nonPunctual), T).\end{aligned} \tag{2.17}$$

$$\begin{aligned}&\texttt{terminatedAt}(\texttt{drivingQuality}(Id, low), T) \leftarrow\\&\quad \texttt{terminatedAt}(\texttt{punctuality}(Id, nonPunctual), T).\end{aligned} \tag{2.18}$$

$$\begin{aligned}&\texttt{terminatedAt}(\texttt{drivingQuality}(Id, low), T) \leftarrow\\&\quad \texttt{terminatedAt}(\texttt{drivingStyle}(Id, unsafe), T).\end{aligned} \tag{2.19}$$

Clauses (2.12) and (2.13) state that a period of time for which vehicle Id is said to be *non-punctual* is initiated if it enters a stop later, or leaves a stop earlier than the scheduled time. Clauses (2.14) and (2.15) state that the period for which vehicle Id is said to be non-punctual is terminated when the vehicle arrives at a stop earlier than, or at the scheduled time. The definition of non-punctual vehicle uses two SDEs, $stopEnter$ and $stopLeave$.

Clauses (2.16)-(2.19) define *low driving quality*. Essentially, driving quality is said to be low when the driving style is unsafe and the vehicle is non-punctual. Driving quality is defined in terms of CEs (we omit the definition of driving style to save space). Therefore, the bodies of the clauses defining driving quality include initiatedAt and terminatedAt literals.

ILED vs XHAIL

In this experiment, we tried to learn simultaneously definitions for all target concepts, a total of nine interrelated CEs, seven of which are level-1, one is level-2 and one is level-3. According to the employed language bias, each such CE must be learnt, while at the same time it may be present in the body of another CE in the form of (potentially negated) `holdsAt`/2, `initiatedAt`/2, or `terminatedAt` predicate. The total number of SDEs involved is 22.

We used tenfold cross validation with replacement, on small amounts of data, due to the complexity of the learning task. In each run of the cross validation, we randomly sampled 20 examples from the CTM dataset, 90% of which was used for training and 10% was retained for testing. This example size was selected after experimentation, in order for XHAIL to be able to perform in an acceptable time frame. Each sample consisted of approximately 150 atoms (narrative and annotation). The examples were given to ILED in windows of granularity 5 and 10, and to XHAIL in one batch. Table 2.4 presents the average training times, hypothesis size, number of revisions, precision and recall.

ILED took on average 1-2 hours to complete the learning task, for windows of 5 and 10 examples, while XHAIL required more than 4 hours on average to learn hypotheses from batches of 20 examples. Compared to activity recognition, the learning setting requires larger Kernel Set structures that are hard to reason with. An average Kernel Set generated from a batch of just 20 examples consisted of approximately 30-35 clauses, with 60-70 literals each.

Like the activity recognition experiments, precision and recall scores for ILED are comparable to those of XHAIL, with the latter being slightly better. Unlike the activity recognition experiments, precision and recall had a large diversity between different runs. Due to the complexity of the CTM dataset, the constructed hypotheses had a large diversity, depending on the random samples that were used for training. For example, some CE definitions were unnecessarily lengthy and difficult to be understood by a human expert. On the other hand, some level-1 definitions could in some runs of the experiment, be learnt correctly even from a limited amount of data. Such definitions are fairly simple, consisting of one initiation and one termination rule, with one body literal in each case.

This experiment demonstrates several limitations of learning in large and complex applications. The complexity of the domain increases the intensity of the learning task, which in turn makes training times forbidding, even for small amount of data such as 20 examples (approximately 150 atoms). This forces one to process small sets of examples at time, which in complex domains like CTM, results to over-fitted theories and rapid increase in hypothesis size.

Learning With Hierarchical Bias

In an effort to improve the experimental results, we utilised domain knowledge about the event hierarchy in CTM and attempted to learn CEs in different levels separately. To do so, we had to learn a complete definition from the entire dataset for a CE, before utilising it as background knowledge in the learning process of a higher-level event. To facilitate the learning task further, we also used expert knowledge about the relation between specific SDEs and CEs, excluding from the language bias mode declarations which were irrelevant to the CE that is being learnt at each time.

The experimental setting was therefore as follows: Starting from the level-1 target events, we processed the whole CTM dataset in windows of 10, 50 and 100 examples with ILED. Each CE was learnt independently of the others. Once complete definitions for all level-1 CEs were constructed, they were added to the background knowledge. Then we proceeded with learning the definition for the single level-2 event. Finally, after successfully constructing the level-2 definition, we performed learning in the top-level of the hierarchy, using the previously constructed level-1 and level-2 event definitions as background knowledge. We did not attempt a comparison with XHAIL, since due to the amounts of data in CTM, the latter is not able to operate on the entire dataset.

Table 2.5 presents the results. For level-1 events, scores are presented as minimum-maximum pairs. For instance, the training times for level-1 events with windows of 10 examples, ranges from 4.46 to

	ILED		
level-1	$G = 10$	$G = 50$	$G = 100$
Training Time (min)	4.46 – 4.88	5.78 – 6.44	6.24 – 6.88
Revisions	2 – 11	2 – 9	2 – 9
Hypothesis size	4 – 18	4 – 16	4 – 16
Precision	100%	100%	100%
Recall	100%	100%	100%
level-2	$G = 10$	$G = 50$	$G = 100$
Training Time (min)	8.76	9.14	9.86
Revisions	24	17	17
Hypothesis size	31	27	27
Precision	100%	100%	100%
Recall	100%	100%	100%
level-3	$G = 10$	$G = 50$	$G = 100$
Training Time (min)	5.78	6.14	6.78
Revisions	6	5	5
Hypothesis size	13	10	10
Precision	100%	100%	100%
Recall	100%	100%	100%

Table 2.5: ILED with hierarchical bias.

4.88 minutes. Levels 2 and 3 have just one definition each, therefore Table 2.5 presents the respective scores from each run. Training times, hypothesis sizes and overall numbers of revisions are comparable for all levels of the event hierarchy. Level-1 event definitions were the easiest to acquire, with training times ranging approximately between 4.50 to 7 minutes. This was expected since clauses in level-1 definitions are significantly simpler than level-2 and level-3 ones. The level-2 event definition was the hardest to construct with training times ranging between 8 and 10 minutes, while a significant number of revisions was required for all window granularities. The definition of this CE (*drivingStyle*) is relatively complex, in contrast to the simpler level-3 definition, for which training times are comparable to the ones for level-1 events.

The largest parts of training times were dedicated to checking an already correct definition against the part of the dataset that had not been processed yet. That is, for all target events, ILED converged to a complete definition relatively quickly, i.e., in approximately 1.5 to 3 minutes after the initiation of the learning process. From that point on, the extra time was spent on testing the hypothesis against the new incoming data.

Window granularity slightly affects the produced hypothesis for all target CEs. Indeed, the definitions constructed with windows of 10 examples are slightly larger than the ones constructed with larger window sizes of 50 and 100 examples. Notably, the definitions constructed with windows of granularity 50 and 100, were found concise, meaningful and very close to the actual hand-crafted rules that were utilised in PRONTO.

3 Conclusions and Future Work

In this document, we presented techniques for reasoning and learning composite event (CE) definitions under uncertainty and large amounts of data. Uncertainty is unavoidable in real-world applications and may seriously compromise the accuracy of event recognition and forecasting. Furthermore, due to the dynamic nature of the SPEEDD use cases, machine learning must deal with large amounts of data that continuously evolve. As a result, the knowledge base of event definitions may need to be refined or it may need to be enhanced with new definitions.

To overcome the issues that arise from uncertainty, we combine probabilistic with logic-based modelling by employing the Event Calculus formalism and the probabilistic relational framework of Markov Logic Networks (MLN). This combination has the advantage of expressing formally and declaratively CE definitions and domain background knowledge, while the probabilistic modelling allows to perform probabilistic inference under uncertainty. Furthermore, we outline machine learning techniques in order to perform parameter estimation and structure learning from annotated data.

To deal with large amounts of data in machine learning, we presented an incremental ILP system (ILED) for machine learning knowledge bases for event recognition, in the form of Event Calculus theories. ILED combines techniques from non-monotonic ILP and in particular, the XHAIL algorithm, with theory revision. It acquires an initial hypothesis from the first available piece of data, and revises this hypothesis as new data arrive. Revisions account for all accumulated experience. The main contribution of ILED is that it scales-up XHAIL to large volumes of sequential data with a time-like structure, typical of event-based applications. By means of a compressive memory structure that supports clause refinement, ILED has a scalable, single-pass revision strategy, thanks to which the cost of theory revision grows as a tractable function of the perceived experience. In this work, ILED was evaluated on an activity recognition application and a transport management application. The results indicate that ILED is significantly more efficient than XHAIL, without compromising the quality of the generated hypothesis in terms of predictive accuracy and hypothesis size. Moreover, ILED scales adequately to large data volumes which XHAIL cannot handle.

Below we outline the research agenda for the second year of SPEEDD.

- We will implement the max-margin parameter estimation for the probabilistic Event Calculus formalism, in order to estimate the weights of the event definitions (e.g., definitions of traffic congestion, credit card fraudulent activity, etc.) from annotated data. The learnt weights express the confidence value of each definition relative to the other event definitions in the knowledge base, in order to improve the event recognition accuracy — e.g., degrease the number of false positives and false negatives.

- We are planning to combine the probabilistic and logic-based modelling of the Event Calculus and Markov Logic Networks with the incremental learning techniques of the ILED method, in order to efficiently refine, as all as introduce new event definitions, and thus improve the recognition performance — e.g., increase the recall.
- We will evaluate the accuracy of the learnt parameters and definitions for both fraud and traffic management applications.

For the implementation of the machine learning algorithms we are planning to use and extend the functionality of the open-source Markov Logic Networks framework LoMRF[1].

[1] https://github.com/anskarl/LoMRF

Bibliography

H. Ade and M. Denecker. AILP: Abductive inductive logic programming. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1995.

U. Apsel and R. I. Brafman. Lifted MEU by Weighted Model Counting. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*. AAAI Press, 2012.

A. Artikis, M. Sergot, and G. Paliouras. A Logic Programming Approach to Activity Recognition. In *Proceedings of the 2nd International Workshop on Events in Multimedia (EiMM)*, pages 3–8. ACM, 2010a.

A. Artikis, A. Skarlatidis, and G. Paliouras. Behaviour recognition from video content: A logic programming approach. *International Journal on Artificial Intelligence Tools*, 19(2):193–209, 2010b.

A. Artikis, M. Sergot, and G. Paliouras. An event calculus for event recognition. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2014.

D. Athakravi, D. Corapi, K. Broda, and A. Russo. Learning through hypothesis refinement using answer set programming. In *Proceedings of the 23rd International Conference of Inductive Logic Programming (ILP 2013)*, 2013.

L. Badea. A refinement operator for theories. In *Inductive Logic Programming*, pages 1–14. Springer, 2001.

M. Biba, T. M. A. Basile, S. Ferilli, and F. Esposito. Improving scalability in ilp incremental systems. 2008.

M. Biba, F. Xhafa, F. Esposito, and S. Ferilli. Engineering SLS Algorithms for Statistical Relational Models. In *Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, pages 502–507. IEEE Computer Society, 2011.

E. Boros and P. L. Hammer. Pseudo-Boolean optimization. *Discrete Applied Mathematics*, 123(1–3): 155–225, 2002. ISSN 0166-218X. doi: http://dx.doi.org/10.1016/S0166-218X(01)00341-9. URL http://www.sciencedirect.com/science/article/pii/S0166218X01003419.

S. Bragaglia and O. Ray. Nonmonotonic learning in large biological networks. In *Proc. 24th Int. Conf. on Inductive Logic Programming*, 2014.

M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 994–999. IEEE Computer Society, 1997.

W. L. Buntine. Operations for Learning with Graphical Models. *Journal of Artificial Intelligence Research (JAIR)*, 2:159–225, 1994.

G. Casella and E. I. George. Explaining the Gibbs Sampler. *The American Statistician*, 46(3):167–174, 1992. ISSN 00031305. doi: 10.2307/2685208. URL http://dx.doi.org/10.2307/2685208.

M. Cattafi, E. Lamma, F. Riguzzi, and S. Storari. Incremental declarative process mining. *Smart Information and Knowledge Management*, pages 103–127, 2010.

K. L. Clark. Negation as Failure. In *Logic and Data Bases*, pages 293–322, 1977.

M. Collins. Discriminative training methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 10, pages 1–8. Association for Computational Linguistics, 2002.

D. Corapi, A. Russo, and E. C. Lupu. Inductive logic programming as abductive search. In *Technical Communications of the 26th International Conference on Logic Programming (ICLP)*, 2010.

D. Corapi, A. Russo, and E. Lupu. Inductive logic programming in answer set programming. In *ILP*, 2011.

P. Damlen, J. Wakefield, and S. Walker. Gibbs sampling for Bayesian non-conjugate and hierarchical models by using auxiliary variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(2):331–344, 1999.

G. V. den Broeck, N. Taghipour, W. Meert, J. Davis, and L. de Raedt. Lifted Probabilistic Inference by First-Order Knowledge Compilation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2178–2185. IJCAI/AAAI, 2011.

M. Denecker and A. Kakas. Abduction in logic programming. *Computational Logic: Logic Programming and Beyond*, 2407:402–437, 2002.

T. G. Dietterich, P. Domingos, L. Getoor, S. Muggleton, and P. Tadepalli. Structured machine learning: the next ten years. *Machine Learning*, 73:3–23, 2008.

P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.

K. Eshghi and R. Kowalski. Abduction compared with negation by failure. In *6th International Conference on Logic Programming*, 1989.

F. Esposito, G. Semeraro, N. Fanizzi, and S. Ferilli. Multistrategy theory revision: Induction and abduction in inthelex. *Machine Learning*, 28(1-2):133–156, 2000.

O. Etzion and P. Niblett. *Event Processing in Action*. Manning Publications Company, 2010. ISBN 978-1-935182-21-4.

D. Furcy and S. Koenig. Limited discrepancy beam search. In L. P. Kaelbling and A. Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 125–131. Professional Book Center, 2005. URL http://www.ijcai.org/papers/0596.pdf.

A. Gal, S. Wasserkrug, and O. Etzion. Event Processing over Uncertain Data. In S. Helmer, A. Poulovassilis, and F. Xhafa, editors, *Reasoning in Event-Based Distributed Systems*, volume 347 of *Studies in Computational Intelligence*, pages 279–304. Springer, 2011. ISBN 978-3-642-19723-9.

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

V. Gogate and P. Domingos. Probabilistic Theorem Proving. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 256–265. AUAI Press, 2011.

S. Gong and T. Xiang. Recognition of Group Activities using Dynamic Probabilistic Networks. In *Proceedings of the 9th International Conference on Computer Vision (ICCV)*, volume 2, pages 742–749. IEEE Computer Society, 2003.

J. Gonzalez, Y. Low, C. Guestrin, and D. R. O'Hallaron. Distributed Parallel Inference on Large Factor Graphs. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 203–212. AUAI Press, 2009.

H. H. Hoos and T. Stützle. *Stochastic Local Search: Foundations & Applications*. Elsevier / Morgan Kaufmann, 2004. ISBN 1-55860-872-9.

T. N. Huynh and R. J. Mooney. Max-Margin Weight Learning for Markov Logic Networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, volume 5781 of *Lecture Notes in Computer Science*, pages 564–579. Springer, 2009.

T. N. Huynh and R. J. Mooney. Online Max-Margin Weight Learning for Markov Logic Networks. In *Proceedings of the 11th SIAM International Conference on Data Mining (SDM11)*, pages 642–651, Mesa, Arizona, USA, April 2011a.

T. N. Huynh and R. J. Mooney. Online structure learning for markov logic networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2011)*, volume 2, pages 81–96, September 2011b. URL http://www.cs.utexas.edu/users/ai-lab/?huynh:ecml11.

T. Joachims. A support vector method for multivariate performance measures. In *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, pages 377–384. ACM Press, 2005. URL http://doi.acm.org/10.1145/1102351.1102399.

T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009. ISSN 0885-6125. doi: 10.1007/s10994-009-5108-8. URL http://dx.doi.org/10.1007/s10994-009-5108-8.

A. Kakas and P. Mancarella. Generalised stable models: A semantics for abduction. In *ninth European Conference on Artificial Intelligence (ECAI-90)*, pages 385–391, 1990.

A. Kakas, R. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2:719–770, 1993.

H. Kautz, B. Selman, and Y. Jiang. A General Stochastic Approach to Solving Problems with Hard and Soft Constraints. In D. Gu, J. Du, and P. Pardalos, editors, *The Satisfiability Problem: Theory and Applications*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 573–586. AMS, 1997.

K. Kersting. Lifted probabilistic inference. In L. de Raedt, C. Bessière, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, and P. J. F. Lucas, editors, *20th European Conference on Artificial Intelligence, Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 33–38. IOS Press, 2012.

K. Kersting, B. Ahmadi, and S. Natarajan. Counting Belief Propagation. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 277–284. AUAI Press, 2009.

S. Kok and P. Domingos. Learning the structure of Markov logic networks. In *Proceedings of the 22nd international conference on Machine learning*, pages 441–448. ACM, 2005.

S. Kok and P. Domingos. Learning markov logic network structure via hypergraph lifting. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 505–512. ACM, 2009.

S. Kok and P. Domingos. Learning markov logic networks using structural motifs. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 551–558. Citeseer, 2010.

R. Kowalski and M. Sergot. A Logic-based Calculus of Events. *New Generation Computing*, 4(1): 67–95, 1986.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pages 282–289. Morgan Kaufmann, 2001.

P. Langley. *Learning in Humans and Machines: Towards an Interdisciplinary Learning Science*, chapter Order Effects in Incremental Learning. Elsevier, 1995.

N. Lavrac and S. Dzeroski. *Inductive Logic Programming: Techniques and Applications*. Routledge, 1993.

H.-F. Li and S.-Y. Lee. Mining frequent itemsets over data streams using efficient window sliding techniques. *Expert Systems with Applications*, 36(2):1466–1477, 2009.

H.-F. Li, S.-Y. Lee, and M.-K. Shan. An efficient algorithm for mining frequent itemsets over the entire history of data streams. In *Proc. of First International Workshop on Knowledge Discovery in Data Streams*, 2004.

L. Liao, D. Fox, and H. A. Kautz. Hierarchical Conditional Random Fields for GPS-Based Activity Recognition. In *International Symposium of Robotics Research (ISRR)*, volume 28 of *Springer Tracts in Advanced Robotics (STAR)*, pages 487–506. Springer, 2005.

T. List, J. Bins, J. Vazquez, and R. B. Fisher. Performance evaluating the evaluator. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 129–136. IEEE, 2005.

D. Lowd and P. Domingos. Efficient Weight Learning for Markov Logic Networks. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, volume 4702 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 2007.

D. Luckham and R. Schulte. EPTS Event Processing Glossary v2.0. Technical report, 2011.

N. D. Mauro, F. Esposito, S. Ferilli, and T. B. . 110-121. Avoiding order effects in incremental learning. In *AIIA 2005: Advances in Artificial Intelligence,*, 2005.

L. Mihalkova and R. Mooney. Bottom-up learning of markov logic network structure. In *Proceedings of the 24th international conference on Machine learning*, pages 625–632. ACM, 2007.

R. Miller and M. Shanahan. Some Alternative Formulations of the Event Calculus. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, Lecture Notes in Computer Science, pages 452–490. Springer, 2002.

T. Minka. Discriminative models, not discriminative training. Technical report, Microsoft Research, 2005. Available at: http://research.microsoft.com/pubs/70229/tr-2005-144.pdf.

E. T. Mueller. Event Calculus. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 671–708. Elsevier, 2008.

S. Muggleton. Inverse entailment and progol. *New Generation Computing*, 13(3&4):245–286, 1995.

S. Muggleton and L. D. Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629679, 1994.

S. Muggleton, L. D. Raedt, D. Poole, I. Bratko, P. Flach, K. Inoue, and A. Srinivasan. ILP turns 20 - biography and future challenges. *Machine Learning*, 86(1):3–23, 2012.

K. P. Murphy. *Dynamic Bayesian Networks: representation, inference and learning*. PhD thesis, University of California, 2002.

J. Noessner, M. Niepert, and H. Stuckenschmidt. RockIt: Exploiting Parallelism and Symmetry for MAP Inference in Statistical Relational Models. In M. desJardins and M. L. Littman, editors, *Proceedings of the 27th AAAI Conference on Artificial Intelligence, 2013, USA*. AAAI Press, 2013. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6240.

H. Poon and P. Domingos. Sound and Efficient Inference with Probabilistic and Deterministic Dependencies. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence*, pages 458–463. AAAI Press, 2006.

L. R. Rabiner and B.-H. Juang. An introduction to Hidden Markov Models. *Acoustics, Speech, and Signal Processing Magazine (ASSP)*, 3(1):4–16, 1986.

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

S. Riedel. Improving the Accuracy and Efficiency of MAP Inference for Markov Logic. In *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 468–475. AUAI Press, 2008.

C. Sakama. Inverse entailment in nonmonotonic logic programs. In *n Proceedings of the 10th International Conference on Inductive Logic Programming*, 2000.

M. Shanahan. The Event Calculus Explained. In M. Wooldridge and M. Veloso, editors, *Artificial Intelligence Today*, volume 1600 of *Lecture Notes in Computer Science*, pages 409–430. Springer, 1999.

V. D. Shet, J. Neumann, V. Ramesh, and L. S. Davis. Bilattice-based Logical Reasoning for Human Detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE Computer Society, 2007.

P. Singla and P. Domingos. Discriminative Training of Markov Logic Networks. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 868–873. AAAI Press / The MIT Press, 2005.

P. Singla and P. Domingos. Memory-Efficient Inference in Relational Domains. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence*, pages 488–493. AAAI Press, 2006.

P. Singla and P. Domingos. Lifted First-Order Belief Propagation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1094–1099. AAAI Press, 2008.

A. Skarlatidis. *Event Recognition Under Uncertainty and Incomplete Data*. PhD thesis, Department of Digital Systems, University of Piraeus, October 2014.

C. Sutton and A. McCallum. An Introduction to Conditional Random Fields for Relational Learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*, pages 93–127. MIT Press, 2007.

D. L. Vail, M. M. Veloso, and J. D. Lafferty. Conditional Random Fields for Activity Recognition. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1331–1338. IFAAMAS, 2007.

S. Wrobel. First order theory refinement. In L. D. Raedt, editor, *Advances in Inductive Logic Programming*, pages 14 – 33, 1996.

T.-y. Wu, C.-c. Lian, and J. Y.-j. Hsu. Joint Recognition of Multiple Concurrent Activities using Factorial Conditional Random Fields. In *Proceedings of the Workshop on Plan, Activity, and Intent Recognition (PAIR)*, pages 82–88. AAAI Press, 2007.