UNIVERSITY OF PIRAEUS

DOCTORAL THESIS

Event Recognition Under Uncertainty and Incomplete Data

Author:

Anastasios SKARLATIDIS

Supervisor:

Prof. George G. VOUROS

A thesis submitted in fulfilment of the requirements

for the degree of Doctor of Philosophy

in the

University of Piraeus

Artificial Intelligence Group

Department of Digital Systems

in collaboration with the

National Centre for Scientific Research "Demokritos"

Institute of Informatics and Telecommunications

Software and Knowledge Engineering Laboratory

October 2014

“I do not fear computers. I fear the lack of them.”

— Isaac Asimov

Abstract

Symbolic event recognition systems have been successfully applied to a variety of application domains, extracting useful information in the form of events, allowing experts or other systems to monitor and respond when significant events are recognised. In a typical event recognition application, however, these systems often have to deal with a significant amount of uncertainty. In this thesis, we address the issue of uncertainty in logic-based event recognition by extending the Event Calculus with probabilistic reasoning. The temporal semantics of the Event Calculus introduce a number of challenges for the proposed model. We show how and under what assumptions we can overcome these problems. Additionally, we study how probabilistic modelling changes the behaviour of the formalism, affecting its key property, the *inertia* of fluents. Furthermore, we demonstrate the advantages of the probabilistic Event Calculus through examples and experiments in the domain of activity recognition, using a publicly available dataset for video surveillance.

Περίληψη

Τα συμβολικά συστήματα αναγνώρισης γεγονότων έχουν χρησιμοποιηθεί επιτυχώς σε μία ποικιλία εφαρμογών. Τα συστήματα αυτά εξάγουν χρήσιμη πληροφορία υπό την μορφή γεγονότων, που δίνουν τη δυνατότητα σε ειδικούς, ή σε άλλα συστήματα, να παρακολουθούν και να ανταποκρίνονται στην παρουσία γεγονότων σημαντικού ενδιαφέροντος. Ωστόσο είναι πολύ συχνό σε μία τυπική εφαρμογή αναγνώρισης γεγονότων να παρουσιάζεται σημαντική αβεβαιότητα. Σε αυτή την διατριβή, εστιάζουμε στα προβλήματα που προκύπτουν από την παρουσία της αβεβαιότητας στην αναγνώριση γεγονότων. Επεκτείνουμε ένα φορμαλισμό λογικής άλγεβρας γεγονότων με πιθανοτικό συμπερασμό. Η χρονικές σχέσεις του λογικού φορμαλισμού εισάγουν ένα πλήθος δυσκολιών στα πιθανοτικά μοντέλα και παρουσιάζουμε τον τρόπο και τις προϋποθέσεις με τις οποίες μπορούμε να ξεπεράσουμε αυτές τις δυσκολίες. Παράλληλα, μελετάμε τον τρόπο με τον οποίο η πιθανοτική μοντελοποίηση επηρεάζει την συμπεριφορά του φορμαλισμού. Επιπλέον, παρουσιάζουμε τις δυνατότητες και τα προτερήματα των πιθανοτικών μεθόδων που αναπτύξαμε με εκτενή πειραματισμό και ανάλυση στον τομέα της αναγνώρισης συμπεριφορών από βίντεο.

Acknowledgements

The completion of my PhD has been a long journey and so far is the most challenging activity of my life. Pursuing a PhD is an astonishing but often an overwhelming experience. The best and worst moments of my doctoral journey have been shared with many people. It would not have been possible to write this doctoral thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

First of all, I am deeply grateful to my supervisors, Professor George Vouros and Dr. Giorgos Paliouras for their continuous support, invaluable assistance and guidance all these years. They taught me many things including tenacity, patience and the importance of holding high scientific standards. Professor George Vouros has been supportive, patient and always had valuable suggestions to my work. I enjoyed all discussions that we had together and I appreciate his limitless encouragement, especially when I was facing difficulties to my work. Dr. Giorgos Paliouras has been a steady influence throughout my PhD work. I attribute my level of PhD degree to his excellent scientific guidance and willingness to share his extensive knowledge. He always challenged me with interesting research questions, especially when we were discussing new ideas. I appreciate that he taught me how to tackle new problems and how to develop new techniques to solve them.

During my PhD I had the privilege to collaborate with Dr. Alexander Artikis. I wish to thank him for his insightful comments and valuable advice to my work. I appreciate all motivating discussions that we had together throughout our collaboration. I am grateful for all the concepts that he taught me about logic programming and the Event Calculus.

I would like to thank my former colleague Jason Filippou for our cooperation and discussions about probabilistic logic programming. I wish to thank Professor Larry S. Davis and Dr. Vlad I. Morariu from the University Maryland for their valuable comments and suggestions to my work.

I gratefully acknowledge the funding sources that made my PhD work possible. I would like to thank the National Centre for Scientific Research “Demokritos” for the award of a postgraduate research studentship that provided the necessary financial support of this work. Additionally, this work has been partially funded by the European Commission, in the context of the PRONTO project (FP7-ICT 231738). I would like to thank the Institute of Informatics and Telecommunications of the NCSR “Demokritos” and its staff, for the computer facilities and the scientific support.

I would like to thank the EC Funded CAVIAR project/IST 2001 37540 for the publicly available activity recognition dataset, which I have used extensively in all of my experiments for this thesis.

Special thanks to my friends Ilias Zavitsanos, Matina Politi, Pantelis Nasikas, Aris Kosmopoulos, Antonis Anastasiadis, George Giannakopoulos and Vassilis Vatikiotis for their emotional support, valuable friendship and for helping me get through the difficult times. I really enjoyed the discussions that we had and the time that we spent together with Ilias Zavitsanos, George Giannakopoulos and Aris Kosmopoulos, either while we were working in the lab or enjoying a cup of coffee somewhere outside the research centre. It was a pleasure to work with them and to benefit from their knowledge and friendship. I appreciate all the insightful discussions that we had with Pantelis Nasikas about software engineering, but most importantly I am grateful for the long-term friendship that we have from our undergraduate studies.

I wish to thank my parents for all the moral support and the amazing chances they have given me over the years. They were always supportive to my life and career choices, to my successes and most important to my failures. I am so lucky to have them as my parents and I hope that this work will make them proud.

I would like to thank my brother Stelios. When we where kids, he was the one that taught me how to use a personal computer for the very first time. I appreciate all the things that we did together and everything that he taught me. I am grateful for his continuous support and encouragement to my work and my life.

Last, but by no means least, I need to thank my ever supportive partner in life. I would like to dedicate this work to my wife Maria Vasiliou. She was constantly supportive, from the very beginning of my undergraduate studies to the completion of the PhD. We have shared together all the moments of my doctoral journey, including difficulties and successes. Without her love and endless encouragement it would be impossible to complete this work. Thank you Maria.

Anastasios Skarlatidis
Athens, October 2014

Contents

List of Figures

List of Tables

To my beloved wife, Maria.

1 Introduction

"The only reason for time is so that everything doesn't happen at once."
— Albert Einstein

Today's organisations 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. In public transport management, for example, buses, trams and trains, are equipped with multiple sensors, such as accelerometers, GPS devices, etc. In video surveillance systems, multiple cameras send streams of video frames. In computer networks, network packages are continuously exchanged between multiple computers. At each point in time, data are exchanged as unrelated pieces of information that flows silently across the information systems. When such information is aggregated and correlated, it might become a source of significant knowledge and represent activities of importance for an organisation. For example, at some point in time the sensor data from public transportation vehicles may describe an ongoing traffic congestion, or that the passenger safety in a bus is being significantly reduced. In video surveillance, the stream of video frames may indicate that an accident may have happened. Finally, in computer networks an abnormal high volume of network packet exchange may indicate a possible attempt of distributed denial of service attack. These pieces of information, together with their temporal occurrence, can be represented by *events*.

An event[1] 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 video frame, a financial transaction, a GPS coordinate, as well as the result of some intelligent processing, e.g., a person walking to some direction, traffic congestion, etc. Regardless of the type of information that it curries, 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

[1] In general, the concept of 'event' is defined differently in scientific disciplines of Computing, Artificial Intelligence, Philosophy, Mathematics, Physics, Psychology, etc. A unified definition of 'event' is beyond the scope of this thesis. We investigate events only from the scope of Artificial Intelligence and Complex Event Processing.

sizes. It may happen instantaneously at some point in time (e.g., today at 17:00 o'clock) or during some interval of time (e.g., today from 17:00 to 17:05 o'clock). 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 video surveillance the events representing that some people are walking together at the same time (temporal relation), in close distance and in a similar direction (spatial relations), may indicate the situation that those particular persons are moving together (event pattern).

The automatic detection of such event patterns has received attention in a variety of application domains, such as health care monitoring, public transport management, telecommunication network monitoring, credit card fraud detection and activity recognition [Artikis et al., 2012b; Etzion and Niblett, 2010; Luckham, 2002, 2011; Turaga et al., 2008]. Systems implementing *symbolic event recognition* methods — also known as *event pattern matching* or *event pattern detection* systems — aim 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, surveillance cameras, network activity logs, etc.). The extracted information can be exploited by other systems or human experts, in order to monitor an environment and respond to the occurrence of significant events.

As shown in Figure 1.1, the input to a symbolic event recognition system is composed of at least one linearly ordered sequence of time-evolving data, i.e., a *stream*. That input stream consists of time-stamped symbols, called *simple, derived events* (SDEs). Each SDE is an event that is generated by some external process applied to a single observation. Consider, for example, a video tracking system detecting that someone is walking. That system processes the raw video frames and generates a sequence of SDEs that represent that someone is walking for some points in time. Based on such time-stamped SDEs as input, the symbolic event recognition system detects patterns of events. Every recognised instance of event pattern is represented as a *composite event* (CE). For instance, that some people are moving together. The event patterns are predefined from domain experts and capture knowledge of significant CE for the target application. A CE is like a regular event, but could only happen when other related events (SDE or CE) have happened under some specific constraints (e.g., temporal and spatial relations). Therefore, its recognition is associated with the occurrence of various SDEs and/or other CEs, involving multiple entities, e.g., people, vehicles or other objects, etc. CEs, are therefore, relational structures over other sub-events, either CEs or SDEs.

1.1 Motivation

Symbolic approaches that adopt logic-based representation, in particular, can naturally and compactly represent relational CE structures — the Event Calculus formalism [Artikis

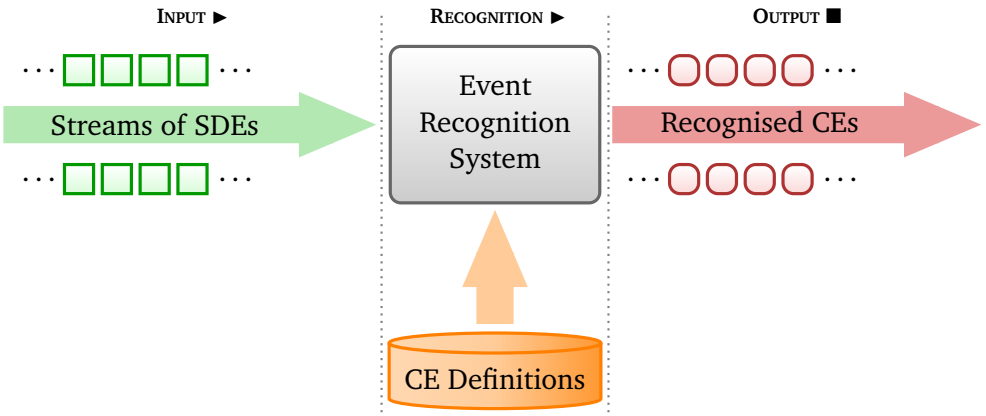


FIGURE 1.1: High-level view of an event recognition system. Domain experts provide a set of CE patterns to the event recognition system. Thereafter, the event recognition system accepts input streams of SDEs and constantly seeks to identify CEs. The output of the system is a stream or multiple streams of recognised CEs.

et al., 2010a; Kowalski and Sergot, 1986] and the Chronicle Recognition System [Dousson, 1996; Dousson and Maigat, 2007] are examples of such approaches. Logic-based systems employ formal and declarative semantics. Moreover, they provide solutions that allow one to easily incorporate and exploit prior knowledge from domain experts, in order to improve the accuracy of the event recognition system. Specifically, CEs with multiple complex temporal and spatial relations among other events can naturally be defined using a declarative language. Given that a declarative language states what is to be computed, not necessarily how it is to be computed, declarative semantics can be more easily applied to a variety of settings, not just those that satisfy some low-level operational criteria. However, pure logic-based systems cannot handle uncertainty which naturally exists in many real-world event recognition applications.

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]. Under situations of uncertainty, the performance of a Symbolic Event Recognition system may be seriously compromised. Below we outline the types of uncertainty that might appear in an event recognition application:

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

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

Imperfect CE 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.

Statistical approaches, in particular variants of probabilistic graphical models, have been successfully applied to event recognition in a variety of applications where uncertainty exists (e.g., [Brand et al., 1997; Shi et al., 2006]). Probabilistic graphical models can naturally model uncertainty through the use of probability theory [Getoor and Taskar 2007, Chapter 2; Koller and Friedman 2009]. Moreover, the structure of graphical models exploits the independence properties that exist in real-world applications, in order to compactly represent high-dimensional probability distributions. Although, graphical models provide a general-purpose probabilistic approach to model uncertain relations among entities, they lack a declarative and logic-based representation language. As a result the definition of CEs and the use of background knowledge is not straightforward. Furthermore, statistical approaches are data-driven and they are therefore dependent upon the data used for training. Background knowledge (e.g., knowledge expressed by domain experts) may describe situations that do not appear in the training data or are difficult to collected and annotate.

In conclusion, the motivation behind this thesis is twofold. One motivating factor is the requirement to compactly represent relational definitions of CEs and background knowledge, using a logic-based declarative representation language. The other source of motivation comes from the necessity to robustly handle various forms of uncertainty, using statistical methods.

1.2 Contributions

In this thesis we focus on symbolic event recognition with a logic-based representation, under situations where various forms of uncertainty hold. We explore the domain of Statistical Relational Learning under the scope of Event Recognition and propose two probabilistic and logic-based methods `MLN-EC` and `ProbLog-EC`. To demonstrate the benefits of the proposed approaches, the methods are evaluated in the real-life event recognition task of human activity recognition.

Both `MLN-EC` and `ProbLog-EC` extend discrete variants of the Event Calculus [Kowalski and Sergot, 1986] formalism with probabilistic modelling. The Event Calculus is a formalism for reasoning about events and their effects. Beyond its advantages as logic-based formalism with formal and declarative semantics, one of the most interesting

properties of the Event Calculus is that it handles the persistence of CEs with domain-independent axioms. Specifically, CEs persist over time unless they are interrupted by the occurrence of other events (SDEs or CEs).

MLN–EC employs the statistical relational framework of Markov Logic Networks (MLNs) [Domingos and Lowd, 2009] that combines the expressivity of first-order logic with the formal probabilistic properties of undirected graphical models — see de Salvo Braz et al. [2008], de Raedt and Kersting [2010] and Blockeel [2011] for surveys on logic-based probabilistic models. By combining the Event Calculus with MLNs, we present a principled and powerful probabilistic logic-based method for event recognition. The probabilistic modelling of MLN–EC uses weights in CE definitions, in order to handle the uncertainty that stems from limited dictionary of SDEs, incomplete SDE streams, annotation inconsistencies and imperfect CE definitions.

ProbLog–EC combines the Event Calculus with the probabilistic logic programming framework of ProbLog [Kimmig et al., 2011]. In contrast to MLN–EC, this dialect of the probabilistic Event Calculus handles noise in the input stream (i.e., erroneous SDEs), represented as detection probabilities of the SDE.

In particular, the contributions of this thesis are the following:

- MLN–EC is the first probabilistic version of the Event Calculus that is suitable for event recognition. The method inherits the domain-independent properties of the Event Calculus and supports the probabilistic recognition of CEs under situations of limited dictionary of SDEs, incomplete SDE streams, annotation inconsistencies and imperfect CE definitions.
- MLN–EC provides an efficient representation of the Event Calculus axioms and CE definitions in MLNs, in order to avoid the combinatorial explosion caused by the expressivity of the logical formalism.
- MLN–EC extends MLNs with a preprocessing step that translates the entire knowledge base into a compact form, in order to reduce the complexity of learning and inference.
- Under different conditions of interest, MLN–EC can model various types of CE persistence, ranging from deterministic to purely probabilistic. This is illustrated by a thorough study of the behaviour of CE persistence.
- ProbLog–EC is probabilistic logic-programming version of the Event Calculus that handles noise in the input stream.
- Experimental evaluation of the performance for both probabilistic methods in real-world benchmark dataset for activity recognition. The experimental analysis outlines the advantages and the benefits of uncertainty handling by employing probabilistic modelling.

Parts of this thesis have been published or are in the process of being published in the following articles:

Journal publications

- Skarlatidis A., Paliouras G., Artikis A. and Vouros G. Probabilistic Event Calculus for Event Recognition, Transactions of Computational Logic (TOCL), under review (accepted with minor changes).
- Skarlatidis A., Artikis A., Filippou J. and Paliouras G. A Probabilistic Logic Programming Event Calculus, Theory and Practice of Logic Programming (TPLP), To appear.
- Artikis A., Skarlatidis A., Portet F. and Paliouras G. Logic-Based Event Recognition, Knowledge Engineering Review, 27(4):469-506, 2012.
- Artikis A., Skarlatidis A. and Paliouras G. Behaviour Recognition from Video Content: A Logic Programming Approach, International Journal of Artificial Intelligence Tools, 19(2), 2010.

Conference publications

- Skarlatidis A., Paliouras G., Vouros G. and Artikis A. Probabilistic Event Calculus based on Markov Logic Networks. Proceedings of International Symposium on Rules (RuleML), Springer, 2011.
- Artikis A., Paliouras G., Portet F. and Skarlatidis A. Logic-Based Representation, Reasoning and Machine Learning for Event Recognition, International Conference on Distributed Event-Based Systems (DEBS), ACM, 2010.

1.3 Outline of the Thesis

In Chapter 2, we provide the required background for the Event Calculus and Statistical Relational Learning. We also review and discuss the literature in probabilistic and symbolic event recognition. Chapter 3 presents the first proposed method (`MLN-EC`) for event recognition that handles uncertainty with the use of probabilistic graphical models. We then, in Chapter 4, present the evaluation results of `MLN-EC` using a publicly available dataset for activity recognition. In Chapter 5, we present the second proposed approach (`ProbLog-EC`) for event recognition that handles uncertainty using probabilistic logic programming techniques. In Chapter 6, we present the evaluation results of `ProbLog-EC` on the same dataset for activity recognition. Finally, in Chapter 7, we present the main conclusions of this work along with discussion regarding the open issues and future directions.

2 Background

"There should be no such thing as boring mathematics."
— Edsger Dijkstra

In this thesis we focus on symbolic event recognition with a logic-based representation under situations where various forms of uncertainty hold. This chapter provides the required background for our work. Starting from the running example, in Section 2.1 we present the activity recognition application where we draw our examples from throughout this thesis, as well as demonstrate the features and the performance of the proposed methods. The methods that are developed in this work combine the Event Calculus formalism with Statistical Relational Learning. In particular, Section 2.2 presents briefly the Event Calculus formalism. Then in Section 2.3, we present the two state-of-the-art Statistical Relational Learning frameworks that we used in this work; that is Markov Logic Networks and ProbLog. Finally, in Section 2.4, we present related work in the domain of probabilistic and symbolic event recognition.

2.1 Running Example: Activity Recognition

To demonstrate the methods that we have developed in this work, we use the publicly available benchmark dataset of the CAVIAR project[1]. The aim is to recognise complex activities that take place between multiple persons, by exploiting information about simple observed individual activities. This dataset comprises 28 surveillance videos, where each frame is annotated by human experts from the CAVIAR team on two levels. The first level contains simple, derived events (SDEs) that concern activities of individual persons or the state of objects. The second level contains composite event (CE) annotations, describing the activities between multiple persons and/or objects — i.e., people `meeting` and `moving` together, `leaving an object` and `fighting`. Below we briefly describe the input of the activity recognition application:

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

1. The input to our methods is a stream of SDEs, representing people `walking`, `running`, staying `active`, or `inactive`. The first and the last time that a person or an object is tracked are represented by the SDEs `enter` and `exit`. Additionally, the coordinates of tracked persons or objects are also taken into consideration in order to express qualitative spatial relations, e.g. two persons being relatively close to each other.

2. The definitions of the CEs `meeting`, `moving`, `leaving an object` and `fighting` were developed in Artikis et al. [2010b]. These definitions take the form of common sense rules and describe the conditions under which a CE starts or ends. For example, when two persons are *walking* together with the same orientation, then `moving` starts being recognised. Similarly, when the same persons walk away from each other, then `moving` stops being recognised.

Based on the input stream of SDEs and the CE definitions, the aim is to recognise instances of the CEs of interest.

In this work, we do not process the raw video data in order to recognise individual activities. Instead we use the SDEs provided in the CAVIAR dataset. Despite its manual annotation, the level of ambiguity in the dataset is high. First, there is a limited dictionary of SDEs and context variables and thus the same pattern of SDEs may be ambiguously annotated by the evaluators [List et al., 2005]. Additionally, the CE definitions that we use in this work comprise common domain knowledge and do not follow the annotated data strictly. As a result, regarding the annotation of the dataset, the CE definitions are imperfect. This issue is addressed by the `MLN-EC` method in Chapter 3 and demonstrated in Chapter 4.

Furthermore, in reality the input SDEs are commonly recognised with some certainty by some external low-level classifier. This certainty is usually given in the form of probability, expressing the level of uncertainty that an SDE is detected by the low-level classifier. The `ProbLog-EC` method deals with the uncertainty in the input, as it is presented in Chapter 5 and demonstrated in Chapter 6.

Further details about the dataset, examples and the formal definitions of CEs are given throughout the following chapters.

2.2 The 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

Predicate	Meaning
$\texttt{initially}_\texttt{P}(F)$	Fluent F holds at time-point 0
$\texttt{initially}_\texttt{N}(F)$	Fluent F does not hold at time-point 0
$\texttt{holdsAt}(F, T)$	Fluent F holds at time-point T
$\texttt{happens}(E, T)$	Event E occurs at time-point T
$\texttt{initiates}(E, F, T)$	Event E initiates fluent F at time-point T
$\texttt{terminates}(E, F, T)$	Event E terminates fluent F at time-point T
$\texttt{clipped}(F, T_0, T_1)$	Fluent F is terminated at some time-point in the interval $[T_0, T_1)$
$\texttt{declipped}(F, T_0, T_1)$	Fluent F is initiated at some time-point in the interval $[T_0, T_1)$

TABLE 2.1: The Event Calculus predicates.

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.

2.2.1 Domain-independent Axiomatisation

In this work we consider finite domains of time-points, events and fluents, that are represented by the finite sets $\mathcal{T}$, $\mathcal{E}$ and $\mathcal{F}$, respectively. Similarly, 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 $\mathcal{O}$. Among the many variants of the formalism, we chose as a starting point for our work to present two dialects of the Event Calculus that are formally defined in first-order logic.

Full Event Calculus

Shanahan's Full Event Calculus is formally defined in first-order logic [Shanahan, 1999]. For simplicity and without loss of generality the predicate `releases` is excluded. This predicate is domain-dependent and defines under which conditions the law of inertia for a fluent is disabled. All fluents, therefore, are subject to inertia. Table 2.1 summarises the main elements of the Event Calculus. 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.

The domain-independent axioms of the Full Event Calculus are presented below, where $F \in \mathcal{F}$, $E \in \mathcal{E}$ and $T, T_0, T_1 \in \mathcal{T}$. Specifically, the axioms that determine when a fluent holds are defined as follows:

$$\begin{aligned} &\texttt{holdsAt}(F, T) \Leftarrow \\ &\qquad \texttt{initially}_\texttt{P}(F) \wedge \\ &\qquad \neg\texttt{clipped}(F, 0, T) \end{aligned} \tag{2.1}$$

$$\begin{aligned} &\texttt{holdsAt}(F, T) \Leftarrow \\ &\qquad \texttt{happens}(E, T_0) \wedge \\ &\qquad \texttt{initiates}(E, F, T_0) \wedge \\ &\qquad T_0 < T \wedge \\ &\qquad \neg\texttt{clipped}(F, T_0, T) \end{aligned} \tag{2.2}$$

According to axiom (2.1), a fluent holds at time T if it held initially and has not been terminated in the interval $[0, T)$. According to axiom (2.2), a fluent holds at time T if it was initiated at some earlier time T_0 and has not been terminated between T_0 and T.

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

$$\begin{aligned} &\neg\texttt{holdsAt}(F, T) \Leftarrow \\ &\qquad \texttt{initially}_\texttt{N}(F) \wedge \\ &\qquad \neg\texttt{declipped}(F, 0, T) \end{aligned} \tag{2.3}$$

$$\begin{aligned} &\neg\texttt{holdsAt}(F, T) \Leftarrow \\ &\qquad \texttt{happens}(E, T_0) \wedge \\ &\qquad \texttt{terminates}(E, F, T_0) \wedge \\ &\qquad T_0 < T \wedge \\ &\qquad \neg\texttt{declipped}(F, T_0, T) \end{aligned} \tag{2.4}$$

Axiom (2.3) defines that a fluent does not hold at time T if it did not hold initially and has not been initiated in the interval $[0, T)$. Axiom (2.4) defines that a fluent does not hold at time T if it was terminated earlier at T_0 and has not been initiated between T_0 and T.

The auxiliary domain-independent predicates `clipped` and `declipped` are defined below:

$$\begin{aligned} &\texttt{clipped}(F, T_0, T_1) \Leftrightarrow \\ &\quad \exists\, E, T\ \texttt{happens}(E, T) \wedge \\ &\qquad T_0 \leq T \wedge T < T_1 \wedge \\ &\qquad \texttt{terminates}(E, F, T) \end{aligned} \tag{2.5}$$

$$\begin{aligned}\texttt{declipped}(F,\, T_0,\, T_1) \Leftrightarrow \\ \exists\, E,\, T\, \texttt{happens}(E,\, T) \wedge \\ T_0 \leq T \wedge T < T_1 \wedge \\ \texttt{initiates}(E,\, F,\, T)\end{aligned} \tag{2.6}$$

According to axiom (2.5), a fluent is clipped in $[T_0, T_1)$ when the occurrence of an event terminates the fluent in that interval. In the same manner, axiom (2.6) defines that a fluent is declipped in $[T_0, T_1)$ when the occurrence of an event initiates the fluent in that interval. Below we present two commonly used dialects of the Event Calculus that are both formalised in first-order logic.

Discrete Event Calculus

An alternative formalisation of the Event Calculus is the Discrete Event Calculus[2] (DEC), which has been proved to be logically equivalent to the Event Calculus when the domain of time-points is limited to integers [Mueller, 2008] — i.e., $\mathcal{T} \in \mathbb{Z}$. The original DEC is composed of twelve domain-independent axioms. Similar to the Full Event Calculus, 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. We do not consider the predicates and axioms stating when a fluent is not subject to inertia (`releases` and `releasedAt`), as well as its discrete change based on some domain-specific mathematical function (`trajectory` and `antiTrajectory`).

The domain-independent axioms of the DEC are presented below, where $F \in \mathcal{F}$, $E \in \mathcal{E}$ and $T \in \mathcal{T}$. The axioms that determine when a fluent holds are defined as follows:

$$\begin{aligned}\texttt{holdsAt}(F,\, T{+}1) \Leftarrow \\ \texttt{happens}(E,\, T) \wedge \\ \texttt{initiates}(E,\, F,\, T)\end{aligned} \tag{2.7}$$

$$\begin{aligned}\texttt{holdsAt}(F,\, T{+}1) \Leftarrow \\ \texttt{holdsAt}(F,\, T) \wedge \\ \neg\exists\, E\, \texttt{happens}(E,\, T) \wedge \\ \texttt{terminates}(E,\, F,\, T)\end{aligned} \tag{2.8}$$

According to axiom (2.7), when an event E that initiates a fluent F occurs at time T, the fluent holds at the next time-point. Axiom (2.8) implements the inertia of fluents, dictating that a fluent continues to hold unless an event terminates it.

[2] An open-source implementation of the Discrete Event Calculus is available in http://decreasoner.sourceforge.net

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

$$\begin{aligned} \neg\texttt{holdsAt}(F, T{+}1) \Leftarrow \\ \texttt{happens}(E, T) \wedge \\ \texttt{terminates}(E, F, T) \end{aligned} \tag{2.9}$$

$$\begin{aligned} \neg\texttt{holdsAt}(F, T{+}1) \Leftarrow \\ \neg\texttt{holdsAt}(F, T) \wedge \\ \neg\exists\, E\ \texttt{happens}(E, T) \wedge \\ \texttt{initiates}(E, F, T) \end{aligned} \tag{2.10}$$

Axiom (2.9) states that when an event E that terminates a fluent F occurs at time T, then the fluent does not hold at the next time-point. Axiom (2.10) specifies that a fluent continues not to hold unless an event initiates it.

In contrast to the Full Event Calculus, DEC axioms are defined over successive time-points. In particular, axioms (2.7)–(2.10) are quantified over a single time-point variable T. As a result, the number of ground clauses is substantially smaller than in the Full Event Calculus. This simplification over time variables is particularly useful in situations where we require to ground the entire knowledge base, in order to perform event recognition over all time-points of an input stream of events.

2.2.2 Domain-dependent Axiomatisation

In the Event Calculus the predicates `happens`, `initiates` and `terminates` are defined only in a domain-dependent manner. In particular, the predicates `initiates` and `terminates` specify under which circumstances a fluent — representing a CE in event recognition — is to be initiated or terminated at a specific time-point. Consider, for example, that the CE `meeting` between two persons begins to be recognised when the persons are standing close to each other. Similarly, the `meeting` stops being recognised when the persons are walking away. Due to the common sense law of inertia, the `meeting` activity holds at any time-point between the initiation and termination time-points. A fluent does not hold at the time of the event that initiates it but does hold at the time of the event that terminates it. Thus the intervals over which fluents hold are left-open and right-closed. Therefore, as illustrated in Figure 2.1 at this interval the CE `meeting` is recognised.

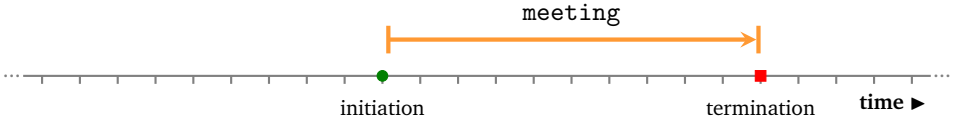


FIGURE 2.1: Event recognition with the Event Calculus

The input evidence, expressing the occurrence of SDEs at specific time-points, is represented using a collection of `happens` predicates. This collection forms a *narrative* in the Event Calculus. Specifically, a narrative comprises a collection of (ground) `happens` predicates that represent a distinguished history of SDEs occurrences. All axioms appeal, directly or indirectly, to the narrative.

Detailed examples about the structure of the domain-dependent axioms and the input narrative are given throughout the chapters of the thesis.

2.3 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 [Lafferty et al., 2001; Murphy, 2002; Rabiner and Juang, 1986], 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. Traditionally, such methods have been developed to exploit data in propositional attribute-value form. Typically, the data is represented in tabular form, where the rows of the single data table represent the example instances and the columns represent the attributes [Getoor and Taskar, 2007; Kersting, 2006, Chapter 1]. 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]; Vail et al. [2007]; Wu et al. [2007] and Liao et al. [2005].

To deal with both uncertainty and relational modelling, the research communities of Inductive Logic Programming (ILP) and statistical Machine Learning (ML) have began to incorporate aspects of complementary techniques under the domain of Statistical Relational Learning (SRL) — detailed surveys about SRL methods can be found in de Raedt and Kersting [2008, 2010]; de Salvo Braz et al. [2008]; Getoor [2001]; Getoor and Taskar [2007]; Kersting [2006] and Blockeel [2011]. In particular the ILP community developed logic-based methods that incorporate probabilistic or stochastic modelling — see for example the methods proposed by Cussens [1999]; Muggleton [2000]; Ng and Subrahmanian [1992]; Ngo and Haddawy [1997] and Sato and Kameya [2001]. On the other hand, the statistical ML community extended probabilistic methods to handle relational data modelling — see for example the methods proposed by Getoor [2003, 2006] and Sutton and McCallum [2007].

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 2.2, 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.

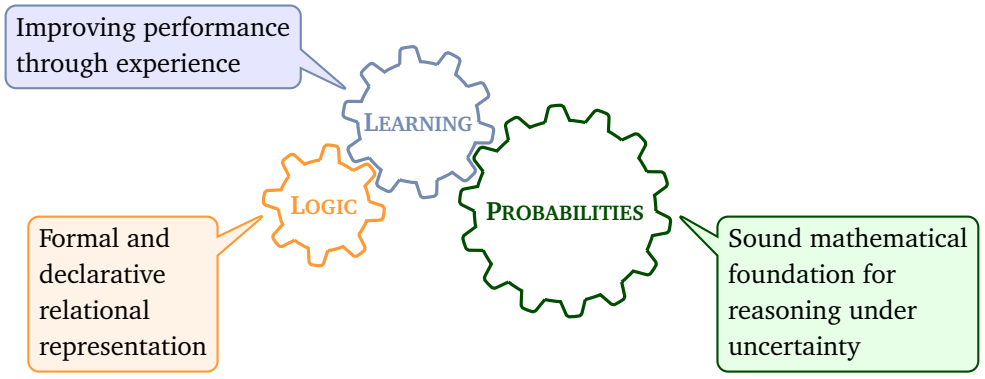


FIGURE 2.2: The research domain of Statistical Relational Learning combines logic-based representation with probabilistic modelling and machine learning.

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.

In the following sub-sections we briefly present the two state-of-the-art SRL methods that we employ in this thesis. The first method is Markov Logic Networks, which is a framework that combines first-order logic representation with Markov Network modelling. The second is ProbLog, which is a probabilistic extension to logic programming.

2.3.1 Markov Logic Networks

Markov Logic Networks[3] (MLNs) [Domingos and Lowd, 2009] 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 — e.g., recognise the `moving` activity, given a narrative of input events that represent people, `walking`, `running`, etc. 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 $M_{L,\mathcal{C}}$

[3] Systems implementing MLN reasoning and learning algorithms can be found at the following addresses:
`http://alchemy.cs.washington.edu`
`http://research.cs.wisc.edu/hazy/tuffy`
`http://code.google.com/p/thebeast`
`http://ias.cs.tum.edu/probcog-wiki`

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:

$$P(Y{=}\mathbf{y} \mid X{=}\mathbf{x}) = \frac{1}{Z(\mathbf{x})} exp\left(\sum_{i=1}^{|F_c|} w_i n_i(\mathbf{x}, \mathbf{y})\right) \tag{2.11}$$

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 (2.11) 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 [Minka, 2005; Singla and Domingos, 2005; Sutton and McCallum, 2007].

Inference

Still, directly computing equation (2.11) 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 [Biba et al., 2011; Poon and Domingos, 2006; Singla and Domingos, 2006], variants of Belief Propagation [Gonzalez et al., 2009; Kersting, 2012; Kersting et al., 2009; Singla and Domingos, 2008], Integer Linear Programming [Huynh and Mooney, 2009; Noessner et al., 2013; Riedel, 2008], lifted model counting [Apsel and Brafman, 2012; den Broeck et al., 2011; Gogate and Domingos, 2011], etc.

Two types of inference can be performed in MLNs, i.e., marginal inference and maximum a-posteriori inference (MAP). In event recognition, the former type of inference computes the conditional probability that CEs hold given a narrative of observed SDEs, i.e., $P(\texttt{holdsAt}(CE, T){=}True \mid narrative\ of\ SDEs)$. 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.

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

i.e., $\underset{\texttt{holdsAt}}{\text{argmax}}\, P(\texttt{holdsAt}(CE, T) \mid narrative\ of\ SDEs)$. In MLNs 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., Huynh and Mooney [2009]; Noessner et al. [2013]; Riedel [2008]). In this thesis the we employ the LP-relaxed Integer Linear Programming method proposed by Huynh and Mooney [2009]. In particular, 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 $\texttt{holdsAt}(CE, T)$ 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].

Learning

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 (2.11). 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:

$$-\log P_w(Y{=}\mathbf{y} \mid X{=}\mathbf{x}) = \log Z(\mathbf{x}) - \sum_{i=1}^{|F_c|} w_i n_i(\mathbf{x}, \mathbf{y}) \tag{2.12}$$

The vector $\mathbf{x}$ is the assignment of truth values to the evidence random variables X, according to the training data. Consider, for example, a set of ground predicates that represent a stream of SDEs — people walking, running, standing still at specific points in time. All given instantiations of predicates represent the occurrence of SDE and thus their corresponding assignment in $\mathbf{x}$ is *True*. Conversely, the absence of any SDE is represented the by the assignment of *False* in $\mathbf{x}$. 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 [Lowd and Domingos, 2007; Singla and Domingos, 2005]. 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.

One approach to this problem is the diagonal Newton method, in order optimise the CLL function. In particular, the method iteratively estimates the parameters of the model using the following equation:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \boldsymbol{\alpha}\mathbf{H}^{-1}\mathbf{g} \tag{2.13}$$

The vector $\mathbf{w}$ holds the weights w_i of all soft-constrained clauses in F_c of the MLN. The initial value of the weights can be randomly assigned, or set to zero or set to any other value (e.g., prior weights). At each step t the algorithm estimates the current value of the parameters $\mathbf{w}_t$, by calculating the difference between the previous weights $\mathbf{w}_{t-1}$ and the product of the gradient $\mathbf{g}$ with the inverse Hessian $\mathbf{H}^{-1}$ matrix and the step size $\boldsymbol{\alpha}$. The gradient $\mathbf{g}$ is a vector, which is composed of the partial derivatives of the negative CLL with respect to the corresponding weight. Each partial derivative represents the difference between the expected number of satisfied groundings of the corresponding clause and the actual number in the training data. The Hessian is a square matrix, comprising the second-order partial derivatives of the negative CLL, representing the correlation between two clauses. However, inverting the Hessian may become infeasible. For that reason, diagonal Newton assumes that all off-diagonal entries of the Hessian are zero. The product between the inverse Hessian and the gradient defines the direction of the weight update. The step size $\boldsymbol{\alpha}$, is a vector that defines the magnitude of the corresponding weight update and is computed using the full Hessian matrix. The exptected counts that are used to compute the gradient and the Hessian of the negative CLL are efficiently approximated using the *MC-SAT* algorithm.

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, 2011]. Instead of optimising the CLL, max-margin maximise the following ratio:

$$\frac{P(Y{=}\mathbf{y}|X{=}\mathbf{x},\mathbf{w})}{P(Y{=}\hat{\mathbf{y}}|X{=}\mathbf{x},\mathbf{w})} \tag{2.14}$$

The above equation measures the ratio between the probability of correct truth assignment $\mathbf{y}$ of CEs and the closest competing incorrect truth assignment $\hat{\mathbf{y}}=\underset{\bar{\mathbf{y}}\in\mathbf{Y}\setminus\mathbf{y}}{\operatorname{argmax}} P(Y=\bar{\mathbf{y}}|X=\mathbf{x})$. Huynh and Mooney [2009] formulates the max-margin problem as 1-slack structural SVMs [Joachims et al., 2009] and estimates the weights by employing linear programming techniques.

In this work we are interested in recognising CEs given a stream of SDEs. Depending on the needs of the event recognition application, we may either want to recognise CEs with probabilities (indicating a degree of belief) or aim to maximise the event recognition accuracy. For the former type of recognition, we estimate the weights by maximising the CLL using the second-order diagonal Newton method of Singla and Domingos [2005]. For the accuracy maximisation task, we employ the max-margin training method proposed by Huynh and Mooney [2009].

2.3.2 Probabilistic Logic Programming with ProbLog

ProbLog[4] [de Raedt et al., 2007; Kimmig et al., 2011] is a probabilistic extension of the well-known logic programming language Prolog. In contrast to Prolog, the facts in ProbLog are associated with a probability value. Each fact has the form $p_i :: \mathtt{f}_i$ where p_i is a probability value and $\mathtt{f}_i$ is an atom. When the $\mathtt{f}_i$ is not ground (i.e., contains at least one variable in its arguments), then the probability p_i is applied to all possible groundings of $\mathtt{f}_i$. All ground facts correspond to random variables, where each ground fact $\mathtt{f}_i$ is true with probability p_i. When multiple instances of the same fact are given, then they correspond to different random variables. Classic Prolog facts that are not associated with a probability are silently given probability 1, indicating that they are true with absolute certainty.

ProbLog makes an independence assumption on its random variables. This means that a rule which is defined as a conjunction of some probabilistic facts has a probability equal to the product of the probabilities of these facts. When a predicate appears in the head of more than one rule then its probability is computed by calculating the probability of the implicit disjunction created by the multiple rules. For example, for a predicate $\mathtt{p}$ with two rules $\mathtt{p} \leftarrow \mathtt{l}_1$ and $\mathtt{p} \leftarrow \mathtt{l}_2, \mathtt{l}_3$, the probability of predicate $\mathtt{p}$ being true is $P(\mathtt{p})$ and is computed as follows:

$$\begin{aligned} P(\mathtt{p}) &= P\big((\mathtt{p} \leftarrow \mathtt{l}_1) \vee (\mathtt{p} \leftarrow \mathtt{l}_2, \mathtt{l}_3)\big) \\ &= P(\mathtt{p} \leftarrow \mathtt{l}_1) + P(\mathtt{p} \leftarrow \mathtt{l}_2, \mathtt{l}_3) - P\big((\mathtt{p} \leftarrow \mathtt{l}_1) \wedge (\mathtt{p} \leftarrow \mathtt{l}_2, \mathtt{l}_3)\big) \\ &= P(\mathtt{l}_1) + P(\mathtt{l}_2) \times P(\mathtt{l}_3) - P(\mathtt{l}_1) \times P(\mathtt{l}_2) \times P(\mathtt{l}_3) \end{aligned}$$

[4] ProbLog has been fully integrated in the YAP Prolog system (http://www.dcc.fc.up.pt/~vsc/Yap). Further details, examples and code samples are available on the ProbLog website (http://dtai.cs.kuleuven.be/problog/problog1/problog1.html)

More formally, a ProbLog *program* Γ is composed of a set F of n probabilistic facts $p_i :: \mathtt{f}_i$ and a set of *definite clauses* that express arbitrary *background knowledge* (BK) — i.e., $\Gamma = \{p_1 :: \mathtt{f}_1, \ldots, p_n :: \mathtt{f}_n\} \cup BK$. The groundings of each probabilistic fact are defined by a finite number of substitutions, i.e., $\{\theta_{i1}, \ldots, \theta_{ij_i}\}$. Therefore, the maximal set of ground logical facts that can be added to the BK is $L_\Gamma = \{\mathtt{f}_1\theta_{11}, \ldots, \mathtt{f}_1\theta_{1j_1}, \ldots, \mathtt{f}_n\theta_{n1}, \ldots, \mathtt{f}_n\theta_{nj_n}\}$. As the random variables corresponding to facts in L_Γ are mutually independent, the ProbLog program defines a probability distribution over ground logic programs (or sub-programs) $L_g \subseteq L_\Gamma$:

$$P(L_g \,|\, \Gamma) = \prod_{\mathtt{f}_i\theta_j \in L_g} p_i \prod_{\mathtt{f}_i\theta_j \in L_\Gamma \setminus L_g} (1 - p_i) \tag{2.15}$$

The rules in BK are fixed and thus each sub-program L_g corresponds to a unique Herbrand interpretation of $L_g \cup BK$. In other words, according to equation (2.15) a ProbLog program also defines a distribution over these Herbrand interpretations. Although these semantics can be generalised to the countably infinite case (see Sato [1995] for further details), in this work we assume that all probabilistic facts are ground. Furthermore, in the context of event recognition, the probabilistic facts are ground SDEs and the BK comprises arbitrary rules, e.g., CE definitions.

With the help of Equation (2.15), one could compute the probability that a query q holds in a ProbLog program — named *success* probability — by summing the probabilities of all sub-programs that entail it. Specifically, the success probability $P_s(q \,|\, \Gamma)$ of a query q in a ProbLog program Γ is given by the marginal of $P_s(L_g \,|\, \Gamma)$ with respect to q:

$$P_s(q \,|\, \Gamma) = \sum_{L_g \subseteq L_\Gamma} \mathbf{1}_q(L_g) P(L_g \,|\, \Gamma) \tag{2.16}$$

where $\mathbf{1}_q(L_g) = 1$ when there exists a substitution θ such that $L_g \cup BK \models q\theta$, and $\mathbf{1}_q(L_g) = 0$ otherwise.

Inference

However, computing the success probability through equation (2.16) is computationally infeasible for large programs, as the number of sub-programs to be checked is exponential in the number of probabilistic facts. By combining equations (2.15) and (2.16) and eliminating redundant terms, we end up with the following characterisation:

$$P_s(q \,|\, \Gamma) = P\Big(\bigvee_{e \in Proofs(q)} \bigwedge_{f_i \in e} f_i \Big) \tag{2.17}$$

That is, the task of computing the success probability of a query q is transformed into the task of computing the probability of the Disjunctive Normal Form (DNF) formula

of equation (2.17). Practically, equation (2.17) expresses that the success probability of query q is equal to the probability that at least one of its proofs is sampled. This, unfortunately, is not a question of straightforwardly transforming the probability of the DNF to a sum of products. If we were to translate equation (2.17) to a sum of products, we would assume that all different proofs (conjunctions) are disjoint, meaning that they represent mutually exclusive possible worlds, which does not hold in the general case. In order to make the proofs disjoint, one would have to enhance every conjunction with negative literals, thus excluding worlds whose probability has already been computed in previous conjunctions of the DNF. This problem is known as the *disjoint-sum problem* and is #P-hard [Valiant, 1979].

ProbLog's approach to this problem consists of using Binary Decision Diagrams (BDDs) [Bryant, 1986] to compactly represent the DNF of equation (2.17). A BDD is a binary decision tree with redundant nodes removed and isomorphic sub-trees merged. The BDD nodes represent the probabilistic facts of the ProbLog program. Every node has a *positive* and *negative* outward edge, leading to either a child node or the special *true* or *false* terminal nodes. The positive outward edge of the BDD node is labelled with the probability of the respective probabilistic fact and the negative edge is labelled with the complement of that probability. Positive and negative edges represent distinct decisions on inclusion of the relevant fact in the currently sampled possible world; a positive edge signifies that the fact represented by its parent node is included in the sample with the labelled probability, whereas a negative edge signifies that the fact is not included in the sample with the complement of the same probability. Therefore, by following a path from the root node to the true terminal node, one could sample a conjunction of the DNF formula of equation (2.17). The negative outward edges offer a compact representation of the negated literals required to enhance the DNF formula in order to make it represent a disjunction over disjoint conjunctions.

To summarise, inference follows three steps. The first step is to gather all proofs of the query q by scanning the Selective Linear Definite (SLD) tree of proofs and represent them as the DNF formula of equation (2.17). Afterwards, the DNF is translated to a BDD. Finally, the probability of this BDD is computed recursively, starting from the root node and assuming a probability of 1 for the true terminal and 0 for the false terminal. In this work, all ProbLog experiments have been performed using this exact inference algorithm.

In some applications that include large knowledge bases, building a BDD for all proofs may become intractable. Methods that approximate the success probability by selecting a subset of proofs, have been proposed in the literature. de Raedt et al. [2007] proposed an iterative deepening algorithm that approximates the success probability from a subset of proofs, given a desired approximation factor ϵ. However, the algorithm may still need to select large sets of proofs. A more efficient alternative is the k-best inference algorithm [Kimmig et al., 2011], that selects only the k most probable proofs. However, the search

strategy of k-best may not select the optimal set of proofs. The set may contain highly redundant proofs and does not guarantee better approximation than any other set. The k-Optimal algorithm [Renkens et al., 2012] overcomes these issues and ensures that the set of k proofs is of provably good quality and that the set of proofs is diverse, i.e., less likely to select proofs that share similar facts, leading to better approximation of the success probability with fewer proofs.

Instead of searching for a near optimal subset of proofs, an alternative way to approximate the success probability is the use of Monte Carlo sampling techniques. The program sampling algorithm of Kimmig et al. [2008, 2011] generates samples by exploring the SLD tree. The DNF sampling algorithm of Shterionov et al. [2010] performs sampling over the possible worlds that contain a proof of the query of interest. Finally, Fierens et al. [2011] translates probabilistic logic programs and probabilistic facts into an equivalent weighted CNF form of Markov Logic Networks. As a result, the state-of-the-art high-performance algorithms MaxWalkSAT and MC-SAT algorithms can be used, in order to perform MAP or marginal inference, respectively.

2.4 Related Work

In this section we present methods that can represent and reason about events with complex relational structure, as well as handle various forms of uncertainty. In particular, we outline action formalisms and present symbolic and probabilistic event recognition methods. Events with complex relational structure require formal, declarative knowledge representation and temporal reasoning. Action formalisms provide well-defined logical representation of events and change that can be exploited by temporal reasoning algorithms for recognition, planning, decision making and forecasting. Symbolic event recognition systems are knowledge-driven methods that use a declarative representation for event definitions. Finally, probabilistic symbolic event recognition methods combine declarative representation capabilities with probabilistic modelling, in order to recognise CEs under uncertainty.

2.4.1 Action formalisms

Event recognition can be modelled as a dynamic system — i.e., a system whose state may change over time due to the occurrence of events [Sandewall, 1994, Chapter 1]. Representing and reasoning about events and change is a fundamental research field in the domain of Artificial Intelligence. Numerous formalisms have been proposed in the literature, such as the Situation Calculus [McCarthy, 1983; McCarthy and Hayes, 1968; Reiter, 2001], the Event Calculus [Kowalski and Sergot, 1986; Miller and Shanahan, 2002; Mueller, 2008; Patkos and Plexousakis, 2008; Shanahan, 1999] the Fluent Calculus [Thielscher, 1999, 2001], the action language $\mathcal{C}+$ [Akman et al., 2004; Giunchiglia

et al., 2004] and Temporal Action Logics [Doherty et al., 1998; Kvarnström, 2005]. Action formalisms focus on the development of axioms that model events (or actions) and their affects, in order to deal with the dynamic aspects of the world and reason about change. Comparisons and proofs of equivalence between action formalisms can be found in Kowalski and Sadri [1997], Van Belleghem et al. [1997], Chittaro and Montanari [2000], Miller and Shanahan [2002], Schiffel and Thielscher [2006], Mueller [2006, Chapter 15], Craven [2006] and Paschke and Kozlenkov [2009]. All action formalisms rely on the following two fundamental elements:

1. *Events* (or *actions*) [Kowalski and Sergot, 1986; McCarthy, 2002] are known to have occurred (e.g., observed/detected by some other external process or system) or are planned to happen in the future. Events compose the conditions under which they have some effect on the actual states of the modelled world.

2. *Fluents* (or *features*) [McCarthy, 1983; Sandewall, 1994] represent properties of the domain which may change in response to the execution of events. An important characteristic of action formalisms, is that fluents are *inert*. In other words, the value of a fluent remains the same (i.e., persists) until there is a specific reason for it to change — e.g., the occurrence of some events.

Furthermore, logic representation action formalisms can compactly represent composite events with complex relations and directly incorporate domain-specific background knowledge.

The representation of time is an important property of event recognition and different action formalisms may use different time models. Situation Calculus and Fluent Calculus employ a discrete and forward-branching time model, where each point in time may have several successors, but only one predecessor. In fact, the occurrence of an event may give rise to a different possible future. A point in time is represented by a *situation*, which is a result of a possible sequence of events. Therefore, events are assumed to occur sequentially and atemporally. On the other hand, in the Event Calculus, $\mathcal{C}+$ and Temporal Action Logics, there is a single time-line on which events occur. The time-line may be continuous (i.e., the time domain as a set of real numbers) [Kowalski and Sergot, 1986; Shanahan, 1999] or discrete (i.e., the time domain is restricted to integer numbers) [Mueller, 2008]. The Event Calculus has also been extended with branching time in order to support reasoning for hypothetical events — see the proposed Event Calculus dialects of Provetti [1996], Kowalski and Sadri [1997], Lévy and Quantz [1998] and Mueller [2007]. Similarly, the Situation Calculus has been extended with explicit time representation and actual events — see for example the proposed variants of Pinto and Reiter [1993, 1995] and Reiter [1996, 1998]. The branching time model is particularly suited to planning or forecasting problems, where hypothetical events may guide to different possible future solutions. On the other hand, in the single time-line model

the events are not hypothetical and the time is expressed explicitly, allowing to directly express temporal constraints in the CE definitions. Therefore, the single time-line model is more suitable for event recognition, since the task is to recognise CEs of interest in time-stamped streams of observed SDEs (i.e., actual events that have happened).

As presented in Section 1.1, noise is an unavoidable aspect of real-world applications. For example, the input observations may be affected by a significant amount of noise, resulting in erroneous recognition of CE. Action formalisms have been extended to support various forms of uncertainty in planning and decision making.

Most of the probabilistic extensions concern the Situation Calculus Bacchus et al. [1995, 1999] proposed a probabilistic variant of the Situation Calculus, in order to deal with noisy sensors. Based on Reiter's formulation of the Situation Calculus [Reiter, 1991], the probabilistic formalism supports non-determinism in actions and associates a weight to each situation, capturing a degree of belief. The formalism allows to reason probabilistically about an agent's degrees of belief and how these beliefs change when actions are executed. Similarly, a probabilistic variant of the Situation Calculus is proposed by Reiter [2001, Chapter 12], in order to reason about actions with an uncertain outcome. Mateus et al. [2001] extends the Situation Calculus with actions that have uncertain effects, using discrete, continuous, and mixed probability distributions. Boutilier et al. [2001] proposed a probabilistic variant of the Situation Calculus that formulates first-order Markov decision processes. Hajishirzi and Amir [2008, 2010] model Reiter's probabilistic Situation Calculus by a graphical model in the form of a Bayesian Network for answering queries given a sequence of actions.

There are also proposals for extending action languages other than Situation Calculus with probabilities. Hölldobler et al. [2006] proposed a probabilistic extension of the Fluent Calculus that models first-order Markov decision processes in planning domains with actions having probabilistic effects. $\mathcal{PC}+$ is a probabilistic generalisation of $\mathcal{C}+$ that incorporates probabilistic knowledge about the effects of events [Eiter and Lukasiewicz, 2003]. $\mathcal{PC}+$ supports non-deterministic and probabilistic effects of events, as well as probabilistic uncertainty about the initial state of the application. Similar to most other probabilistic variants of common sense reasoning languages, the method focuses on planning under uncertainty while inertia remains deterministic. Such probabilistic planning and reasoning approaches assume that the parameters of the model are known (i.e., probabilities or weights).

2.4.2 Event Recognition

Event recognition systems operate by observing streams of events that occur in the environment. They interpret and combine these events, in order to identify composite events (CEs). In this section we present, event recognition systems that can recognise

CEs from heterogeneous streams of information, such as sensor data, log files or event streams produced by other external special purpose algorithms (e.g., motion tracking, object identification, etc). There are numerous event recognition approaches, that have been developed for different application domains. Following a declarative approach, the structure of CEs is defined in terms of logical, temporal or spatial constraints that indicate the conditions under which CEs occur. Each definition represents a pattern that isolates relevant events and combines their mutual relations (e.g., temporal), in order to recognise CEs of interest. Such a declarative representation can directly incorporate and take advantage of domain specific knowledge.

The chronicle recognition system (CRS) [Dousson, 1996; Dousson et al., 1993; Dousson and Maigat, 2007] is a symbolic event recognition method that has been applied to a variety of problems, such as cardiac arrhythmia recognition [Callens et al., 2008], computer network monitoring [Dousson, 1996; Dousson and Maigat, 2007], airport ground traffic monitoring [Choppy et al., 2009], etc. CE definitions are represented using a declarative temporal language and translated into temporal constraint networks (TCN), in order to perform efficient event recognition. In particular, a CE definition is a conjunctive formula with predicates that represent events (SDEs or other CEs), as well as persistency or event absence. The time model is linear and the temporal constraints are expressed over time-points. A TCN is a directed symbolic network, where vertices correspond to instantiations of some event (CE or SDE) and edges represent temporal constraints between the involved events. Vu et al. [2003] proposed a scenario recognition method for video interpretation that translates CE definitions into TCN. For reasons of efficiency the networks are automatically decomposed into several sub-networks and the recognition is performed hierarchically.

A Petri-Net is a mathematical modelling formalism that can express a variety of dynamic behaviours, such as sequencing, concurrency and synchronisation [Petri, 1966]. It is represented by a bipartite graph and there are numerous forms of Petri-Nets — see Murata [1989] and David and Alla [1994] for comprehensive surveys on Petri-Net variants. Choppy et al. [2009] translated the CRS language into Petri-Nets, which are executed to recognise CEs. Petri-Net modelling has been applied, among others, to computer vision — see, for example, the methods of Castel et al. [1996], Lavee et al. [2007] and Ghanem et al. [2004]. Each Petri-Net graph represents a CE, which is composed of SDEs (place nodes) and temporal or logical constraints (transition nodes). Place nodes may hold tokens, indicating the current state of the model. Transition nodes define the conditions under which tokens can move between places, indicating the fulfillment of a constraint (e.g., temporal, logical, spatial) between the SDEs. When a token reaches the terminal node the CE is recognised, since all related events have happened under the required constraints.

Rota and Thonnat [2000] proposed a logic-based declarative language for expressing CE definitions in the domain of video surveillance. The definition of a CE consists of

the relations between objects of the scene and the SDEs, using spatial and temporal constraints. The knowledge base of definitions is translated into a constraint satisfaction problem, in order to perform event recognition. Shet et al. [2005] proposed a logic programming approach to represent and recognise CEs of human activities. Input SDEs are detected by low-level computer vision algorithms. CE definitions are represented by logical rules and event recognition is performed using a Prolog-based reasoning engine. The method can continuously monitor and recognise predefined CEs, as well as answer queries about events that have been archived.

Chen et al. [2008] employ Event Calculus representation and reasoning, in order to recognise activities of daily living for the personalised assistance of elderly and disabled persons in smart homes. The method models complex aspects of events, such as sequence, non-deterministic choice of actions, concurrency, iteration and conditionals, using extra-logical operators. An other method for recognising human behaviour activities based on the Event Calculus is proposed by Artikis and Paliouras [2009]. The method extends the logic-programming Event Calculus of Kowalski and Sergot [1986] with temporal intervals. The input is formed by a narrative of SDEs that express short-term behaviours (e.g., people walking or standing still). Definitions expressing the spatio-temporal conditions under which long-term human behaviours occur are represented using Event Calculus language. The method recognises CE over maximal intervals. Artikis et al. [2012a] propose an Event Calculus dialect for efficient run-time event recognition that can scale to large data streams of SDEs. The method supports expressive CE definitions with interval-based temporal constraints. The temporal reasoning of the method processes the SDE streams on-line over sliding windows. Furthermore, the method can be used in applications where data might arrive with a delay from, or might be revised by, the underlying event sources.

Additionally, there are some approaches to event recognition, which use ontological modelling. Saguna et al. [2011] performs CE recognition by combining ontological modelling with spatio-temporal reasoning. Temporal and spatial knowledge is expressed as properties of ontological concepts. At the input level, SDEs are provided by external low-level classifiers (e.g., decision trees) or derived by sensors. Using ontological and rule-based reasoning CEs are recognised. Okeyo et al. [2014, 2012] propose a hybrid approach that combines ontologies with temporal reasoning for modelling activities of daily living. Based on the Web Ontology Language (OWL) [Horrocks, 2005] the method represents and reason about domain knowledge in terms of concepts, properties and relations. The method augments OWL with the concept of 4D-Fluents [Batsakis and Petrakis, 2011; Welty and Fikes, 2006], in order introduce a temporal dimension to the knowledge base and represent time-points, time-intervals and fluents. CE definitions model human activities, in terms of other events (SDEs or CEs) that are temporally related using Allen's interval relations [Allen, 1983]. Furthermore, the method employs entailment rules with which it can reason about dependencies among ongoing SDEs and

CEs — i.e, axioms that derive and assert intervals, assert instances of fluents, as well as derive and assert CEs. Event recognition is performed by processing streams of sensor data against a given set of CE definitions.

2.4.3 Event Recognition Under Uncertainty

The event recognition methods that have been outlined in the previous section, can compactly represent and reason about composite events (CEs). In many real-world applications, however, uncertainty naturally exists and may seriously compromise event recognition accuracy. The aforementioned methods cannot consider, model or handle any form of uncertainty. All input SDEs are considered as a stream of deterministic facts. Similarly, CE definitions together with the event recognition engine form a deterministic model.

Symbolic methods that can handle uncertainty have been developed especially for applications that involve considerable amounts of uncertain information, e.g., activity recognition from video data. Table 2.2 summarises the main features of symbolic methods for event recognition under uncertainty. In columns two to four, we consider three types of uncertainty. First, the uncertainty that arises from incomplete input streams. For example, due to some sensor failure, some SDEs may be missing from the input stream. Second, low-level classifiers (e.g., tracking, object recognition, etc) usually detect SDEs with some confidence. This confidence value can be propagated to the event recognition system. Third, CE definitions may not capture perfectly the conditions under which CEs occur. For example, due to the limited dictionary of SDEs or inconsistent training data, the definitions of CEs may not be perfect. In that case, it is desirable to model the uncertainty of CE definitions, e.g., associate definition with confidence value. In the fifth column we present the temporal model of each method. Temporal modelling is an important characteristic in event recognition. Most methods impose either sequential ordering or temporal constraints over SDEs and CEs. Finally in the last column we briefly provide additional characteristics of interest for each method, such as the probabilistic model (e.g., Bayesian Network) the knowledge representation language (e.g., logic-based) or the support for inertia.

Work	Uncertainty			Temporal Model	Method Details
	Incomplete input	SDE conf.	CE conf.		
Albanese et al. [2007]			✓	Sequential	Stochastic Automata.
Albanese et al. [2011]			✓	Sequential; Intervals	Stochastic Automata with temporal duration constraints between subsequent SDEs. Recognition in the presence of intervening sub-events.
Molinaro et al. [2014]			✓	Sequential	Probabilistic Penalty Graphs. Recognition in the presence of intervening sub-events.
Albanese et al. [2008]			✓	Sequential	Probabilistic Petri-Nets.
Lavee et al. [2013]		✓		Sequential	Stochastic Petri-Nets.
Ryoo and Aggarwal [2009]	✓	✓		Intervals	Hierarchical CE representation with probabilistic inference.
Albanese et al. [2010]		✓	✓	Time-points	Probabilistic and Logic-based.
Shet et al. [2007, 2011]		✓	✓	Sequential	Billatice and logic-based.
Filippaki et al. [2011]	✓	✓		Intervals	Weighted logic-based definitions.
Romdhane et al. [2013]	✓	✓		Time-points; Intervals	Probabilistic extension of Vu et al. [2003] event description language.
Kersting et al. [2006]			✓	Sequential	Logic-based Hidden Markov Models.
Natarajan et al. [2008]			✓	Sequential	Logic-based Hierarchical Hidden Markov Models.
Manfredotti [2009]; Manfredotti et al. [2010]		✓	✓	Sequential	Logic-based Dynamic Bayesian Networks.
Biswas et al. [2007]			✓	Sequential; Time-points	Dynamic Markov Logic Networks. Probabilistic inertia.
Helaoui et al. [2011]			✓	Time-points	Markov Logic Networks. Probabilistic inertia.

This table continues to the next page ▶

This table continues from the previous page.

Work	Uncertainty			Temporal Model	Method Details
	Incomplete input	SDE conf.	CE conf.		
Tran and Davis [2008]	✓	✓	✓	Intervals	Markov Logic Networks. Temporal intervals are provided by input SDEs.
Morariu and Davis [2011]			✓	Intervals	Markov Logic Networks. Temporal intervals are provided by input SDEs.
Song et al. [2013a,b]			✓	Intervals	Markov Logic Networks. Domain-independent axioms. Temporal intervals are provided by input SDEs.
Brendel et al. [2011]; Selman et al. [2011]			✓	Intervals	Probabilistic Event Logic. Intervals are inferred.
Sadilek and Kautz [2012]			✓	Time-points	Hybrid Markov Logic Networks. Real-valued spatial constraints that affect the recognition confidence of CE.
Helaoui et al. [2013]			✓	Sequential	Log-linear Description Logics.
Wasserkrug et al. [2005, 2008, 2012]		✓	✓	Time-points; Intervals	Logic-based Bayesian Networks.
Cugola et al. [2014]		✓	✓	Time-points; Intervals	Probabilistic extension of Cugola and Margara [2010] event description language. Bayesian Networks.

TABLE 2.2: Symbolic Event Recognition methods that can handle uncertainty.

Albanese et al. [2007] developed a recognition method that models activities using a stochastic automaton. However, their initial method cannot represent temporal constraints. The method is extended in [Albanese et al., 2011], in order to identify situations that cannot be explained sufficiently by any of the known CEs. In particular, the stochastic automaton is extended with temporal constraints, specifying that each subsequent SDE can occur within a user-defined temporal interval. Using possible-worlds modelling,

the method finds totally (or partially) unexplained activities. One limitation of stochastic automata, is that the recognition of a CE is suspended in situations where other events are intervening between the required events (SDEs or other CEs). The probabilistic penalty graphs (PPGs) of Molinaro et al. [2014] overcome this limitation by extending the stochastic automaton of Albanese et al. [2007] with belief degradation. The edges that connect subsequent events in a PPG, forming the structure of a CE, are associated with a probability (noise) value that degrades the belief of the CE when other events intervene. As a result, the CE is being recognised, with reduced probability. The method can also discover event patterns that do not belong to the set of known CEs. Additionally, for scalability reasoning PPGs can be combined by merging common sub-PPGs, indexed and executed in parallel.

A probabilistic extension to Petri-Nets has been proposed by Albanese et al. [2008], in order to perform probabilistic recognition of CEs that represent human activities. Each Petri-Net expresses a CE and is formed by SDEs that are connected through constraints (i.e., spatial constraints, temporal durations and forbidden actions). The transition between a pair of subsequent SDEs is associated with a probability value. Given a sequence of SDEs, the method can identify segments of the sequence in which a CE occurs with probability above a specified threshold or infer the most likely CE in the sequence. Lavee et al. [2013] proposed a stochastic variant of Petri-Nets for modelling the uncertainty of input SDEs. Specifically, SDEs are recognised with some certainty, by lower level classification algorithms. CEs are represented by Petri-Nets, in terms of SDEs that are associated with certainties and temporal constraints.

Ryoo and Aggarwal [2009] proposed a hierarchical description-based method that combines a formal syntax for representing CE definitions, introduced in Ryoo and Aggarwal [2006], with probabilistic recognition. The method aims to probabilistically detect the time intervals in which CEs occur. Input SDEs are associated with probabilities, indicating a degree of belief. Based on a context-free grammar representation scheme, CE definitions are expressed in terms of other events (SDEs or CEs) and form a hierarchical model. Furthermore, events are related through logical (i.e., conjunction, disjunction and negation), spatial and temporal interval constraints [Allen, 1983]. In situations where the input stream is incomplete, the method generates the missing SDEs with lower confidence. The probability of a CE is calculated by exploiting the dependency information between the CE and its sub-events. When the calculated probability of the CE is above a specified threshold, it is considered to be recognised.

Albanese et al. [2010] proposed a probabilistic activity description language for expressing CE on top of an image processing suit. Their method merges a logic-based representation with probabilistic modelling. The input SDEs can be either Boolean or probabilistic. CEs are defined by users in a first-order logic where the dependencies between SDEs are modelled by triangular norms [Fagin, 1996]. The method provides efficient algorithms for off-line and on-line recognition. Another logic-based method for handling uncertainty

in activity recognition is proposed by Shet et al. [2007, 2011]. The method employs logic programming and handles uncertainty using the Bilattice framework [Ginsberg, 1988]. The knowledge base consists of domain-specific rules, expressing CEs in terms of SDEs. Each CE or SDE is associated with two uncertainty values, indicating a degree of information and confidence respectively. The underlying idea of the method is that the more confident information is provided, the stronger the recognition belief for the corresponding CE becomes.

Filippaki et al. [2011] proposed a logic-based method that recognises user activities over noisy or incomplete data. The method recognises CEs from SDEs using rules that impose temporal and spatial constraints between SDEs. Some of the constraints in CE definitions are optional. As a result, a CE can be recognised from incomplete information, but with lower confidence. The confidence of a CE increases when more of the optional SDEs are recognised. Due to noisy or incomplete information, the recognised CEs may be logically inconsistent with each other. The method resolves those inconsistencies using the confidence, duration and number of involved SDEs. Another method that handles missing SDEs is presented in Romdhane et al. [2013]. The method extends the scenario recognition method of Vu et al. [2003], in order to deal with uncertain and incomplete knowledge. Sub-events of a CE are associated with *utility* values in the definition of the CE, in order to express the importance or priority of sub-events in the recognition of the CE. The higher the utility value, the higher the importance of the sub-event. As a result, in situations where the stream of input SDEs becomes incomplete (e.g., due to some type of error or noise) and some SDEs are missing the method can still recognise the target CE. Using a hierarchical Bayesian inference approach, the method can handle uncertainty at the input level, as well as at the event recognition level of a CE.

Probabilistic graphical models have been successfully applied to a variety of event recognition tasks where a significant amount of uncertainty exists. Since event recognition requires the processing of streams of time-stamped SDEs, numerous event recognition methods are based on sequential variants of probabilistic graphical models, such as Hidden Markov Models (HMM) [Rabiner and Juang, 1986], Dynamic Bayesian Networks (DBN) [Murphy, 2002] and linear-chain Conditional Random Fields (CRF) [Lafferty et al., 2001]. Such models can naturally handle uncertainty but their propositional structure provides limited representation capabilities. To overcome this limitation, graphical models have been extended to model interactions between multiple entities [Brand et al., 1997; Gong and Xiang, 2003; Vail et al., 2007; Wu et al., 2007], to capture long-term dependencies between states [Hongeng and Nevatia, 2003] and to model the hierarchical composition of events [Liao et al., 2005; Natarajan and Nevatia, 2007]. However, the lack of a formal representation language makes the definition of structured CEs complicated and the use of background knowledge very hard.

Recently, statistical relational learning (SRL) methods have been applied to event recognition. These methods combine logic with probabilistic models, in order to represent

complex relational structures and perform reasoning under uncertainty. Using a declarative language as a template, SRL methods specify probabilistic models at an abstract level. Given an input stream of SDE observations, the template is partially or completely instantiated, creating lifted or propositional graphical models on which probabilistic inference is performed [de Raedt and Kersting, 2010; de Salvo Braz et al., 2008].

Among others, HMMs have been extended in order to represent states and transitions using logical expressions [Kersting et al., 2006; Natarajan et al., 2008]. In contrast to standard HMM, the logical representation allows the model to represent compactly probability distributions over sequences of logical atoms, rather than propositional symbols. Similarly, DBNs have been extended using first-order logic [Manfredotti, 2009; Manfredotti et al., 2010]. A tree structure is used, where each node corresponds to a first-order logic expression, e.g., a predicate representing a CE, and can be related to nodes of the same or previous time instances. Compared to their propositional counterparts, the extended HMM and DBN methods can compactly represent CE that involve various entities.

Markov Logic Networks have also recently been used for event recognition in the literature. The knowledge base of weighted first-order logic formulas in MLNs defines an arbitrarily structured undirected graphical model. Therefore, MLNs provide a generic SRL framework, which subsumes various graphical models, e.g., HMM, CRF, etc., and can be used with expressive logic-based formalisms, such as the Event Calculus. Additionally, MLNs support discriminative modelling and thus avoid common independence assumptions over the input SDEs. Biswas et al. [2007] proposed a first-order probabilistic method that models dynamic MLNs. The method combines the information provided by different low-level computer vision algorithms with the use of MLNs, in order to recognise CEs that represent human activities from video data. CEs are represented by fluents that persist over successive time-points. The formalism does not support the simultaneous occurrence of CEs and requires that at each time-point a CE must occur. A more expressive approach that can represent persistent and concurrent CEs, as well as their starting and ending points, is proposed by Helaoui et al. [2011]. However, that method has a quadratic complexity to the number of time-points.

Tran and Davis [2008] address the issues of missing and noisy SDEs, as well as imperfect definitions of CEs, in activity recognition. CEs are defined by common sense domain knowledge and expressed in first-order logic, which may not capture perfectly the conditions under which CEs occur. SDEs are detected by low-lever computer vision algorithms with some degree of belief. The degree is propagated inside the MLN using utility weighted formulas. Rules with disjunctive effects help the recognition of missing SDEs. CE definitions are associated with weight values indicating a degree of certainty, that is defined manually by the domain expert. The temporal model of CEs is represented through temporal intervals that are defined by a specific number of video frames. Each CE rule is defined over events that happened in the same interval. Morariu and Davis

[2011] proposed an MLN-based method that uses employs temporal constraints over interval relations in events. Similar to Tran and Davis [2008] the method uses first-order logic representation, but it employs temporal relations from the Interval Algebra of Allen [1983]. The ontology of the knowledge base is composed of properties, event definitions and observations. Properties are expressed by predicates that represent the state of the world in each temporal interval. Spatio-temporal prerequisites of relevant properties are expressed by event definitions, using interval constraints over the occurrence of sub-events. Observations provide the evidence and are generated by computer vision algorithms. The method determines the sequence of CEs that is most likely, given the observation sequences. In order to avoid the combinatorial explosion of possible intervals and eliminate the existential quantifiers in CE definitions, a bottom-up process eliminates the unlikely event hypotheses. Their elimination process is guided by the observations and the interval relations of the event definitions. Song et al. [2013a,b] also combine MLNs with Allen's Interval Algebra relations to recognise and forecast CE. Computer vision algorithms detect and track the state of objects (e.g., items, human hands, etc.), which is expressed in terms of symbolic features that span over time intervals — i.e., spatial and qualitative and object to object relations that occur for periods of time. Events are generated by combining the features. The events are hierarchical and related with part-of relations. Furthermore, Song et al. propose a set of domain-independent prediction axioms. These axioms express that the occurrence of an event implies the occurrence of its parts. Constraint axioms ensure the integrity of the (temporal) relations among CE and its parts. Finally, a set of abduction axioms allow events to be inferred on the basis of their parts — i.e., inferring when some events are missing.

In the aforementioned MLN-based methods intervals are not inferred, but provided as input evidence. A different approach to interval-based activity recognition, is the Probabilistic Event Logic (PEL) [Brendel et al., 2011; Selman et al., 2011]. Similar to MLNs, the method defines a log-linear model from a set of weighted formulas, but the formulas are represented in Event Logic [Siskind, 2001]. Each formula defines a soft constraint over some events, using interval relations that are represented by the *spanning intervals* data structure. The method performs inference via a local-search algorithm (based on the MaxWalkSAT of Kautz et al. [1997]), but using the spanning intervals it avoids grounding all possible time intervals.

Sadilek and Kautz [2012] employ hybrid-MLNs [Wang and Domingos, 2008] in order to recognise successful and failed interactions between humans, using noisy location data from GPS devices. The method uses hybrid formulas that de-noise the location data. Hybrid formulas are defined as normal soft-constrained formulas, but their weights are also associated with real-valued functions, e.g., the distance between two persons. As a result, the strength of the constraint that a hybrid rule imposes is determined by both its weight and function — e.g., the smaller the distance between two people, the stronger the constraint. The weights are estimated from training data with supervised learning.

Furthermore, Sadilek and Kautz also proposed a knowledge refinement method that revises the CE definitions, in order to achieve higher recognition accuracy.

Helaoui et al. [2013] proposed an activity recognition system that employs the probabilistic framework of log-linear Description Logics [Niepert et al., 2011]. Similar to MLNs, Log-Linear Description Logics combine a logic-based representation with log-linear modelling. In particular, the method can represent events (SDEs or CEs) and CE definitions expressed in the OWL 2 Web Ontology Language [Grau et al., 2008] and incorporate both probabilistic and deterministic dependencies between the CE definitions. CE definitions can be associated with predefined weight values, indicating a degree of confidence. One major limitation is that the OWL 2 language does not support temporal reasoning [Riboni et al., 2011]. To overcome this limitation, the method recognises events progressively. Starting from the input level (e.g., atomic gestures of humans), events are collected within a specified time window and are added to the ontology as a sequence of events — i.e., each event has a property that defines its order of execution within the time window. Thereafter, using maximum a-posteriori reasoning, the recognised events of the current level (e.g., CE representing the activities of a single person) forms the input for the next level (e.g., CE representing complex activities between multiple persons).

Wasserkrug et al. [2005, 2008, 2012] introduced a generic framework for probabilistic event processing. The representation of CE definitions is logic-based with temporal constraints over time-points and intervals. To model the uncertainty, input SDEs and CE definitions are associated with probability values. The knowledge base of CE definitions forms a template for a Bayesian network that is dynamically constructed every time new SDE is received. CEs are continuously recognised using probabilistic inference over the Bayesian network.

Cugola et al. [2014] extend the TESLA [Cugola and Margara, 2010] event specification language with probabilistic modelling, in order to handle the uncertainty in input SDEs, as well as recognise CEs that have imperfect definitions. The semantics of the TESLA language are formally specified by a first-order logical representation with temporal constraints, which expresses the length of time intervals quantitatively [Ghezzi et al., 1990; Morzenti et al., 1992] — e.g., an event X must occur within five minutes after the occurrence of some other event Y. At the input level, the method supports the uncertainty in the occurrence of SDEs, as well as the uncertainty regarding the attributes of the SDEs. In the former case, SDEs are associated with probabilities that indicate a degree of belief. In the latter case, the attributes of a SDE are modelled as random variables with some measurement error. The probability distribution function of the measurement error is assumed to be known (e.g., Gaussian distribution). The method also models the uncertainty of CE definitions, by building a Bayesian network for each rule. The probabilistic parameters of the network are manually estimated by domain experts.

2.4.4 Discussion

In general, CEs are defined by complex event patterns and their recognition depends on rich temporal (e.g., sequence, duration, temporal distance, etc.) and logical (e.g., conjunction, disjunction, negation, etc.) relationships among their sub-events. Action formalisms and symbolic event recognition methods can compactly represent and reason about CEs (see Sections 2.4.1 and 2.4.2). Their formal and declarative semantics can naturally represent CEs and domain knowledge. Such knowledge-driven approaches have been used in numerous applications that combine information from heterogeneous sources — e.g., video processing algorithms, sensor networks, computer network traffic, etc. In many cases, however, the knowledge is imperfect and the input sources are noisy. As a result, the performance of knowledge-driven approaches may be seriously compromised. On the other hand, pure data-driven methods employ probabilistic modelling and handle uncertainty naturally — e.g., Hidden Markov Models, Dynamic Bayesian networks, Conditional Random Fields, etc. Although data-driven approaches are robust to uncertainty, they cannot represent directly domain knowledge and CE definitions. Furthermore, in order to represent rich temporal and logical relationships between events, their structure may become prohibitively complex with a large number of parameters. Furthermore, purely data-driven models require large amounts of training data, which is difficult to obtain in practice.

Symbolic methods augmented with probabilistic modelling combine the advantages of knowledge-driven and data-driven approaches in a single *hybrid* model (see Section 2.4.3). Such models can directly and compactly represent knowledge with rich temporal and logical relations, while handling various forms of uncertainty. The structure of the hybrid model is based on the given domain knowledge — e.g., logic-based rules. Depending on the application, the parameters of the model (i.e., probabilities or weight values) can be either manually specified by experts or automatically estimated from training data using machine learning techniques. As the number of the parameters grows, however, their manual setting becomes tedious and error prone — especially in methods that rely on probabilistic graphical models. In contrast to pure data-driven approaches, hybrid methods exploit the given domain knowledge in order to reduce the required amount of training data. Compared to knowledge-driven approaches, hybrid methods consider, model and handle uncertainty. First, there are hybrid methods that can directly incorporate the uncertainty of input SDEs (e.g., Lavee et al. [2013]; Shet et al. [2007]; Tran and Davis [2008]). Since domain knowledge is usually imperfect, hybrid methods can also associate CEs with probabilities, indicating a degree of confidence (e.g., Molinaro et al. [2014]; Morariu and Davis [2011]; Song et al. [2013a]). Furthermore, by combining background knowledge and probabilistic modelling hybrid methods can also handle situations where the input data become incomplete (e.g., Romdhane et al. [2013]; Ryoo and Aggarwal [2009]; Tran and Davis [2008]).

In contrast to action formalisms, the majority of the hybrid methods do not employ any generic formalism for representing the events and their effects and thus their knowledge base is restricted to domain-dependent CE definitions (e.g., Kersting et al. [2006], Biswas et al. [2007], Albanese et al. [2010], Shet et al. [2011], Lavee et al. [2013], etc.). There are methods that employ some domain-independent ontology for expressing temporal constraints, but their knowledge base is still composed of domain-dependent definitions (e.g., Morariu and Davis [2011], Brendel et al. [2011], Selman et al. [2011], etc). Domain-independent axioms characterise formally the properties and the behaviour of the formalism, making it easier to use in different event recognition applications. For instance, the hybrid method of Song et al. [2013a,b] incorporates domain-independent axioms, in order to support prediction, retain the temporal integrity and infer missing events. A further difference, between action formalisms and the hybrid methods presented in this section, is that very few of the latter, namely those Biswas et al. [2007] and Helaoui et al. [2011], support a form of inertia. However, even in these methods introduce elements of inertia into the domain-dependent definitions, thus limiting their applicability to the specific event recognition applications.

The methods that we have developed in this work combine a formal, logic-based representation with probabilistic modelling. Similar to the hybrid approaches, the developed methods can compactly express CE definitions and employ background domain knowledge, while probabilistic modelling allows them to handle various forms of uncertainty. For the probabilistic modelling we employ the state-of-the-art probabilistic and relational frameworks of MLNs and ProbLog. Both frameworks provide powerful logic-based representation that allows to knowledge with complex relations among events. We exploit their generic representation capabilities, in order to incorporate the Event Calculus action formalism. We chose the Event Calculus, in order to provide a approach to reasoning about events and their effects. Furthermore, with the linear temporal model of the Event Calculus, we can directly define temporal constraints in the CE definitions. Additionally, the developed methods inherit and extend the domain-independent property of inertia with probabilistic modelling.

3 | MLN–EC: Probabilistic Event Calculus based on Markov Logic Networks

"The probable is what usually happens."
— Aristotle

As mentioned in Section 1.1, logic-based event recognition approaches can compactly represent complex event relations and background knowledge. However, event recognition systems often have to deal with data that involves a significant amount of uncertainty. A missed or an erroneous simple, derived event (SDE) detection may have a significant effect on the event recognition results, e.g., causing the erroneous recognition of a composite event (CE). When Machine Learning techniques are used, the training set often involves inconsistent CE annotations. Furthermore, CE definitions and background knowledge, either learnt from data or derived by domain experts, cannot strictly follow the annotation. The definitions, therefore, cannot capture perfectly the conditions under which CEs occur. Under such situations of uncertainty, the performance of an Event Recognition system can be seriously compromised.

To deal with uncertainty and retain the advantages of logic-based representation, we combine a discrete variant of the Event Calculus with the probabilistic framework of Markov Logic Networks. Figure 3.1 outlines the structure of the method (MLN–EC). The input to MLN–EC is composed of a stream of SDE occurrences and a set of domain-dependent CE definitions. The CE definitions take the form of common-sense rules and describe the conditions under which a CE starts or ends. The method combines the given definitions with the domain-independent axioms of MLN–EC, generating a compact knowledge base that forms a template for the production of Markov Networks. The method employs Markov Logic Networks, in order to perform learning and probabilistic inference. Learning takes place once, in order to estimate the parameters of the model from data. Using probabilistic inference, MLN–EC recognises the CE of interest for the given input stream of SDEs.

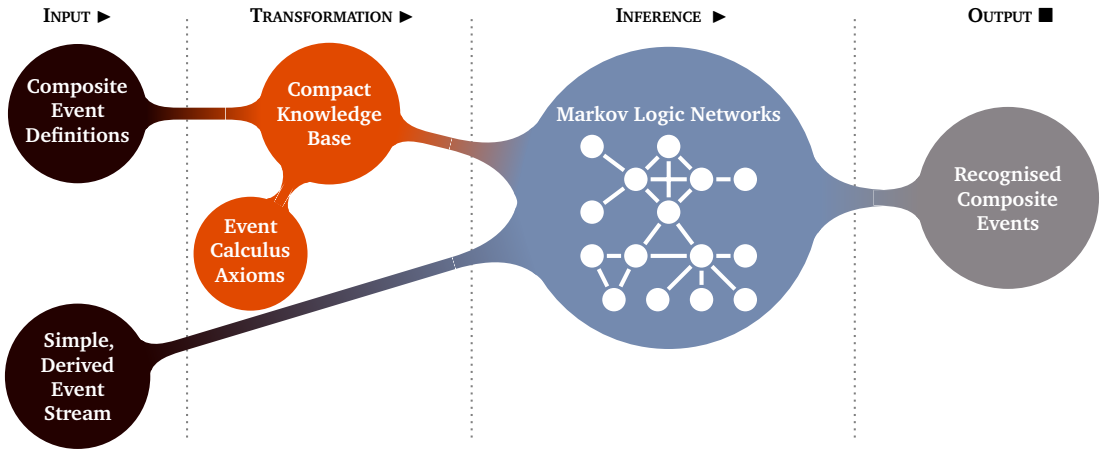


FIGURE 3.1: The structure of the MLN–EC

3.1 Axiomatisation

We base our model on an axiomatisation of a discrete version of the Event Calculus (DEC) in first-order logic (see Section 2.2). 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 3.1 summarises the main elements of the proposed Event Calculus (MLN–EC). 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
$\texttt{happens}(E, T)$	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 3.1: The MLN–EC predicates.

The MLN–EC axioms that determine when a fluent holds are defined as follows:

$$\texttt{holdsAt}(F, T{+}1) \Leftarrow \texttt{initiatedAt}(F, T) \tag{3.1}$$

$$\texttt{holdsAt}(F, T{+}1) \Leftarrow \texttt{holdsAt}(F, T) \wedge \neg\texttt{terminatedAt}(F, T) \tag{3.2}$$

Axiom (3.1) defines that if a fluent F is initiated at time T, then it holds at the next time-point. Axiom (3.2) specifies that a fluent continues to hold unless it is terminated.

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

$$\begin{aligned} \neg\texttt{holdsAt}(F, T{+}1) \Leftarrow \\ \qquad \texttt{terminatedAt}(F, T) \end{aligned} \tag{3.3}$$

$$\begin{aligned} \neg\texttt{holdsAt}(F, T{+}1) \Leftarrow \\ \qquad \neg\texttt{holdsAt}(F, T) \wedge \\ \qquad \neg\texttt{initiatedAt}(F, T) \end{aligned} \tag{3.4}$$

According to axiom (3.3), if a fluent F is terminated at time T then it does not hold at the next time-point. Axiom (3.4) states that a fluent continues not to hold unless it is initiated.

The predicates `happens`, `initiatedAt` and `terminatedAt` are defined only in a domain-dependent manner. `happens` expresses the input evidence, determining the occurrence of a SDE at a specific time-point. The input stream of observed SDEs, therefore, is represented in the MLN–EC as a *narrative* of ground `happens` predicates. `initiatedAt` and `terminatedAt` specify under which circumstances a fluent — representing a CE — is to be initiated or terminated at a specific time-point. The domain-dependent rules of MLN–EC, 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:

$$\begin{aligned} &\texttt{initiatedAt}(\texttt{fluent}_1(X, Y), T) \Leftarrow \\ &\qquad \texttt{happens}(\texttt{event}_\texttt{i}(X), T) \ \wedge \ \ldots \ \wedge \\ &\qquad \texttt{holdsAt}(\texttt{fluent}_\texttt{j}(X), T) \ \wedge \ \ldots \ \wedge \\ &\qquad \texttt{Conditions}[X, Y, T] \\ &\texttt{terminatedAt}(\texttt{fluent}_1(X, Y), T) \Leftarrow \\ &\qquad \texttt{happens}(\texttt{event}_\texttt{k}(X), T) \ \wedge \ \ldots \ \wedge \\ &\qquad \texttt{holdsAt}(\texttt{fluent}_1(X), T) \ \wedge \ \ldots \ \wedge \\ &\qquad \texttt{Conditions}[X, Y, T] \end{aligned} \tag{3.5}$$

In this work we consider finite domains of time-points, events and fluents, that 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 (3.5) 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 `happens` 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 `happens` 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 `happens`, `holdsAt` and $\texttt{Conditions}[X, Y, T]$ is optional and varies according to the requirements of the target event recognition application.

3.2 Application to Activity Recognition

As it has been presented in Section 2.1, to demonstrate our method we apply it to video surveillance in public spaces using the publicly available benchmark dataset of the CAVIAR project. The aim is to recognise CE that involve multiple persons, by exploiting information about observed SDEs. In this work, we focus on the recognition of the `meeting` and `moving` CEs, for which the dataset contains a sufficient amount of training examples.

The input to `MLN–EC` is a narrative of ground `happens` predicates (SDEs), representing people *walking*, *running*, staying *active*, or *inactive*. The first and the last time that a person or an object is tracked are represented by the SDEs *enter* and *exit*. The coordinates of tracked persons or objects are preprocessed and represented by the utility predicates `close` and `orientationMove`, expressing qualitative spatial relations. As an example, consider the following fragment of a narrative:

$$\begin{aligned}
&\ldots\\
&\texttt{happens}(\texttt{walking}(id_1),\ 100)\\
&\texttt{happens}(\texttt{walking}(id_2),\ 100)\\
&\texttt{orientationMove}(id_1,\ id_2,\ 100)\\
&\texttt{close}(id_1,\ id_2,\ 34,\ 100)\\
&\ldots\\
&\texttt{happens}(\texttt{active}(id_1),\ 200)\\
&\texttt{happens}(\texttt{active}(id_2),\ 200)\\
&\ldots
\end{aligned} \tag{3.6}$$

According to the above narrative, it has been observed that at time-point 100 the persons id_1 and id_2 are *walking* close to each other with similar orientation. The ground predicate `close` indicates that the distance between the two persons at time-point 100 is below the specified threshold of 34 pixels. Similarly, `orientationMove` states that their orientation is almost the same (e.g., the difference is below 45 degrees). Later at time-point 200 both persons are staying active, e.g., they are moving their arms, while staying at the same position.

The input CE definitions of the `meeting` and `moving` activities were developed in [Artikis et al., 2010b]. These definitions are common-sense rules that are expressed using the general form (3.5). Consider the following definition of the `meeting` CE in our example.

$$\begin{aligned}
\texttt{initiatedAt}(\texttt{meeting}(ID_1,\ ID_2),\ T) \Leftarrow\\
\texttt{happens}(\texttt{active}(ID_1),\ T)\ \wedge\\
\neg\texttt{happens}(\texttt{running}(ID_2),\ T)\ \wedge\\
\texttt{close}(ID_1,\ ID_2,\ 25,\ T)
\end{aligned} \tag{3.7}$$

$$\begin{aligned}\texttt{initiatedAt}(\texttt{meeting}(ID_1,\, ID_2),\, T) \Leftarrow\\ \texttt{happens}(\texttt{inactive}(ID_1),\, T) \wedge\\ \neg\texttt{happens}(\texttt{running}(ID_2),\, T) \wedge\\ \neg\texttt{happens}(\texttt{active}(ID_2),\, T) \wedge\\ \texttt{close}(ID_1,\, ID_2,\, 25,\, T)\end{aligned} \tag{3.8}$$

$$\begin{aligned}\texttt{terminatedAt}(\texttt{meeting}(ID_1,\, ID_2),\, T) \Leftarrow\\ \texttt{happens}(\texttt{walking}(ID_1),\, T) \wedge\\ \neg\texttt{close}(ID_1,\, ID_2,\, 34,\, T)\end{aligned} \tag{3.9}$$

$$\begin{aligned}\texttt{terminatedAt}(\texttt{meeting}(ID_1,\, ID_2),\, T) \Leftarrow\\ \texttt{happens}(\texttt{running}(ID_1),\, T)\end{aligned} \tag{3.10}$$

$$\begin{aligned}\texttt{terminatedAt}(\texttt{meeting}(ID_1,\, ID_2),\, T) \Leftarrow\\ \texttt{happens}(\texttt{exit}(ID_1),\, T)\end{aligned} \tag{3.11}$$

According to rules (3.7) and (3.8), the `meeting` CE is initiated when the people involved interact with each other, i.e., at least one of them is active or inactive, the other is not running, and the measured distance between them is at most 25 pixels. The `meeting` CE is terminated when people walk away from each other (rule 3.9), when one of them is running (rule 3.10), or has exited the scene (rule 3.11).

The definition of the CE that people are `moving` together is represented as follows:

$$\begin{aligned}\texttt{initiatedAt}(\texttt{moving}(ID_1,\, ID_2),\, T) \Leftarrow\\ \texttt{happens}(\texttt{walking}(ID_1),\, T) \wedge\\ \texttt{happens}(\texttt{walking}(ID_2),\, T) \wedge\\ \texttt{orientationMove}(ID_1,\, ID_2,\, T) \wedge\\ \texttt{close}(ID_1,\, ID_2,\, 34,\, T)\end{aligned} \tag{3.12}$$

$$\begin{aligned}\texttt{terminatedAt}(\texttt{moving}(ID_1,\, ID_2),\, T) \Leftarrow\\ \texttt{happens}(\texttt{walking}(ID_1),\, T) \wedge\\ \neg\texttt{close}(ID_1,\, ID_2,\, 34,\, T)\end{aligned} \tag{3.13}$$

$$\begin{aligned}\texttt{terminatedAt}(\texttt{moving}(ID_1,\, ID_2),\, T) \Leftarrow\\ \texttt{happens}(\texttt{active}(ID_1),\, T) \wedge\\ \texttt{happens}(\texttt{active}(ID_2),\, T)\end{aligned} \tag{3.14}$$

$$\begin{aligned}\texttt{terminatedAt}(\texttt{moving}(ID_1,\, ID_2),\, T) \Leftarrow\\ \texttt{happens}(\texttt{active}(ID_1),\, T) \wedge\\ \texttt{happens}(\texttt{inactive}(ID_2),\, T)\end{aligned} \tag{3.15}$$

$$\begin{aligned}\texttt{terminatedAt}(\texttt{moving}(ID_1,\, ID_2),\, T) \Leftarrow\\ \texttt{happens}(\texttt{running}(ID_1),\, T)\end{aligned} \tag{3.16}$$

$$\texttt{terminatedAt}(\texttt{moving}(ID_1, ID_2), T) \Leftarrow \texttt{happens}(\texttt{exit}(ID_1), T) \tag{3.17}$$

According to rule (3.12), the `moving` CE is initiated when two persons ID_1 and ID_2 are walking close to each other (their distance is at most 34 pixels) with almost the same orientation. The `moving` CE is terminated under several cases: (a) When people walk away from each other, i.e., they have a distance larger than 34 pixels, (b) when they stop moving, i.e., either both are active, or one is active while the other is inactive, represented by rules (3.14) and (3.15). (c) When one of them is running or exiting the scene, represented by rules (3.16) and (3.17), respectively.

The output of MLN–EC is a stream of recognised activities. The clauses that are produced from MLN–EC axioms (i.e., axioms (3.1)–(3.4)) and input CE definitions (i.e., rules (3.7)–(3.17)) are associated with weights. Given a narrative of SDEs, MLN–EC probabilistically determines whether a CE holds or not at any time-point. For example, the SDEs and spatial conditions at time-points 100 and 200, according to narrative (3.6), satisfy the conditions of rules (3.12) and (3.14), respectively. As it is shown in Figure 3.2, at time-point 100 the `moving` CE is initiated. This situation increases the belief that CE `moving` will hold at the next time-point — i.e., $\texttt{holdsAt}(\texttt{moving}(id_1, id_2), 101)$. On the other hand, at time-point 200 the CE `moving` is terminated. As a result, this situation reduces the belief that `moving` holds at time-point 201. Therefore, according to this example the CE `moving` has been recognised with some probability in the interval $(100, 200]$. In the following sections we give a detailed description of how clause weights are determined, affecting the probability of the recognised CEs.

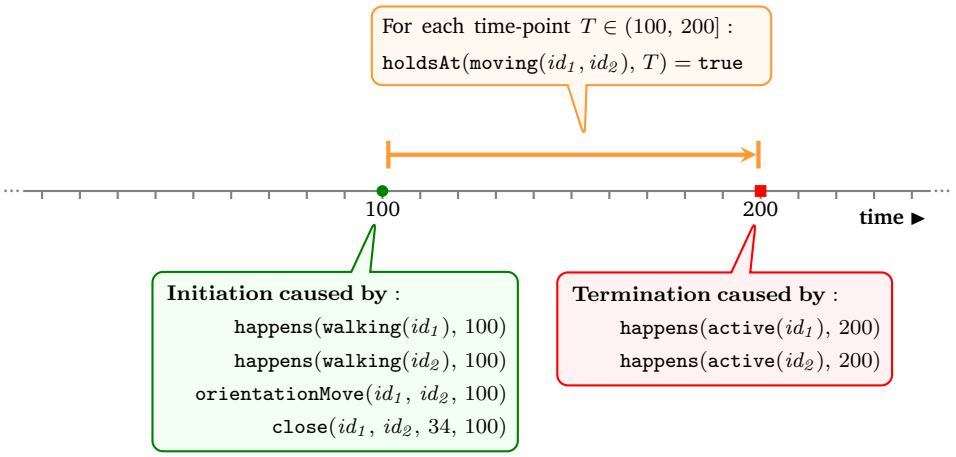


FIGURE 3.2: An example of event recognition with the MLN–EC

3.3 Compact Knowledge Base Construction

The use of MLNs for inference and learning requires the grounding of the entire knowledge base used for event recognition. Unless optimised, this process leads to unmanageably large ground Markov Networks, where inference and learning become practically infeasible. We address this problem at two levels:

1. We use the simplified dialect of the Event Calculus that is presented in Section 3.1.
2. We extend MLNs with a preprocessing step that transforms the knowledge base into a logically stronger one. The simplifications that are made during the preprocessing reduce the number of random variables in the ground Markov network, simplifying the complexity of inference and learning.

3.3.1 Simplified Representation

The choice of Event Calculus dialect, as presented in Section 3.1, has a significant impact on the grounding process. For example, Shanahan's Full Event Calculus [Shanahan, 1999] employs axioms that contain triply quantified time-point variables (see Section 2.2). As a result, the number of their groundings has a cubic relation to the number of time-points. Furthermore, that formalism contains existentially quantified variables over events and time-points. During MLN grounding existentially quantified formulas are replaced by the disjunction of their groundings [Domingos and Lowd, 2009]. This leads to a large number of disjunctions and a combinatorial explosion of the number of clauses, producing unmanageably large Markov networks.

In contrast, the proposed Event Calculus (MLN–EC) is based on the DEC [Mueller, 2008], where the domain-independent axioms are defined over successive time-points (see Section 2.2). The inertia axioms (2.8) and (2.10) of DEC, however, still contain the existentially quantified variable E over a conjunction of predicates. Each of these axioms will be transformed into $2^{|\mathcal{E}|}$ clauses, each producing $|\mathcal{F}| \times |\mathcal{T}|$ groundings[1]. Moreover, each ground clause will contain a large number of disjunctions, causing large cliques in the ground Markov network. On the other hand, in MLN–EC the initiation and termination predicates are only defined in terms of fluents and time-points (see the general form (3.5)). This representation can be used to reduce further the number of variables and eliminate the existential quantification in the domain-independent axioms. For example, axiom (3.1) produces one clause in Conjunctive Normal Form and has two distinct variables F and T. Therefore, the number of its groundings is determined by the Cartesian

[1] Mueller [2008] uses the technique of sub-formula renaming [Nonnengart and Weidenbach, 2001] in order to avoid the creation of $2^{|\mathcal{E}|}$ clauses. However, the existential quantification remains in the axioms, causing the creation of large cliques in the ground network. Additionally, sub-formula renaming introduces utility predicates that do not belong in the input evidence, creating hidden variables in the network. As a result, the complexity of inference and learning increases.

product of the corresponding variable-binding constraints, that is $|\mathcal{F}|\times|\mathcal{T}|$. Assuming that the domain of fluents $\mathcal{F}$ is relatively small compared to the domain of time-points $\mathcal{T}$, the number of groundings of axiom (3.1) grows linearly to the number of time-points. Thus, compared to the Discrete Event Calculus and the Full Event Calculus (see Section 2.2), MLN–EC produces a substantially smaller number of ground clauses.

3.3.2 Knowledge Base Transformation

In addition to choosing an Event Calculus dialect that makes the number of ground clauses linearly dependent on the number of time-points, we can achieve significant improvements in the size of the ground Markov Networks, by making the closed-world assumption.

A knowledge base with domain-dependent rules in the form of (3.5) describes explicitly the conditions in which fluents are initiated or terminated. It is usually impractical to define also when a fluent is *not* initiated and *not* terminated. However, the open-world semantics of first-order logic result in an inherent uncertainty about the value of a fluent for many time-points. In other words, if at a specific time-point no event that terminates or initiates a fluent happens, we cannot rule out the possibility that the fluent has been initiated or terminated. As a result, we cannot determine whether a fluent holds or not, leading to the loss of inertia.

This is a variant of the well-known frame problem and one solution for the Event Calculus in first-order logic is the use of circumscription [Doherty et al., 1997; Lifschitz, 1994; McCarthy, 1980; Mueller, 2008; Shanahan, 1997]. The aim of circumscription is to automatically rule out all those conditions which are not explicitly entailed by the given formulas. Hence, circumscription introduces a closed-world assumption to first-order logic.

Technically, we perform circumscription by predicate completion — a syntactic transformation where formulas are translated into logically stronger ones. In particular, we perform predicate completion on `initiatedAt` and `terminatedAt` predicates. Due to the form of CE definitions (see formalisation (3.5)), the result of predicate completion concerns each CE separately, e.g., $\texttt{initiatedAt}(\texttt{meeting}(ID_1, ID_2), T)$, rather than a generic $\texttt{initiatedAt}(F, T)$ predicate. Similar to Mueller [2008], we also eliminate the `initiatedAt` and `terminatedAt` predicates from the knowledge base, by exploiting the equivalences resulting from predicate completion. In cases where the definitions of the initiation or termination of a specific CE are missing, the corresponding initiation or termination is considered *False* for all time-points, e.g., $\texttt{terminatedAt}(\texttt{fluent}(X, Y), T) \Leftrightarrow False$.

To illustrate the form of the resulting knowledge base, consider the domain-dependent definition of `meeting` — i.e., rules (3.7)–(3.11). After predicate completion, these rules

will be replaced by the following formulas:

$$\begin{aligned}
&\texttt{initiatedAt}(\texttt{meeting}(ID_1,\, ID_2),\, T) \Leftrightarrow \\
&\qquad \big(\texttt{happens}(\texttt{active}(ID_1),\, T) \wedge \\
&\qquad \neg\texttt{happens}(\texttt{running}(ID_2),\, T) \wedge \\
&\qquad \texttt{close}(ID_1,\, ID_2,\, 25,\, T)\big) \bigvee \\
&\qquad \big(\texttt{happens}(\texttt{inactive}(ID_1),\, T) \wedge \\
&\qquad \neg\texttt{happens}(\texttt{running}(ID_2),\, T) \wedge \\
&\qquad \neg\texttt{happens}(\texttt{active}(ID_2),\, T) \wedge \\
&\qquad \texttt{close}(ID_1,\, ID_2,\, 25,\, T)\big)
\end{aligned} \tag{3.18}$$

$$\begin{aligned}
&\texttt{terminatedAt}(\texttt{meeting}(ID_1,\, ID_2),\, T) \Leftrightarrow \\
&\qquad \big(\texttt{happens}(\texttt{walking}(ID_1),\, T) \wedge \\
&\qquad \neg\texttt{close}(ID_1,\, ID_2,\, 25,\, T)\big) \bigvee \\
&\qquad \texttt{happens}(\texttt{running}(ID_1),\, T) \bigvee \\
&\qquad \texttt{happens}(\texttt{exit}(ID_1),\, T)
\end{aligned} \tag{3.19}$$

The resulting rules (3.18) and (3.19) define all conditions under which the `meeting` CE is initiated or terminated. Any other event occurrence cannot affect this CE, as it cannot initiate the CE or terminate it. Based on the equivalence in formula (3.18), the domain-independent axiom (3.1) is automatically re-written into the following specialised form[2]:

$$\begin{aligned}
&\texttt{holdsAt}(\texttt{meeting}(ID_1,\, ID_2),\, T{+}1) \Leftarrow \\
&\qquad \texttt{happens}(\texttt{active}(ID_1),\, T) \wedge \\
&\qquad \neg\texttt{happens}(\texttt{running}(ID_2),\, T) \wedge \\
&\qquad \texttt{close}(ID_1,\, ID_2,\, 25,\, T) \\
&\\
&\texttt{holdsAt}(\texttt{meeting}(ID_1,\, ID_2),\, T{+}1) \Leftarrow \\
&\qquad \texttt{happens}(\texttt{inactive}(ID_1),\, T) \wedge \\
&\qquad \neg\texttt{happens}(\texttt{running}(ID_2),\, T) \wedge \\
&\qquad \neg\texttt{happens}(\texttt{active}(ID_2),\, T) \wedge \\
&\qquad \texttt{close}(ID_1,\, ID_2,\, 25,\, T)
\end{aligned} \tag{3.20}$$

[2]This direct re-writing of (3.18) results in a single formula that contains the disjunction found in formula (3.18). However, for reasons that have to do with the handling of uncertainty in MLN and will be discussed in a later section, in (3.20) we choose to equivalently represent it using two separate formulas.

Similarly, the inertia axiom (3.2) can be re-written according to (3.19) as follows:

$$\begin{aligned}\texttt{holdsAt}(\texttt{meeting}(ID_1,\, ID_2),\, T{+}1) \Leftarrow\\ \texttt{holdsAt}(\texttt{meeting}(ID_1,\, ID_2),\, T) \wedge\\ \neg\Big(\big(\texttt{happens}(\texttt{walking}(ID_1),\, T) \wedge\\ \neg\texttt{close}(ID_1,\, ID_2,\, 25,\, T)\big) \bigvee\\ \texttt{happens}(\texttt{running}(ID_1),\, T) \bigvee\\ \texttt{happens}(\texttt{exit}(ID_1),\, T)\Big)\end{aligned} \tag{3.21}$$

The result of this transformation procedure replaces the original set of domain-independent axioms and domain-dependent CE definitions with a logically stronger knowledge base. The rules in the resulting knowledge base form the template that MLNs will use to produce ground Markov networks. The transformed formulas produce considerably more compact ground Markov networks than the original ones, as the clauses to be grounded are reduced. Moreover, the predicates `initiatedAt` and `terminatedAt` are eliminated and the corresponding random variables are not added to the network. This reduction decreases substantially the space of possible worlds, since the target random variables of the network (Y in equation (2.11)) are limited only to the corresponding `holdsAt` ground predicates. Specifically, the space of possible worlds is reduced from $2^{3\times|\mathcal{F}|\times|\mathcal{T}|}$ to $2^{|\mathcal{F}|\times|\mathcal{T}|}$ — where $|\mathcal{T}|$ and $|\mathcal{F}|$ denote the number of distinct time-points and fluents, respectively. These reductions improve the computational performance of the probabilistic inference. Additionally, due to the reduced space of possible worlds, the same number of sampling iterations results in better probability estimates.

Formally, the resulting knowledge base is composed of rules having the following form:

$$\Sigma = \begin{cases} \texttt{holdsAt}(\texttt{fluent}_1(X,\, Y),\, T{+}1) \Leftarrow & \\ \quad \texttt{happens}(\texttt{event}_\texttt{i}(X),\, T) \wedge \ldots \wedge \texttt{Conditions}[X,Y,T] & (3.22)\\ \ldots & \\ \neg\texttt{holdsAt}(\texttt{fluent}_1(X,\, Y),\, T{+}1) \Leftarrow & \\ \quad \texttt{happens}(\texttt{event}_\texttt{j}(X),\, T) \wedge \ldots \wedge \texttt{Conditions}[X,Y,T] & (3.23)\\ \ldots & \end{cases}$$

$$\Sigma' = \begin{cases} \texttt{holdsAt}(\texttt{fluent}_1(X,\, Y),\, T{+}1) \Leftarrow & \\ \quad \texttt{holdsAt}(\texttt{fluent}_1(X,\, Y),\, T) \wedge & \\ \quad \neg\big(\,(\,\texttt{happens}(\texttt{event}_\texttt{j}(X),\, T) \wedge \ldots \wedge \texttt{Conditions}[X,Y,T]\,) \bigvee \ldots\big) & (3.24)\\ \ldots & \\ \neg\texttt{holdsAt}(\texttt{fluent}_1(X,\, Y),\, T{+}1) \Leftarrow & \\ \quad \neg\texttt{holdsAt}(\texttt{fluent}_1(X,\, Y),\, T) \wedge & \\ \quad \neg\big(\,(\,\texttt{happens}(\texttt{event}_\texttt{i}(X),\, T) \wedge \ldots \wedge \texttt{Conditions}[X,Y,T]\,) \bigvee \ldots\big) & (3.25)\\ \ldots & \end{cases}$$

The rules in (3.22)–(3.25) can be separated into two subsets. The former set Σ contains specialised definitions of axioms (3.1) and (3.3), specifying that a fluent holds (or does not hold) when its initiation (or termination) conditions are met. The latter set Σ' contains specialised definitions of the inertia axioms (3.2) and (3.4), specifying whether a specific fluent continues to hold or not at any instance of time.

The presented knowledge transformation procedure reduces the size of the produced network, based only on the rules of the knowledge base. It is automated and taken place as a preprocessing step, preceding the standard grounding procedure of MLNs.

3.4 The Behaviour of the MLN–EC

Recall that the compact knowledge base of MLN–EC is separated into the Σ and Σ' subsets (see Section 3.3.2). The set Σ is composed of the formulas that specify when a fluent holds (or does not hold), while Σ' comprises the formulas that specify when a fluent continues to hold or not. As mentioned in Section 2.3.1, by associating weights to such formulas we define soft constraints, allowing some worlds that do not satisfy these formulas to become likely. For example, consider a knowledge base of Event Calculus axioms and CE definitions (e.g. `meeting` and `moving`) compiled in the form of rules (3.22)–(3.25). Given a narrative of SDEs, the probability of a CE to hold at a specific time-point is determined by the probabilities of the worlds in which this CE holds. Each world, in turn, has some probability which is proportional to the sum of the weights of the ground clauses that it satisfies. Consequently, the probability of a CE to hold at a specific instance of time depends on the corresponding constraints of the ground Markov network. Thus, by treating the rules in the Σ and Σ' sets as either hard or soft constraints, we can modify the behaviour of the Event Calculus. Compared to crisp implementations of the Event Calculus, this flexibility of changing the behaviour of the formalism is an important advantage of the proposed method.

3.4.1 Soft-constrained rules in Σ

In order to illustrate how the probability of a CE is affected when its initiation or termination conditions are met, consider the case that the rules in Σ are soft-constrained while the inertia rules in Σ' remain hard-constrained. By soft-constraining the rules in Σ, the worlds violating their clauses become probable. This situation reduces the certainty with which a CE is recognised when its initiation or termination conditions are met. For example, assume that the initiation rules (3.20) of the `meeting` CE are associated with weights. As a result, the `meeting` activity is initiated with some certainty, causing the CE to hold with some probability. Depending on the strength of the weights, the worlds that violate these rules become more or less likely. Thus, we can control the level of certainty with which a CE holds or not under the same conditions.

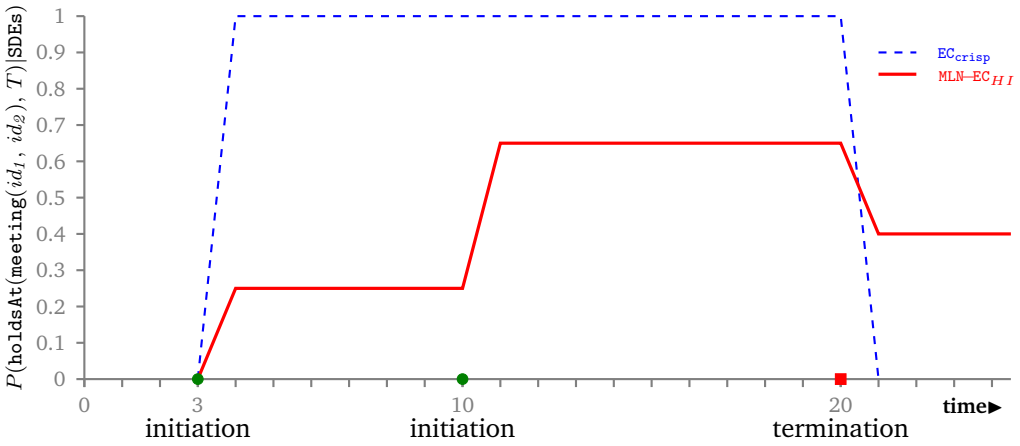


FIGURE 3.3: The probability of the `meeting` CE given some SDE narrative. $\texttt{EC}_{\texttt{crisp}}$ is a crisp Event Calculus. $\texttt{MLN–EC}_{\texttt{HI}}$ is a probabilistic Event Calculus where rules in Σ are soft-constrained, while the inertia rules in Σ' remain hard-constrained.

When the initiation conditions are met, the probability of the CE to hold increases. Equivalently, when the termination conditions are satisfied, the probability of the CE decreases. At the same time, all worlds that do not fully satisfy hard-constrained inertia rules in Σ' are rejected, thus the inertia is retained deterministically as in crisp logic.

Figure 3.3 illustrates this behaviour with the fluent `meeting` that initially does not hold at time 0. According to the narrative of SDEs, the `meeting` activity is initiated at time-points 3 and 10, e.g., satisfying the constraints imposed by rules (3.7) and (3.8) respectively. At time 20, the `meeting` activity is terminated by the conditions of rule (3.9). In crisp Event Calculus, denoted as $\texttt{EC}_{\texttt{crisp}}$, after its first initiation the `meeting` activity holds with absolute certainty. The second initiation at time 10 does not cause any change and the CE continues to hold. The termination at time 20 causes the CE to not hold, again with absolute certainty, for the remaining time-points. In $\texttt{MLN–EC}_{\texttt{HI}}$ (hard-constrained inertia rules), however, the rules in Σ are soft-constrained. As a result, at time-point 4 the probability of `meeting` to hold increases to some value. Similar to $\texttt{EC}_{\texttt{crisp}}$, the inertia is fully retained and the probability of `meeting` deterministically persists in the interval 4 to 10. In contrast to $\texttt{EC}_{\texttt{crisp}}$, the second initiation at time-point 10 increases the certainty of `meeting` to hold. As a result, the probability of `meeting` is higher in the interval 11 to 20. In the same manner, the termination at 20 reduces the probability of `meeting` and the CE continues to hold with some reduced probability.

3.4.2 Soft-constrained inertia rules in Σ'

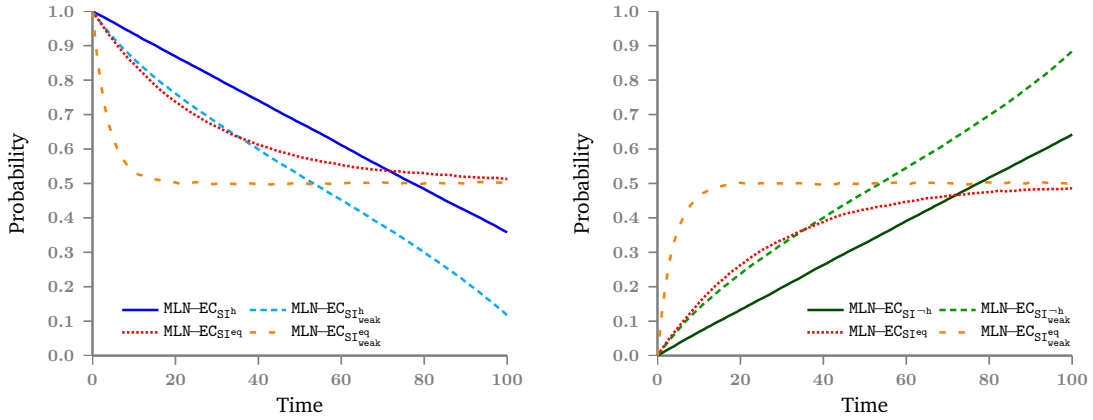
To illustrate how the behaviour of inertia is affected by soft-constraining the corresponding rules in Σ', consider that the rules in Σ are hard-constrained. Consequently, when the initiation (or termination) conditions are met, a CE holds (or does not hold) with absolute certainty. The persistence of a CE depends on its inertia rules in Σ'. If the inertia

of holdsAt is hard-constrained, the worlds in which an initiated CE does not hold are rejected. Similarly, by keeping the inertia of ¬holdsAt hard-constrained, all worlds in which a terminated CE holds are rejected. By soft-constraining these rules we control the strength of the inertia constraints. Even when the inertia constraints are partially violated, the probability of the CE is affected. Thus, the persistence of holdsAt and ¬holdsAt is gradually lost over successive time-points. When allowing the constraints of holdsAt inertia to be violated, the probability of a CE gradually drops. Inversely, by allowing the constraints representing the inertia of ¬holdsAt to be violated, the probability of a CE gradually increases. The lower the value of the weight on the constraint, the more probable the worlds that violate the constraints become. In other words, weight values in Σ' cause CE to persist for longer or shorter time periods.

Since the sum of the probabilities of holdsAt and ¬holdsAt for a specific CE is always equal to 1, the relative strength of holdsAt and ¬holdsAt rules in Σ' determines the type of inertia in the model. The following two general cases can be distinguished.

Equally strong inertia constraints All rules in Σ' are equally soft-constrained, i.e., they are associated with the same weight value. Consequently, both inertia rules of holdsAt and ¬holdsAt for a particular CE impose constraints of equal importance, allowing worlds that violate them to become likely. As a result, in the absence of useful evidence, the probability of holdsAt will converge to the value 0.5. For example, Figure 3.4(a) illustrates soft persistence for the meeting CE when it holds with absolute certainty at time-point 0, and thereafter nothing happens to initiate or terminate it. The curve MLN–EC$_{\mathtt{SI}^{\mathtt{eq}}}$ (soft-constrained inertia rules with equal weights) shows the behaviour of inertia in this case. As time evolves, the probability of meeting appears to gradually drop, converging to 0.5. If we assign weaker weights to the inertia axioms, shown by the MLN–EC$_{\mathtt{SI}^{\mathtt{eq}}_{\mathtt{weak}}}$ curve, the probability of meeting drops more sharply. Similarly, in Figure 3.4(b), the meeting CE is assumed to not hold initially. As time evolves, the probability of meeting gradually increases up to the value 0.5, as shown by the MLN–EC$_{\mathtt{SI}^{\mathtt{eq}}}$ and MLN–EC$_{\mathtt{SI}^{\mathtt{eq}}_{\mathtt{weak}}}$ curves respectively.

Inertia constraints of different strength When the inertia rules of holdsAt and ¬holdsAt for a particular CE in Σ' have different weights, the probability of the CE will no longer converge to 0.5. Since the weights impose constraints with different confidence, worlds violating the stronger constraints become less likely than worlds violating the weaker ones. Depending on the relative strength of the weights, the probability of the CE may converge either to 1.0 or 0.0. The relative strength of the weights affects also the rate at which the probability of CE changes. As an extreme example, in Figure 3.4(a), the rules for the inertia of ¬holdsAt remain hard-constrained. By assigning weights to the rules for the inertia of holdsAt, the persistence of the CE is lost. Since the inertia constraints of holdsAt are weaker than the constraints of ¬holdsAt, worlds violating the



(a) The CE initially holds.

(b) The CE initially does not hold.

FIGURE 3.4: In both figures SDEs occur that lead to the partial satisfaction of the initiation/termination conditions of a CE in the interval 0 to 100. In the left figure the CE holds at time 0 with absolute certainty, while in the right figure the CE does not hold at time 0.

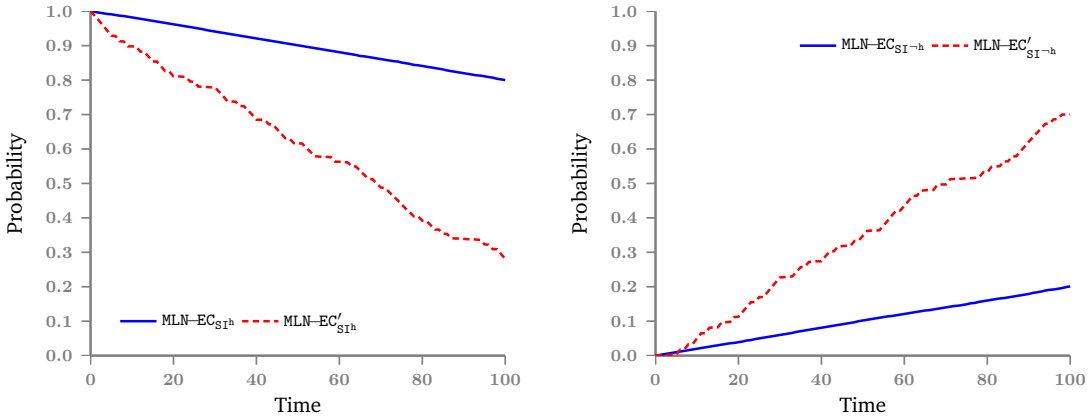
former set of constraints will always be more likely. As a result, the probability of the CE will continue to drop, even below 0.5. The curves MLN–EC$_{SI^h}$ (soft-constrained inertia of `holdsAt`) and MLN–EC$_{SI^h_{weak}}$ (weaker `holdsAt` inertia constraints) illustrate how the probability of `meeting` drops sharply towards 0.0. The weaker the constraints (MLN–EC$_{SI^h_{weak}}$) the steeper the drop. In a similar manner, when the inertia constraints of $\neg$`holdsAt` are weaker than the constraints of `holdsAt`, the probability of CE gradually increases and may reach values above 0.5 — presented by the MLN–EC$_{SI^{\neg h}}$ (soft-constrained inertia of $\neg$`holdsAt`) and MLN–EC$_{SI^{\neg h}_{weak}}$ (weaker $\neg$`holdsAt` inertia constraints) cases in Figure 3.4(b).

As explained in Section 3.3.2, the inertia rule of a CE may consist of a large body of conditions, e.g., rule (3.21). Depending on the number of conditions involved, the rule may be decomposed into several clauses, each corresponding to a different subset of conditions. For instance, the following two clauses are added to Σ' by the inertia rule (3.21):

$$\texttt{happens}(\texttt{walking}(ID_1), T) \vee \texttt{happens}(\texttt{running}(ID_1), T) \vee \texttt{happens}(\texttt{exit}(ID_1), T) \vee \neg\texttt{holdsAt}(\texttt{meeting}(ID_1, ID_2), T) \vee \texttt{holdsAt}(\texttt{meeting}(ID_1, ID_2), T{+}1) \tag{3.26}$$

$$\neg\texttt{close}(ID_1, ID_2, 25, T) \vee \texttt{happens}(\texttt{running}(ID_1), T) \vee \texttt{happens}(\texttt{exit}(ID_1), T) \vee \neg\texttt{holdsAt}(\texttt{meeting}(ID_1, ID_2), T) \vee \texttt{holdsAt}(\texttt{meeting}(ID_1, ID_2), T{+}1) \tag{3.27}$$

The above clauses contain literals from the termination rules of the `meeting` CE. Often, some of these clauses become trivially satisfied. For example, at time-point 10 both



(a) The CE initially holds.

(b) The CE initially does not hold.

FIGURE 3.5: In both figures no useful SDEs occur in the interval 0 to 100. In MLN–EC$_{SI^h}$ and MLN–EC$_{SI^{\neg h}}$ none of the inertia clauses in Σ' become trivially satisfied by the SDEs. However, in MLN–EC$'_{SI^h}$ and MLN–EC$'_{SI^{\neg h}}$ some inertia clauses are trivially satisfied by the SDE. In the left figure, the CE holds at time 0 with absolute certainty, while in the right figure the CE does not hold at time 0.

persons ID_1 and ID_2 are *active*, while their distance is above the threshold of 25 pixels, i.e., close$(ID_1, ID_2, 25, 10)$=*False*. Consequently, the grounding of clause (3.27) at time-point 10 is trivially satisfied for all possible worlds. Although the *meeting* CE is not terminated at time-point 10, because clause (3.26) is not satisfied, the satisfaction of clause (3.27) reduces the probability of holdsAt for the CE. This is because the inertia at time-point 10 is now supported only by the satisfaction of the ground clause (3.26). In other words, the difference between the probabilities of worlds that violate the inertia of holdsAt and worlds that do not, is reduced.

To illustrate this phenomenon, consider the example cases in Figure 3.5(a) where only the rules about the inertia of holdsAt are soft-constrained. Both MLN–EC$_{SI^h}$ and MLN–EC$'_{SI^h}$ cases share the same knowledge base. In the MLN–EC$_{SI^h}$ case, the occurrence of SDEs causes none of the inertia clauses to become trivially satisfied. In the MLN–EC$'_{SI^h}$ case, however, the SDEs are randomly generated and cause a different subset of inertia clauses to become trivially satisfied at each time-point. In both cases the probability of the CE is reduced. In contrast to MLN–EC$_{SI^h}$, however, the inertia in MLN–EC$'_{SI^h}$ drops more sharply, as some of the clauses in Σ' are trivially satisfied by the given SDEs. Additionally, the probability of the CE to hold in MLN–EC$'_{SI^h}$ persists at a different level in each time-point, since different subsets of clauses become trivially satisfied each time. Similarly, in Figure 3.5(b) the rules about the inertia of ¬holdsAt are soft-constrained. In contrast to MLN–EC$_{SI^{\neg h}}$, the occurrence of SDEs leads to the partial satisfaction of the initiation conditions, causing the inertia in MLN–EC$'_{SI^{\neg h}}$ to persist with a different confidence at each time-point, thus increasing the probability of the CE to hold more sharply.

Having analysed the effect of softening the inertia rules, it is worth noting that in many

real cases the entire knowledge base may be soft-constrained. In this case, since the rules in Σ are soft-constrained, CEs are not initiated or terminated with absolute certainty. At the same time, CEs do not persist with certainty, as the rules in Σ' are also soft-constrained.

Depending on the requirements of the target application, various policies regarding the soft-constraining of the knowledge base may be adopted. This flexibility is one of the advantages of combining logic with probabilities in the proposed method. Furthermore, it should be stressed that in a typical event recognition application the knowledge base will contain a large number of clauses. The strength of a constraint imposed by a clause is also affected by the weights of other clauses with which it shares the same predicates. Due to these interdependencies, the manual setting of weights is bound to be suboptimal and cumbersome. Fortunately, the weights can be estimated automatically from training data, using standard parameter optimisation methods.

3.5 Conclusions

We addressed the issue of imperfect CE definitions that stems from the uncertainty that naturally exists in event recognition. MLN–EC is the first probabilistic version of the Event Calculus based on Markov Logic Networks. The method has declarative and formal (probabilistic) semantics, inheriting the properties of the Event Calculus. We placed particular emphasis on the efficiency and effectiveness of our approach. By simplifying the axioms of the Event Calculus and following a knowledge transformation procedure, the method produces compact Markov networks of reduced complexity. Consequently, the performance of probabilistic inference is improved, as it takes place on a simpler model. In contrast to the standard Event Calculus, MLN–EC supports flexible CE persistence, ranging from deterministic to probabilistic. As a result, the behaviour of the persistence can be adjusted, in order to meet the requirements of different applications.

4 Experimental Evaluation of MLN–EC

"It doesn't matter how beautiful your theory is,
it doesn't matter how smart you are.
If it doesn't agree with experiment,
it's wrong."
— Richard P. Feynman

In this chapter we evaluate our MLN–EC method in the domain of video activity recognition. As presented in Section 2.1, we use the publicly available benchmark dataset of the CAVIAR project. The aim of the experiments is to assess the effectiveness of MLN–EC in recognising CEs that occur among people, based on imperfect CE definitions and in the presence of incomplete narratives of SDEs.

MLN–EC combines the benefits of logic-based representation (e.g., direct expression of domain background knowledge) with probabilistic modelling (i.e., uncertainty handling). For comparison purposes, we include in the experiments two approaches that are closely related to our method. First, we use the logic-based activity recognition method of Artikis et al. [2010b], which we call here $\texttt{EC}_{\texttt{crisp}}$. Like our method, $\texttt{EC}_{\texttt{crisp}}$ employs a variant of the Event Calculus and uses the same definitions of CEs (see Section 3.2). Unlike MLN–EC, $\texttt{EC}_{\texttt{crisp}}$ cannot perform probabilistic reasoning. Second, we use a pure probabilistic method that employs a linear-chain Conditional Random Field model [Lafferty et al., 2001], which we call here l–CRF. Similar to our method, l–CRF is a log-linear model that performs probabilistic reasoning over an undirected probabilistic network. On the other hand, l–CRF does not employ a logic-based representation of the domain knowledge.

4.1 Setup

From the 28 videos of the CAVIAR dataset, we have extracted 19 sequences that are annotated with the `meeting` and/or `moving` CEs. The rest of the sequences in the dataset are ignored, as they do not contain examples of the two target CEs. Out of 19 sequences, 8

are annotated with both moving and meeting activities, 9 are annotated only with moving and 2 only with meeting. The total length of the extracted sequences is 12869 frames. Each frame is annotated with the occurrence or not of a CE and is considered an example instance. The whole dataset contains a total of 25738 annotated example instances. There are 6272 example instances in which moving occurs and 3622 in which meeting occurs. Consequently, for both CEs the number of negative examples is significantly larger than the number of positive examples, 19466 for moving and 22116 for meeting.

The input of all three methods consists of a sequence of SDEs, i.e., *active, inactive, walking, running, enter* and *exit*. In both MLN–EC and l–CRF, the spatial constraints close and orientationMove are precomputed and their truth value is provided as input. In situations where no event occurs or the distance of the involved persons is above the highest predefined threshold, the tags *none* and *far* are given to l–CRF, respectively.

The output of the EC_{crisp} method consists of a sequence of ground holdsAt predicates, indicating which CEs are recognised. Since EC_{crisp} performs crisp reasoning, all CEs are recognised with absolute certainty. On the other hand, the output of the probabilistic methods depends on the inference type, i.e., maximum a-posteriori (MAP) or marginal inference. Given a sequence of SDEs, MAP inference outputs the most probable instantiations of CEs for all time-points. On the other hand, marginal inference estimates the probability of each CE of each time-point.

Table 4.1 presents the structure of the training sequences for the probabilistic methods. In particular, Table 4.1(a) shows an example training sequence for MLN–EC. Each sequence is composed of input SDEs (ground happens), precomputed spatial constraints between pairs of people (ground close and orientationMove), as well as the corresponding CE annotations (ground holdsAt). Negated predicates in the training sequence state that the truth value of the corresponding predicate is *False*. Table 4.1(b) shows the equivalent input training data for l–CRF. The random variables *Person A* and *Person B* represent the events that the two persons may be involved in at each time-point and variables *Close* and *Orientation Move* represent the spatial constraints between the two persons. Similar to MLN–EC, l–CRF does not contain any hidden variables between input SDEs and output CEs. As a result, training is fully supervised for both methods.

To estimate the weights in l–CRF, we use the quasi-Newton optimisation algorithm L-BFGS [Byrd et al., 1994]. For MAP and marginal inference we use the *Viterbi* and *Forward-Backward* algorithms [Culotta and McCallum, 2004; Sutton and McCallum, 2007]. In MLN–EC we use the second-order Diagonal Newton method of Lowd and Domingos [2007] and perform marginal inference with the *MC-SAT* algorithm [Poon and Domingos, 2006], taking 1000 samples for each sequence. We additionally perform max-margin training and MAP inference, using the method of Huynh and Mooney [2009]. Details about learning and inference algorithms that we use in MLN–EC are outlined in Section 2.3.1. In

Simple Derived Events	**Composite Events**
...	...
happens(walking(id_1), 100)	
happens(walking(id_2), 100)	holdsAt(moving(id_1, id_2), 100)
orientationMove(id_1, id_2, 100)	holdsAt(moving(id_2, id_1), 100)
close(id_1, id_2, 24, 100)	
...	...
happens(active(id_1), 101)	
happens(walking(id_2), 101)	holdsAt(moving(id_1, id_2), 101)
orientationMove(id_1, id_2, 101)	holdsAt(moving(id_2, id_1), 101)
¬close(id_1, id_2, 24, 101)	
...	...
happens(walking(id_1), 200)	
happens(running(id_2), 200)	¬holdsAt(moving(id_1, id_2), 200)
¬orientationMove(id_1, id_2, 200)	¬holdsAt(moving(id_2, id_1), 200)
¬close(id_1, id_2, 24, 200)	
...	...

(a) Input narrative for MLN–EC.

Person A	**Person B**	**Close**	**Orientation Move**	**Composite Events**
...	...	...	...	...
walking	walking	24	True	Moving
active	walking	Far	True	NotMoving
...	...	...	...	...
walking	running	Far	False	NotMoving
...	...	...	...	...

(b) Input sequence for l–CRF.

TABLE 4.1: Example training sets for CE moving. Table 4.1(a) shows a training set for MLN–EC. The first column is composed of a narrative of SDEs and precomputed spatial constraints for MLN–EC, while the second column contains the CE annotation in the form of ground holdsAt predicates. Table 4.1(b) shows the equivalent training set for l–CRF. Columns *Person A* to *Orientation Move* contain the input SDEs and spatial constraints, while the last column contains the annotation.

the experiments we use the open-source software packages ALCHEMY [Kok et al., 2005] and CRF++[1].

MLN–EC is tested under three different scenarios ($\texttt{MLN–EC}_{\texttt{HI}}$, $\texttt{MLN–EC}_{\texttt{SI}^\texttt{h}}$ and $\texttt{MLN–EC}_{\texttt{SI}}$, see Table 4.2 for a description). In all three variants of MLN–EC, the rules in Σ are soft-constrained while the inertia rules in Σ' are either soft or hard.

Throughout the experimental analysis, the results for marginal inference are presented in terms of F_1 score for threshold values ranging between 0.0 and 1.0. Any CE with probability above the threshold is considered to be recognised. A snapshot of the performance using the threshold value 0.5, is presented in terms of True Positives (TP), False Positives (FP), False Negatives (FN), Precision, Recall and F_1 score. Additionally, the overall performance for marginal inference is measured in terms of area under precision-recall curve (AUPRC). Similar to F_1 score, precision and recall, the AUPRC is insensitive to the number of true negatives, which is much higher that the true positives in the dataset. The evaluation results using MAP inference are presented in terms of True Positives (TP), False Positives (FP), False Negatives (FN), Precision, Recall and F_1 score. All reported experiment statistics are micro-averaged over the instances of recognised CEs in 10 folds.

Scenarios	**Description**
$\texttt{MLN–EC}_{\texttt{HI}}$	All inertia rules in Σ' are hard-constrained.
$\texttt{MLN–EC}_{\texttt{SI}^\texttt{h}}$	The inertia rules of `holdsAt` are soft-constrained, while the rest of Σ' remains hard-constrained.
$\texttt{MLN–EC}_{\texttt{SI}}$	All inertia rules in Σ' are soft-constrained.

TABLE 4.2: Variants of MLN–EC, using hard and soft inertia rules in Σ'.

4.1.1 The Methods Being Compared

Both $\texttt{EC}_{\texttt{crisp}}$ and MLN–EC employ a logic-based representation, implement a variant of the Event Calculus and contain equivalent definitions of CEs. The CE definitions of `meeting` and `moving` of $\texttt{EC}_{\texttt{crisp}}$ are translated into first-order logic for MLN–EC using the formulation proposed in Section 3.1. The definition of `meeting` is given by formulas (3.7)–(3.11), while that of `moving` is given by formulas (3.12)–(3.17). In contrast to $\texttt{EC}_{\texttt{crisp}}$, each clause in MLN–EC may be associated with a weight value, indicating a degree of confidence.

Similar to MLN–EC, l–CRF is a discriminative probabilistic graphical model. In l–CRF, the relationship among CEs at successive time-points is modelled as a Markov network, conditioned on the input evidence of SDEs. A CE is represented by a Boolean random variable, stating whether the CE holds or not at any time-point in the sequence. For

[1] http://code.google.com/p/crfpp

example, the random variables representing the moving CE may take either the value *Moving* or *NotMoving* at some time-point in the sequence.

Despite their similarities, there are also several differences between the two probabilistic methods. In l–CRF, the input SDEs and the spatial constraints are represented by multivariate random variables. For instance, the input SDEs for a particular person are represented by a single random variable that can take any SDE value, e.g., *active, inactive, walking*, etc. The relationship among random variables is defined by two types of feature. The former type associates input SDEs and spatial constraints with output CEs at the same time-point, creating features for all possible instantiations. The latter type associates successive CEs, in order to form linear chains. In particular, features are constructed for each possible pair of CE instantiations at successive time-points. All features in l–CRF are associated with weights and thus all relationships are soft-constrained.

On the other hand, MLN–EC employs a logic-based representation and all features are produced from ground clauses. Domain knowledge is combined with the Event Calculus axioms, in order to form the structure of the network. For example, the relations between successive CE instantiations are formed by the inertia axioms and the corresponding initiation and termination rules. Moreover, the MLN–EC provides control over the behaviour of CE persistence, by allowing the inertia clauses to be defined as either soft or hard constraints. The probabilistic inference in MLN–EC can deal with both deterministic (i.e., hard-constrained clauses) and probabilistic (i.e., soft-constrained clauses) dependencies, as well as arbitrarily structured networks. On the other hand, the structural simplicity of l–CRF allows for specialised and significantly faster inference and learning methods.

4.2 Network Size and Inference Times

The CE definitions of moving (rules (3.12)–(3.17)) and meeting (rules (3.7)–(3.11)) together with the domain-independent axioms of MLN–EC (rules (3.1)–(3.4)) are preprocessed using the transformation procedure presented in Section 3.3.2. Given the SDEs of all 19 sequences, the knowledge bases of moving and meeting are transformed into Markov networks of 128689 ground clauses and 51548 ground predicates.

Our experiments were performed on a machine with an Intel® Core™ i7-980 3.33 GHz processor and 24GB of RAM, running Debian GNU/Linux 7.0. Table 4.3 shows the inference times of all MLN–EC scenarios: we show the inference times for the 12869 video frames of all 19 sequences (this corresponds to a video of 8 minutes and 58 seconds), as well as, the average inference time and standard deviation for a single frame.

Our knowledge base transformation procedure reduces significantly the size and complexity of the produced ground Markov networks. Without this transformation procedure

	MARGINAL		MAP	
Scenarios	**All sequences**	**Single frame**	**All sequences**	**Single frame**
MLN–EC$_{\text{HI}}$	1 m 16 s	6.67 $\pm$ 2.42 ms	1 m 38 s	7.11 $\pm$ 1.47 ms
MLN–EC$_{\text{SI}^h}$	1 m 17 s	6.77 $\pm$ 2.32 ms	1 m 41 s	7.32 $\pm$ 1.40 ms
MLN–EC$_{\text{SI}}$	1 m 40 s	7.96 $\pm$ 3.07 ms	1 m 42 s	7.44 $\pm$ 1.47 ms

(a) `moving`

	MARGINAL		MAP	
Scenarios	**All sequences**	**Single frame**	**All sequences**	**Single frame**
MLN–EC$_{\text{HI}}$	1 m 21 s	6.32 $\pm$ 1.43 ms	0 m 19 s	1.40 $\pm$ 0.29 ms
MLN–EC$_{\text{SI}^h}$	1 m 15 s	6.06 $\pm$ 1.17 ms	0 m 19 s	1.41 $\pm$ 0.28 ms
MLN–EC$_{\text{SI}}$	1 m 30 s	7.16 $\pm$ 1.59 ms	0 m 20 s	1.48 $\pm$ 0.22 ms

(b) `meeting`

TABLE 4.3: Probabilistic inference running times of MLN–EC.

and for the same Event Calculus dialect, the resulting Markov networks for the same 19 SDE sequences are three times larger, consisting of 334452 ground clauses and 154428 ground predicates. As an example, in the case of `meeting` and the MLN–EC$_{\text{HI}}$ scenario, the time for marginal inference is 4 minutes and 33 seconds — this is more than 3 times slower than the inference over the transformed knowledge base (see MLN–EC$_{\text{HI}}$ in Table 4.3(b)).

4.3 Experimental Results of the EC$_{\text{crisp}}$

The CE definitions are domain-dependent rules that together with the domain-independent axioms of the Event Calculus represent common-sense domain knowledge. This knowledge may deviate from knowledge of human captured in an annotated dataset, resulting in errors when recognising events. Such issues can be shown by analysing the performance of EC$_{\text{crisp}}$, which uses the CE definitions also used in MLN–EC and does not involve the representation of probabilistic knowledge.

As shown in Figure 4.1, EC$_{\text{crisp}}$ achieves a similar F_1 score for both activities. However, in terms of precision and recall the situation is quite different, revealing two different cases of imperfect CE definitions. The precision for `moving` is 22 percentage points higher than that of `meeting`. The opposite holds for recall, with the recall for `moving` being 21.6 percentage points lower than that of `meeting`. The lower recall values for `moving` indicate a larger number of unrecognised `moving` activities (FN). In some example sequences `moving` is being initiated late, producing many false negatives. Additionally, the termination rules of `moving` cause the CE to be prematurely terminated in some cases. For example, when the distance of two persons that are moving together becomes greater than 34 pixels for a few frames, rule (3.13) terminates `moving`. On the other hand,

compared to moving, the definition of meeting results in a larger number of erroneously recognised meeting activities (FP). The initiation rule (3.8), for example, causes the meeting activity to be initiated earlier than it should.

Another issue caused by the definitions of meeting and moving is that the two CEs may overlap. According to rules (3.7)–(3.17), the initiation of moving does not cause the termination of meeting. Consider, for example, a situation where two people meet for a while and thereafter they move together. During the interval in which moving is detected, meeting will also remain detected, as it is not terminated and the law of inertia holds. However, according to the annotation of the CAVIAR data set these activities do not happen concurrently. Furthermore, meeting appears to be annotated in cases where the relative distance of both interacting persons is greater than that used in the initiation rules (3.7) and (3.8).

On the other hand, the domain knowledge may describe situations that are not included in the dataset. For example, by allowing the meeting CE to be initiated when the two persons are not very close to each other, one may achieve better results in this subset of the dataset, but erroneously recognise the meeting CE in other situations, e.g., people passing by each other. Additionally, the domain-independent property of inertia, which is included in the background knowledge, helps the method to continue to recognise the occurrence of a CE even when the narrative of SDEs is temporally incomplete, e.g., due to camera failures.

4.4 Experimental Results of the Probabilistic Methods

The experiments for the probabilistic methods are organised into two tasks[2]. In the first task, both probabilistic methods are trained discriminitavely and their performance is evaluated using 10-fold cross-validation. In the second task, we asses the value of inertia by erasing input evidence from randomly chosen successive time-points. In the second task, we use the models trained for the first task, but the testing sequences are incomplete narratives of SDEs.

4.4.1 Task I

In contrast to EC_{crisp}, l–CRF is a probabilistic model and cannot employ background knowledge in the form of common sense rules. The recognition of a CE is affected from all input SDEs that are detected at a particular time-point, as well as from the adjacent CEs that have been recognised. The parameters of the model are estimated from the

[2]MLN and l–CRF formatted version of the dataset, CE definitions of MLN–EC, template files of l–CRF and the results of both probabilistic methods can be found at: http://www.iit.demokritos.gr/~anskarl/pub/mlnec

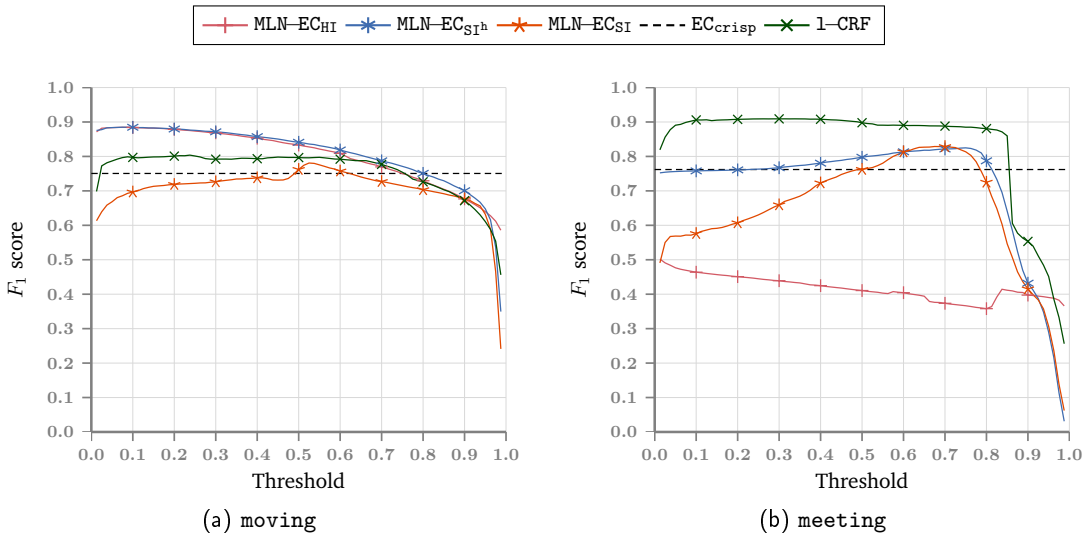


FIGURE 4.1: F_1 scores using different threshold values for the moving and meeting CE.

training set and thus the model is completely data-driven. Compared to EC$_{crisp}$, 1–CRF achieves better performance for both moving and meeting. Using marginal inference, 1–CRF gives higher F_1 scores for most threshold values, as shown in Figures 4.1(a) and 4.1(b). For threshold 0.5, the F_1 score is higher than of EC$_{crisp}$ by 4.6 and 13.5 percentage points for moving and meeting, respectively (see Tables 4.4(a) and 4.4(c)). The recall of moving is higher by 18.4 percentage points, while the precision is lower by 13.7 percentage points. 1–CRF recognises a larger number of moving activities, increasing the number of both true and false positives. As noted in Section 4.3, this can be achieved by looser spatial constraints. Recall and precision scores for meeting are higher by 1.7 and 23.7 percentage points, respectively. In the case of MAP inference, 1–CRF gives almost the same F_1 score as EC$_{crisp}$ for moving (see Tables 4.5(a) and 4.5(b)). In general 1–CRF outperforms EC$_{crisp}$ in both CE. Unlike EC$_{crisp}$, 1–CRF learns to interrupt the recognition of meeting when moving starts.

In the first MLN–EC scenario, indicated by MLN–EC$_{HI}$, rules in Σ are soft-constrained, i.e., they are associated with a weight value after training. Those weights control the certainty with which a CE holds when its initiation or termination conditions are satisfied. The rules in Σ', however, remain hard-constrained and thus the behaviour of inertia for both CEs is preserved deterministically and cannot be adjusted. Compared to EC$_{crisp}$, MLN–EC$_{HI}$ achieves a higher F_1 score using marginal inference for the moving CE, for most threshold values (see Figure 4.1(a)). For threshold 0.5, the recall of MLN–EC is higher by 17.7 percentage points than EC$_{crisp}$ while precision is lower by 6 points, leading the F_1 score of MLN–EC to be higher than EC$_{crisp}$ by 8.2 points (Table 4.4(a)). This improvement in recall is caused by the weights that are learned for the termination conditions, which prevent the moving CE from terminating prematurely. Compared to 1–CRF, MLN–EC$_{HI}$ achieves better $F1$ scores for many thresholds and higher AUPRC by 4.8 percentage points (see Tables 4.4(a) and 4.4(b)). Using MAP inference, MLN–EC$_{HI}$

Method	TP	TN	FP	FN	Precision	Recall	F_1 score	AUPRC
EC_{crisp}	4008	19086	400	2264	**0.9093**	0.6390	0.7506	
MLN–EC_{HI}	5121	18584	902	1151	0.8502	0.8165	0.8330	**0.8847**
MLN–EC_{SI^h}	5233	18542	944	1039	0.8472	**0.8343**	**0.8407**	0.8597
MLN–EC_{SI}	4938	17760	1726	1334	0.7410	0.7873	0.7635	0.8280
l–CRF	5160	17964	1522	1112	0.7722	0.8227	0.7967	0.8358

(a) moving, threshold 0.5. (b) moving

Method	TP	TN	FP	FN	Precision	Recall	F_1 score	AUPRC
EC_{crisp}	3099	20723	1413	523	0.6868	0.8556	0.7620	
MLN–EC_{HI}	1284	20786	1350	2338	0.4875	0.3545	0.4105	0.5160
MLN–EC_{SI^h}	3060	21147	989	562	0.7557	0.8448	0.7978	0.7559
MLN–EC_{SI}	3052	20800	1336	570	0.6955	0.8426	0.7620	0.7730
l–CRF	3160	21874	262	462	**0.9234**	**0.8724**	**0.8972**	**0.8937**

(c) meeting, threshold 0.5. (d) meeting

TABLE 4.4: Results for the moving and meeting CE using marginal inference.

Method	TP	TN	FP	FN	Precision	Recall	F_1 score
EC_{crisp}	4008	19086	400	2264	**0.9093**	0.6390	0.7506
MLN–EC_{HI}	5598	18358	1128	674	0.8323	0.8925	0.8614
MLN–EC_{SI^h}	5902	18398	1088	370	0.8443	**0.9410**	**0.8901**
MLN–EC_{SI}	4040	17911	1575	2232	0.7195	0.6441	0.6797
l–CRF	4716	17848	1638	1556	0.7422	0.7519	0.7470

(a) moving

Method	TP	TN	FP	FN	Precision	Recall	F_1 score
EC_{crisp}	3099	20723	1413	523	0.6868	0.8556	0.7620
MLN–EC_{HI}	3099	20739	1397	523	0.6893	0.8556	0.7635
MLN–EC_{SI^h}	3067	21825	311	555	0.9079	0.8468	0.8763
MLN–EC_{SI}	1083	21641	495	2539	0.6863	0.2990	0.4165
l–CRF	3154	21906	230	468	**0.9320**	**0.8708**	**0.9004**

(b) meeting

TABLE 4.5: Results for the moving and meeting CE using MAP inference

achives higher F_1 score by 11 percentage points than both $\texttt{EC}_{\texttt{crisp}}$ and l–CRF (see Table 4.5(a)). Compared to $\texttt{EC}_{\texttt{crisp}}$, the recall of $\texttt{MLN–EC}_{\texttt{HI}}$ is improved by 25.3 percentage points, while its precision drops by 7 percentage points. $\texttt{MLN–EC}_{\texttt{HI}}$ achieves higher recall and precision scores than l–CRF, by 14 and 9 percentage points, respectively. However, in the case of meeting, $\texttt{MLN–EC}_{\texttt{HI}}$ performs worse than $\texttt{EC}_{\texttt{crisp}}$ and l–CRF in marginal inference, as shown in Figure 4.1(b) and Table 4.4(c). The combination of hard-constrained inertia rules with the fact that meeting does not terminate when moving starts, push the weights of the initiation rules to very low values during training. This situation results in many unrecognised meeting instances and low precision and recall values. Max-margin training in this scenario is not affected as much as the Diagonal Newton weight learning method, leading to a model with similar behaviour and performance as $\texttt{EC}_{\texttt{crisp}}$, as shown in Table 4.5(b).

In the $\texttt{MLN–EC}_{\texttt{SI}^\texttt{h}}$ scenario, while Σ remains soft-constrained, the inertia rules of holdsAt in Σ' are also soft-constrained. As a result, the probability of a CE tends to decrease, even when the required termination conditions are not met and nothing relevant is happening. This scenario is more suitable to our target activity recognition task and MLN–EC learns a model with a high F_1 score for both CEs. In order to explain the effect of soft constraining the inertia of holdsAt, we will use again the example of meeting being recognised and thereafter moving being also recognised. Since meeting is not terminated, it continues to hold and overlaps with moving. During the overlap, all occurring SDE are irrelevant with respect to meeting and cannot cause any initiation or termination. As a result, the recognition probability of meeting cannot be reinforced, by re-initiation. As shown in Section 3.4, in such circumstances the recognition probability of meeting gradually decreases.

For the moving CE, the performance of $\texttt{MLN–EC}_{\texttt{SI}^\texttt{h}}$ using marginal inference is similar to $\texttt{MLN–EC}_{\texttt{HI}}$, as shown in Figure 4.1(a) and Table 4.4(a). Using a threshold of 0.5, recall is higher than that of $\texttt{EC}_{\texttt{crisp}}$ by 19.5 percentage points while precision is lower by 6.2 points, resulting in 9 points increase in F_1 measure. Compared to l–CRF, $\texttt{MLN–EC}_{\texttt{SI}^\texttt{h}}$ achieves higher F_1 scores for many thresholds (see Figure 4.1(a)), as well as higher AUPRC by 2.4 percentage points (see Table 4.4(b)). In the case of MAP inference, $\texttt{MLN–EC}_{\texttt{SI}^\texttt{h}}$ increases further its performance (Table 4.5(a)). Compared to $\texttt{EC}_{\texttt{crisp}}$, recall is higher by 30 percentage points and precision drops only by 6.5 percentage points, resulting in 14 percentage points higher F_1 score. $\texttt{MLN–EC}_{\texttt{SI}^\texttt{h}}$ achieves higher F_1 score, precision and recall than l–CRF by 14.3, 10 and 18.9 percentage points, respectively.

For the meeting CE, the performance of $\texttt{MLN–EC}_{\texttt{SI}^\texttt{h}}$ using marginal inference is significantly better than that of $\texttt{MLN–EC}_{\texttt{HI}}$ (see Figure 4.1(b)). At the 0.5 threshold value, precision increases by 6.9 percentage points over $\texttt{EC}_{\texttt{crisp}}$, while recall falls by only 1 point and thus F_1 is higher by 3.5 points (Table 4.4(c)). However, the F_1 scores of $\texttt{MLN–EC}_{\texttt{SI}^\texttt{h}}$ remain lower than those of l–CRF and its AUPRC is lower by 13.8 points (Table 4.4(d)). Using MAP inference, the performance of MLN–EC improves, but remains worse

than l–CRF, by 2.4 percentage points in terms of F_1 (Table 4.5(b)). MLN–EC$_{\text{SI}^\text{h}}$ misses the recognition of `meeting` at time-points where the persons involved are not sufficiently close to each other according to the initiation rules (3.7) and (3.8), i.e., they have a distance greater than 25 pixels.

Finally, in the MLN–EC$_{\text{SI}}$ scenario, the entire knowledge base is soft-constrained. The weights in Σ allow full control over the confidence that a CE holds when its initiation or termination conditions are met. Additionally, by soft-constraining the rules in Σ', MLN–EC$_{\text{SI}}$ provides full probabilistic inertia. However, this flexibility comes at the cost of an increase in the number of parameters to be estimated from data, as all clauses in the knowledge base are now soft-constrained. As a result, MLN–EC$_{\text{SI}}$ requires more training data. Using marginal inference MLN–EC$_{\text{SI}}$ performs almost the same as EC$_{\text{crisp}}$ in terms of F_1 score, but worse than MLN–EC$_{\text{SI}^\text{h}}$ for both CEs. In the case of MAP inference, MLN–EC$_{\text{SI}}$ performs worse than EC$_{\text{crisp}}$, as well as l–CRF for both CEs.

The three variants of MLN–EC used in the above experiments, illustrated the potential benefits of softening the constraints and performing probabilistic inference in event recognition. In contrast to EC$_{\text{crisp}}$, an important characteristic of MLN–EC is that multiple successive initiations (or terminations) can increase (or decrease) the recognition probability of a CE. By softening the CE definitions, premature initiation or termination can be avoided. In particular, as explained above, the weights learned for the termination definitions of the `moving` CE reduced the number of unrecognised `moving` activities.

The choice of rules to be softened affects significantly the event recognition accuracy. In the presented application, for example, the MLN–EC$_{\text{SI}^\text{h}}$ setting is the best choice, as softening the inertia of `holdsAt` provides advantages over crisp recognition. Depending on the target application and the availability of training data, different types of inertia rules may be softened, varying the inertia behaviour from deterministic to completely probabilistic. This is a key feature of MLN–EC.

4.4.2 Task II

An important difference between the proposed logic-based approach and its purely data-driven competitors, such as l–CRF, is that it is less dependent on the peculiarities of the training data. The use of background knowledge about the task and the domain, in terms of logic, it can make the recognition process more robust to variations in the data. Such variations are very common in practice, particularly in dynamic environments, such as the ones encountered in event recognition. The common assumption made in machine learning that the training and test data share the same statistical properties is often violated in these situations. In order to measure the benefits that we can gain by the combination of background knowledge and learning in MLNs, we examine the recognition of CEs under such situations. In particular, in this second task, we are

comparing the performance of MLN–EC and l–CRF in cases where the input evidence is missing from a number of successive time-points in the test data. The models are not retrained and thus their weights remain as they were estimated in Task I.

The incomplete test sequences are generated artificially by erasing successive input SDEs and associated spatial relations at random starting points. We have generated two variants of incomplete test sequences, containing random blank intervals of length 10 and 20 time-points respectively. The starting time-points of the blank intervals are chosen randomly with probability 0.01, drawn from a uniform distribution. Since the target CEs require the interaction of two persons, erasing SDEs at time-points where only a single person appears to be doing something cannot affect the performance of the recognition methods that we compare. Therefore, blank intervals are created only from time-points where both persons are involved in some SDEs. This process of artificially generating incomplete test sequences is repeated five times, generating corresponding test sets.

The recognition performance can be affected in various ways, depending on the position where evidence is erased from the test sequences. For example, when the beginning of an activity is erased, its recognition will be delayed, increasing the number of false negatives. Similarly, erasing information at the end of an activity will delay the termination of a CE, resulting in a higher number of false positives. In cases where missing information appears during an activity, the recognition of the CE may be interrupted, increasing the number of false negatives. The recognition performance may even be improved in some cases, due to the removal of SDEs that would cause false positives or false negatives.

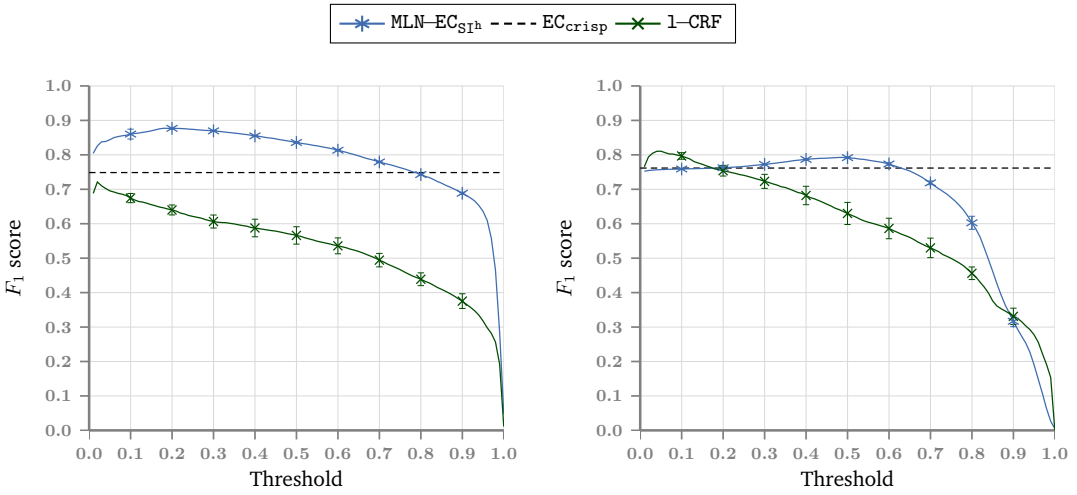
Out of all the methods examined in Task I the performance of EC_{crisp} and $\text{MLN–EC}_{\text{HI}}$ is bound to be affected less by the missing data, due to the use of deterministic inertia. This is because the erased evidence will often be in the interval that a CE is (not) being recognised. In these cases, the erased evidence will not affect the inertia of the CE and the CE will remain (not) recognised. EC_{crisp} and $\text{MLN–EC}_{\text{HI}}$ are only affected when the evidence is erased at the beginning or at the end of an activity, which is less frequent. For this reason, we chose to exclude these two methods from Task II. Furthermore, we excluded $\text{MLN–EC}_{\text{SI}}$, which performed significantly worse than $\text{MLN–EC}_{\text{SI}^{\text{h}}}$ in Task I. Therefore, in this task we compare only $\text{MLN–EC}_{\text{SI}^{\text{h}}}$ against l–CRF.

In the rest of this section we will denote as *medium* and *large incomplete sequences* the test sequences that contain random blank intervals of 10 and 20 time-points, respectively. The evaluation measures are the same as in Task I. Figure 4.2 presents the results in terms of F_1 score for the two methods, using marginal inference. The bar charts of Figure 4.3, on the other hand, present the average AUPRC of the two methods compared also to the AUPRC when no data is removed (from Tables 4.4(b) and 4.4(d) of Task I). The threshold values range between 0.0 and 1.0. Using a similar illustration, Figures 4.4, 4.5 and 4.6 present the results of the two methods using MAP inference. All results are averaged over five runs and error bars correspond to the standard deviation.

Unlike MLN–EC$_{\text{SI}^h}$, l–CRF appears to be affected significantly from incomplete evidence. Using marginal inference on the moving CE in the original test sequences (see Table 4.4(a)) l–CRF achieved an F_1 score of 0.7967. At the same threshold value, the average F_1 score of l–CRF drops to 0.566 and 0.53 for medium and large incomplete sequences, respectively. MLN–EC$_{\text{SI}^h}$ is affected much less, achieving F_1 scores of 0.836 and 0.832, for medium and large incomplete sequences, respectively. In terms of AUPRC (Figure 4.3(a)), the performance of l–CRF also drops by 13 points, while MLN–EC$_{\text{SI}^h}$ remains almost unaffected. When MAP inference is used, the effect of the removal of data seems to be larger. The recall of l–CRF falls by more than 45 percentage points, causing the F_1 score to drop by more than 30 percentage points (Figures 4.5(a) and 4.4(a)). The number of recognised moving activities is reduced, resulting in an increase in precision, but with high variance (Figure 4.5(a)). The precision of MLN–EC$_{\text{SI}^h}$ remains close to the original test set, with a small variance. However, its recall drops and causes the reduction of F_1 score by 8 and 9 percentage points for medium and large incomplete sequences, respectively.

In the case of the meeting CE, MLN–EC$_{\text{SI}^h}$ also seems to resist more than l–CRF to the distortion of the data. The F_1 score is higher than that of l–CRF for many threshold values, using marginal inference (see Figures 4.2(b) and 4.2(d)). For threshold 0.5 in the original test sequences l–CRF achieved an F_1 score that was higher by 10 percentage points than that of MLN–EC$_{\text{SI}^h}$. However, when data are removed its F_1 score for the same threshold drops much lower than that of MLN–EC$_{\text{SI}^h}$ (a difference of more than 15 percentage points). The AUPRC of l–CRF also drops much more (more than 10 points) than that of MLN–EC$_{\text{SI}^h}$ (see Figure 4.3(b)). The effect is even higher when MAP inference is used for the meeting CE. In particular, the recall of l–CRF drops by more than 60 percentage points (see Figure 4.6(b)), while that of MLN–EC$_{\text{SI}^h}$ drops much less. Thus, the F_1 score of MLN–EC$_{\text{SI}^h}$ reduces by less than 10 percentage points, while that of l–CRF is 50 percentage points lower than in the original data (see Figure 4.4(b)).

In summary, this task confirmed that the logic-based MLN–EC method is more robust than its purely statistical l–CRF counterpart, when data are removed from the test set, rendering it less similar to the training set. This is due to the fact that l–CRF is completely data-driven and does not employ explicit domain knowledge. On the other hand, MLN–EC employs background knowledge, including the domain independent axioms of inertia. Consequently, the persistence of a CE is modelled differently by l–CRF and MLN–EC. l–CRF learns to maintain the state of CEs under some circumstances that appear in the training data. However it does not model the inertia explicitly. Therefore, when the circumstances change, its performance is hurt significantly. On the other hand, MLN–EC enjoys the explicit modelling of inertia, provided as background knowledge. Even when this inertia is softened, it remains a strong bias in the model. As a result, MLN–EC avoids overfitting the training data and behaves robustly when the data changes.



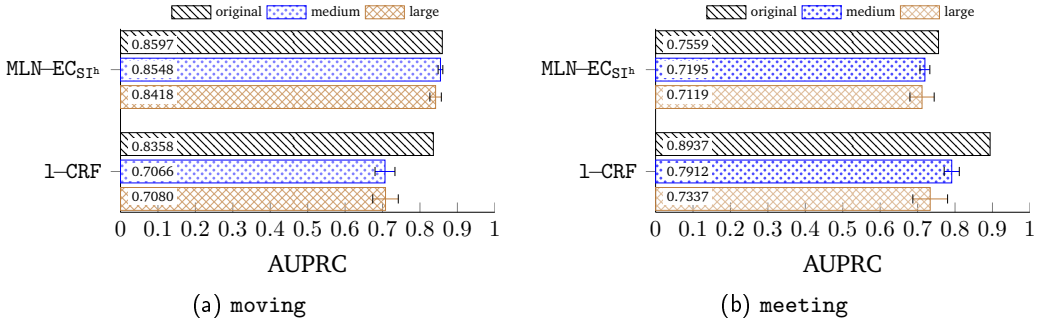
(a) moving – medium incomplete.

(b) meeting – medium incomplete.

(c) moving – large incomplete.

(d) meeting – large incomplete.

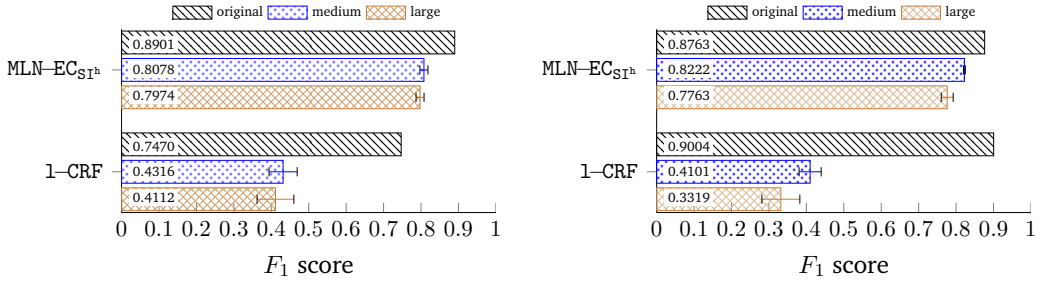
FIGURE 4.2: Average F_1 scores over five runs when data are removed. Marginal inference is used.



(a) moving

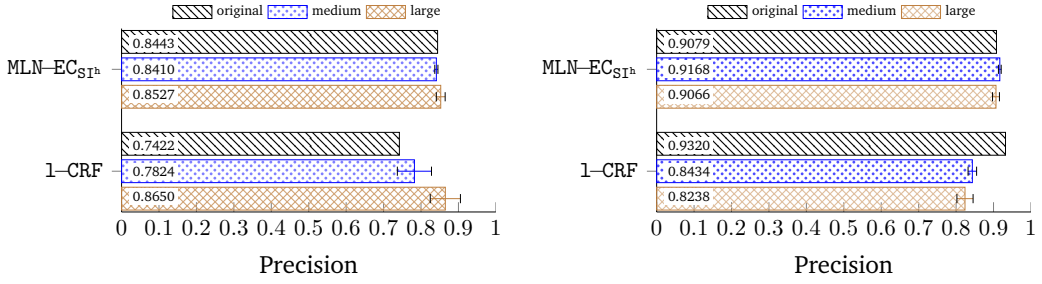
(b) meeting

FIGURE 4.3: Average AUPRC scores over five runs when data are removed. Marginal inference is used.



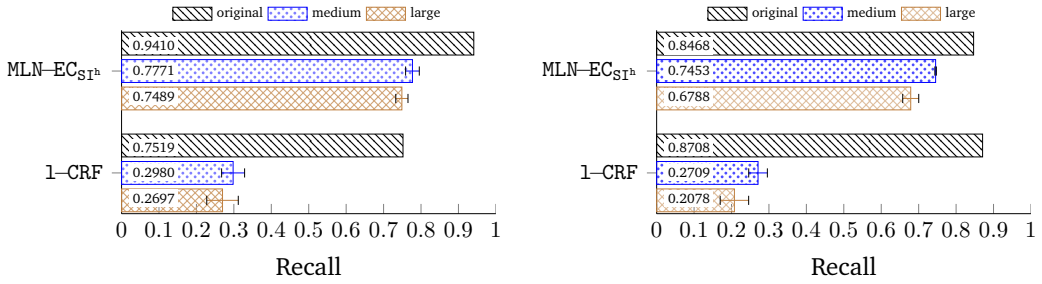
(a) `moving` (b) `meeting`

FIGURE 4.4: Average F_1 scores over five runs when data are removed. MAP inference is used.



(a) `moving` (b) `meeting`

FIGURE 4.5: Average precision over five runs when data are removed. MAP inference is used.



(a) `moving` (b) `meeting`

FIGURE 4.6: Average recall over five runs when data are removed. MAP inference is used.

4.5 Conclusions

Due to the use of MLNs, the method lends itself naturally to learning the weights of event definitions from data, as the manual setting of weights is sub-optimal and cumbersome. MLN–EC is trained discriminatively, using a supervised learning technique. In the experimental evaluation, MLN–EC outperforms its crisp equivalent on a benchmark data, while it matches the performance of a linear-chain Conditional Random Fields method. Furthermore, due to the use of background knowledge in the form of the Event Calculus, MLN–EC is affected less by missing data in the test sequences than its probabilistic akin.

5 | ProbLog–EC: Probabilistic Event Calculus based on Logic Programming

"You don't understand anything until you learn it more than one way."
— Marvin Minsky

The approach to event recognition adopted in this thesis separates low-level from high-level recognition. The lower level provides the input to the high-level recognition, in the form of simple, derived events (SDEs). The higher level combines the input streams of SDEs, in order to recognise CEs of interest. As presented in Section 1.1, uncertainty is an unavoidable aspect of real-world event recognition applications. In particular, in the domain of video activity recognition the lower level is usually processed by computer vision and pattern recognition techniques (e.g., using Hidden Markov Models or Conditional Random Fields) that recognise SDEs of individual activities. For example, such techniques can detect a person walking or staying active. Due to noise, low-level event recognition systems often detect SDEs with some confidence — usually expressed in the form of probabilities.

One commonly used approach in event recognition, is to filter such input SDEs by defining a minimum threshold. Therefore, only the SDEs associated with confidence that exceeds this threshold value form the input to the high-level event recognition system. Although this approach is simple and does not require any changes to the high-level system, it has two main disadvantages. First, it requires to estimate the appropriate threshold value. Second, the information about the recognition confidence of SDEs is not exploited by the high-level system, leading to sub-optimal and often erroneous event recognition.

In this chapter we focus on such erroneous SDE detection and present the ProbLog–EC method that extends the Event Calculus with probabilistic logic programming, in order to handle the uncertainty at the input of the system. ProbLog–EC can directly exploit input SDEs that are associated with probabilities, indicating the degree of confidence.

5.1 Axiomatisation

The Event Calculus dialect presented here is based on probabilistic logic-programming (ProbLog). The time model is linear and discrete (i.e., it includes only integers). For the task of event recognition, we focus only on the domain-independent axioms that determine the influence of events on fluents and the inertia of fluents. Where F is a *fluent* — a property that is allowed to have different values at different points in time — the term $F{=}V$ denotes that fluent F has value V. Boolean fluents are a special case in which the possible values are *true* and *false*. Informally, $F{=}V$ holds at a particular time-point if $F{=}V$ has been *initiated* by an event at some earlier time-point, and not *terminated* by another event in the meantime — *law of inertia*. The axioms and domain-dependent composite event (CE) definitions of the Event Calculus dialect are expressed as definite clauses. Predicates stating the initiation and termination of fluents are defined in terms of fluents and time-points. Table 5.1 summarises the main predicates of the ProbLog–EC. Variables, starting with an upper-case letter, are assumed to be universally quantified unless otherwise indicated. Predicates, function symbols and constants start with a lower-case letter.

Predicate	Meaning
happensAt(E, T)	Event E occurs at time-point T
initially($F{=}V$)	The value of fluent F is V at time 0
holdsAt($F{=}V$, T)	The value of the fluent F is V at time T
initiatedAt($F{=}V$, T)	At time T a period of time for which $F{=}V$ is initiated
terminatedAt($F{=}V$, T)	At time T a period of time for which $F{=}V$ is terminated
clipped($F{=}V$, T)	The value V of fluent F is terminated at time T
negate$_1$($Fact$)	The complement probability of a (probabilistic) $Fact$
negate$_2$($Goal$)	The complement probability of a (probabilistic) $Goal$

TABLE 5.1: Main predicates of the ProbLog–EC.

To express the Event Calculus in ProbLog we had to update our treatment of negation in order to allow for probabilistic atoms. To compute the complement probability of a probabilistic fact, ProbLog provides the built-in predicate problog_not. For any probabilistic fact p_i :: f$_i$, we have that:

$$P_s(\texttt{problog\_not}(f_i)) = 1 - P_s(f_i) = 1 - p_i \tag{5.1}$$

problog_not can only be used on facts that are part of the knowledge base. For facts that do not belong to the knowledge base, such as an SDE that is not part of an input stream, problog_not fails silently and reports a probability of 0. Consequently, translating not to problog_not, where facts are used in our Event Calculus axioms, would lead to

undesirable behaviour. To overcome this issue, we define the following predicate:

$$\texttt{negate}_1(Fact) \leftarrow \texttt{not}\ Fact \tag{5.2}$$

$$\texttt{negate}_1(Fact) \leftarrow \texttt{problog\_not}(Fact) \tag{5.3}$$

With the use of $\texttt{negate}_1$ we produce a probability of 1 whenever the negated SDE is not detected (see rule (5.2)), but also produce the complement of the probability of the SDE whenever it is detected (see rule (5.3)). Note that in the latter case, that is, when an SDE is detected with some probability, then '$\texttt{not}\ Fact$' fails.

Furthermore, the built-in predicate `problog_neg` is an extension of `problog_not` applicable to both probabilistic facts and derived atoms. For a derived atom r, we have:

$$P_s(\texttt{problog\_neg}(r)) = 1 - P_s(r) \tag{5.4}$$

Similarly to `problog_not`, `problog_neg` cannot be used on goals that are not inferable from the knowledge base. To overcome this issue, we define the $\texttt{negate}_2$ predicate:

$$\texttt{negate}_2(Goal) \leftarrow \texttt{not}\ Goal \tag{5.5}$$

$$\texttt{negate}_2(Goal) \leftarrow \texttt{problog\_neg}(Goal) \tag{5.6}$$

With the use of $\texttt{negate}_2$ we produce a probability of 1 whenever the negated goal is not inferable (see rule (5.5)), but also produce the complement of the probability of the goal whenever it is inferable (see rule (5.6)).

The domain-independent rules of the Event Calculus dialect are defined as follows:

$$\begin{aligned}&\texttt{holdsAt}(F{=}V,\,0) \leftarrow \\ &\qquad \texttt{initially}(F{=}V).\end{aligned} \tag{5.7}$$

$$\begin{aligned}&\texttt{holdsAt}(F{=}V,\,T) \leftarrow \\ &\qquad \texttt{initiatedAt}(F{=}V,\,T{-}1).\end{aligned} \tag{5.8}$$

$$\begin{aligned}&\texttt{holdsAt}(F{=}V,\,T) \leftarrow \\ &\qquad \texttt{holdsAt}(F{=}V,\,T{-}1), \\ &\qquad \texttt{negate}_2(\texttt{clipped}(F{=}V,\,T{-}1)).\end{aligned} \tag{5.9}$$

$$\begin{aligned}&\texttt{clipped}(F{=}V,\,T) \leftarrow \\ &\qquad \texttt{terminatedAt}(F{=}V,\,T).\end{aligned} \tag{5.10}$$

$$\begin{aligned}&\texttt{clipped}(F{=}V,\,T) \leftarrow \\ &\qquad \texttt{initiatedAt}(F{=}V',\,T), \\ &\qquad V \neq V'.\end{aligned} \tag{5.11}$$

According to rule (5.7), $F{=}V$ holds at time-point 0 if $F{=}V$ held initially. According to rule (5.8), $F{=}V$ holds at time-point T if the fluent F has been initiated with value V at

the previous time-point. According to rule (5.9) $F{=}V$ continues to hold at time-point T if $F{=}V$ held at the previous time-point and it was not `clipped`. In particular, $F{=}V$ is `clipped` either when it is terminated (rule (5.10)) or the fluent F is initiated with a different value V' (rule (5.11)). Therefore, rule (5.11) ensures that a fluent cannot have more than one value at any time. We do not insist that a fluent must have a value at every time-point. In ProbLog–EC there is a difference between initiating a Boolean fluent $F{=}$`false` and terminating $F{=}$`true`: the first implies, but is not implied by the second.

The definitions of `initiatedAt` and `terminatedAt` are domain-specific. The domain-dependent rules of ProbLog–EC, i.e., the initiation and/or termination of some F to value V takes the following general form:

$$\begin{aligned}
&\texttt{initiatedAt}(F{=}V,\, T) \leftarrow \\
&\qquad \texttt{happensAt}(E_i,\, T),\ \ldots, \\
&\qquad \texttt{holdsAt}(F_j{=}V_l,\, T),\ \ldots, \\
&\qquad \texttt{Conditions}[T] \\
&\texttt{terminatedAt}(F{=}V,\, T) \leftarrow \\
&\qquad \texttt{happensAt}(E_k,\, T),\ \ldots, \\
&\qquad \texttt{holdsAt}(F_l{=}V_l,\, T),\ \ldots, \\
&\qquad \texttt{Conditions}[T]
\end{aligned} \tag{5.12}$$

where `Conditions`$[T]$ is some set of further conditions referring to time-point T. Note that in this Event Calculus formulation, `initiatedAt`$(F{=}V,\, T)$ does not necessarily imply that $F{\neq}V$ at T. Similarly, `terminatedAt`$(F{=}V,\, T)$ does not necessarily imply that $F{=}V$ at T. Suppose, for example, that $F{=}V$ is initiated at time-point 20 and terminated at time-point 30 and that there are no other time-points at which it is initiated or terminated. Then $F{=}V$ holds at all time-points T such that $20 < T \leq 30$. Suppose now that $F{=}V$ is initiated at time-points 10 and 20 and terminated at time-point 30 (and at no other time-points). Then $F = V$ holds at all T such that $10 < T \leq 30$. And suppose finally that $F{=}V$ is initiated at time-points 10 and 20 and terminated at time-points 25 and 30 (and at no other time-points). In that case $F{=}V$ holds at all T such that $10 < T \leq 25$. The predicates `happens` and `holdsAt`, as well as those appearing in `Conditions`$[T]$, may also be negated using the utility predicates $\texttt{negate}_1$ and $\texttt{negate}_2$ (see rules (5.2)–(5.6)). In the following section we illustrate the use of `initiatedAt` and `terminatedAt` rules for expressing CE definitions.

5.2 Application to Activity Recognition

To demonstrate our method we use a modified version of the first dataset of the CAVIAR project[1]. As presented in Section 2.1, the aim in activity recognition is to recognise CEs that take place between multiple persons, given a stream of SDEs. The videos are staged — e.g., actors walk around, sit down, meet one another, leave objects behind, fight, and so on. Each video has been manually annotated by the CAVIAR team, in order to provide the ground truth for both SDEs and CEs. Specifically, the input to ProbLog–EC is formed by the following:

(i) Input SDEs `walking`, `running`, `active` (i.e., non-abrupt body movement in the same position) and `inactive` (i.e., standing still), together with their time-stamps. The original CAVIAR dictionary does not include a SDE for abrupt motion. Specifically, abrupt motion is a form of SDE that can be detected by some state-of-the-art detection systems (e.g., Kosmopoulos et al. [2008]). Our preliminary experiments with this dataset showed that the absence of such SDE compromises the recognition accuracy of some CEs. We have manually modified the SDE annotation of the dataset by changing, when necessary, the label of an SDE to `abrupt`.

All SDEs are mutually exclusive and represented as a narrative by means of ground `happensAt` predicates. For example, $\texttt{happensAt}(\texttt{active}(id_5), 80)$ expresses that $\texttt{id}_6$ displayed active bodily movement at time-point 80. Additionally, the first and the last time a person or object is tracked (i.e., appear and disappear) is represented using the `happensAt` predicate. For example, $\texttt{happensAt}(\texttt{appear}(id_{10}), 300)$ expresses that $\texttt{id}_{10}$ is first tracked at time-point 300.

(ii) The coordinates of the tracked people and objects as pixel positions at each time-point, as well as their orientation. The coordinates are represented with the use of the `holdsAt` predicate. For example, $\texttt{holdsAt}(\texttt{coord}(id_2){=}(14, 55), 10600)$, expresses that the coordinates of $\texttt{id}_2$ are $(14, 55)$ at time-point 10600. Orientation is similarly represented, e.g., $\texttt{holdsAt}(\texttt{orientation}(id_2){=}120, 10600)$ expresses that, in the two-dimensional projection of the video, the same person was forming a 120° angle with the x-axis at the same time-point. This type of information is necessary for computing the distance between two entities, as well as the direction in which a person might be headed.

Event recognition is based on a manually developed knowledge base of CE definitions expressed in terms of `initiatedAt` and `terminatedAt`. Below we present the definitions of the CE knowledge base.

[1]The modified version of the dataset, including ProbLog–EC, the CE definitions and the event recognition results are freely available at: http://www.iit.demokritos.gr/~anskarl/pub/ProbLog-EC

The CE that expresses the `leaving an object` activity is defined by the following rules:

$$\begin{aligned}&\texttt{initiatedAt}(\texttt{leaving\_object}(P,\ Obj)\texttt{=true},\ T) \leftarrow \\ &\quad \texttt{happensAt}(\texttt{appear}(Obj),\ T), \\ &\quad \texttt{happensAt}(\texttt{inactive}(Obj),\ T), \\ &\quad \texttt{holdsAt}(\texttt{close}(P,\ Obj,\ 30)\texttt{=true},\ T), \\ &\quad \texttt{holdsAt}(\texttt{person}(P)\texttt{=true},\ T).\end{aligned} \tag{5.13}$$

$$\begin{aligned}&\texttt{terminatedAt}(\texttt{leaving\_object}(P,\ Obj)\texttt{=true},\ T) \leftarrow \\ &\quad \texttt{happensAt}(\texttt{disappear}(Obj),\ T).\end{aligned} \tag{5.14}$$

In the CAVIAR dataset an object carried by a person is not tracked — only the person that carries it is tracked. The object will be tracked (i.e., appear) only when the person leaves it somewhere. Moreover, objects (as opposed to persons) can only exhibit the `inactive` SDE. Accordingly, rule (5.13) expresses the conditions in which leaving an object is recognised. The fluent recording this activity, $\texttt{leaving\_object}(P, Obj)$, becomes `true` at time T when object Obj appears at T, its SDE at T is inactive, and there is a person P close to Obj at T. Fluent $\texttt{close}(ID_1, ID_2, Threshold)$ expresses that the distance between ID_1 and ID_2 is at most $Threshold$ pixels. This fluent is defined as follows:

$$\begin{aligned}&\texttt{holdsAt}(\texttt{close}(ID_1, ID_2, Threshold)\texttt{=true},\ T) \leftarrow \\ &\quad \texttt{holdsAt}(\texttt{distance}(ID_1, ID_2)\texttt{=}Dist,\ T), \\ &\quad Dist < Threshold.\end{aligned} \tag{5.15}$$

According to rule (5.15), the distance between two tracked objects/people is computed as the Euclidean distance between their coordinates in the two-dimensional projection of the video. All distance threshold values in the knowledge base were determined from an empirical analysis of the dataset.

The CE `leaving an object` is terminated when the object that is picked up by someone is no longer tracked (it disappears) — see rule (5.14).

In the dataset there is no explicit information that a tracked entity is a person or an inanimate object. To deduce whether a tracked entity is a person or an object given the detected SDEs, we define the utility fluent $\texttt{person}(P)$ to have value `true` when P was active, walking, running or moved abruptly at some time-point since P appeared.

$$\begin{aligned}&\texttt{initiatedAt}(\texttt{person}(P)\texttt{=true},\ T) \leftarrow \\ &\quad \texttt{happensAt}(\texttt{active}(P),\ T).\end{aligned}$$

$$\begin{aligned}&\texttt{initiatedAt}(\texttt{person}(P)\texttt{=true},\ T) \leftarrow \\ &\quad \texttt{happensAt}(\texttt{walking}(P),\ T).\end{aligned}$$

$$\begin{aligned}&\texttt{initiatedAt}(\texttt{person}(P)\texttt{=true},\ T) \leftarrow \\ &\quad \texttt{happensAt}(\texttt{running}(P),\ T).\end{aligned}$$

$$\begin{aligned}&\texttt{initiatedAt}(\texttt{person}(P)\texttt{=true}, T) \leftarrow\\ &\qquad \texttt{happensAt}(\texttt{abrupt}(P), T).\end{aligned}$$

$$\begin{aligned}&\texttt{terminatedAt}(\texttt{person}(P)\texttt{=true}, T) \leftarrow\\ &\qquad \texttt{happensAt}(\texttt{disappear}(P), T).\end{aligned}$$

The value of $\texttt{person}(P)$ is time-dependent because in CAVIAR the identifier P of a tracked entity that disappears (is no longer tracked) at some point may be used later to refer to another entity that appears (becomes tracked), and that other entity may not necessarily be a person. Note, finally, that rule (5.13) incorporates a (reasonable) simplifying assumption, that a person entity will never exhibit inactive activity at the moment it first appears (is tracked). If an entity is inactive at the moment it appears it can be assumed to be an object, as in the first two conditions of rule (5.13).

The CE `meeting` is recognised when two people interact with each other, i.e., at least one of them is active or inactive, the other is not running or moving abruptly, and the distance between them is at most 25 pixels. The following rules show the conditions in which `meeting` is initiated:

$$\begin{aligned}&\texttt{initiatedAt}(\texttt{meeting}(P_1, P_2)\texttt{=true}, T) \leftarrow\\ &\qquad \texttt{happensAt}(\texttt{inactive}(P_1), T),\\ &\qquad \texttt{holdsAt}(\texttt{close}(P_1, P_2, 25)\texttt{=true}, T),\\ &\qquad \texttt{holdsAt}(\texttt{person}(P_1)\texttt{=true}, T),\\ &\qquad \texttt{holdsAt}(\texttt{person}(P_2)\texttt{=true}, T),\\ &\qquad \texttt{negate}_1(\texttt{happensAt}(\texttt{running}(P_2), T)),\\ &\qquad \texttt{negate}_1(\texttt{happensAt}(\texttt{abrupt}(P_2), T)),\\ &\qquad \texttt{negate}_1(\texttt{happensAt}(\texttt{active}(P_2), T)).\end{aligned} \tag{5.16}$$

$$\begin{aligned}&\texttt{initiatedAt}(\texttt{meeting}(P_1, P_2)\texttt{=true}, T) \leftarrow\\ &\qquad \texttt{happensAt}(\texttt{active}(P_1), T),\\ &\qquad \texttt{holdsAt}(\texttt{close}(P_1, P_2, 25)\texttt{=true}, T),\\ &\qquad \texttt{holdsAt}(\texttt{person}(P_2)\texttt{=true}, T),\\ &\qquad \texttt{negate}_1(\texttt{happensAt}(\texttt{running}(P_2), T)),\\ &\qquad \texttt{negate}_1(\texttt{happensAt}(\texttt{abrupt}(P_2), T)).\end{aligned} \tag{5.17}$$

The CE `meeting` is terminated by the conditions of the following rules:

$$\begin{aligned}&\texttt{terminatedAt}(\texttt{meeting}(P_1, P_2)\texttt{=true}, T) \leftarrow\\ &\qquad \texttt{happensAt}(\texttt{walking}(P_1), T),\\ &\qquad \texttt{holdsAt}(\texttt{close}(P_1, P_2, 34)\texttt{=false}, T).\end{aligned} \tag{5.18}$$

$$\texttt{terminatedAt}(\texttt{meeting}(P_1, P_2)=\texttt{true}, T) \leftarrow \texttt{happensAt}(\texttt{running}(P_1), T). \tag{5.19}$$

$$\texttt{terminatedAt}(\texttt{meeting}(P_1, P_2)=\texttt{true}, T) \leftarrow \texttt{happensAt}(\texttt{abrupt}(P_1), T). \tag{5.20}$$

$$\texttt{terminatedAt}(\texttt{meeting}(P_1, P_2)=\texttt{true}, T) \leftarrow \texttt{happensAt}(\texttt{disappear}(P_1), T). \tag{5.21}$$

The `meeting` activity stops being recognised when the two people are moving away (rule (5.18)) or when one of the two people involved in the CE starts running (rule (5.19)), moves abruptly (rule (5.20)) or disappears from the scene (rule (5.21)).

The CE `meeting` may have multiple initiations, indicating that two people may be interacting at several time-points. Similarly, `meeting` may have multiple terminations. This is in contrast to `leaving_object` where there is a single initiation and a single termination, i.e., an object appears once before it disappears. In general, there is no fixed relation between the number of initiations and terminations of a fluent, e.g., a CE may have multiple initiations and a single termination.

The CE `moving` represents the activity that two people are walking along together and is defined as follows:

$$\begin{aligned}\texttt{initiatedAt}(\texttt{moving}(P_1, P_2)=\texttt{true}, T) \leftarrow\\ &\texttt{happensAt}(\texttt{walking}(P_1), T),\\ &\texttt{happensAt}(\texttt{walking}(P_2), T),\\ &\texttt{holdsAt}(\texttt{close}(P_1, P_2, 34)=\texttt{true}, T),\\ &\texttt{holdsAt}(\texttt{orientation}(P_1)=Or_1, T),\\ &\texttt{holdsAt}(\texttt{orientation}(P_2)=Or_2, T),\\ &|Or_1 - Or_2| < 45.\end{aligned} \tag{5.22}$$

According to rule (5.22), in order to recognise the CE `moving`, both people involved have to be `walking` while being close to each other. In addition, they have to be facing towards, more or less, the same direction (people walking in opposite directions are not assumed to be walking along together). This is accomplished by constraining their orientations so that they are headed towards the same area while they are walking.

The termination definitions of `moving` are given below:

$$\begin{aligned}\texttt{terminatedAt}(\texttt{moving}(P_1, P_2)=\texttt{true}, T) \leftarrow\\ &\texttt{happensAt}(\texttt{walking}(P_1), T),\\ &\texttt{holdsAt}(\texttt{close}(P_1, P_2, 34)=\texttt{false}, T).\end{aligned} \tag{5.23}$$

$$\begin{aligned}\texttt{terminatedAt}(\texttt{moving}(P_1, P_2)&=\texttt{true}, T) \leftarrow \\ &\texttt{happensAt}(\texttt{active}(P_1), T), \\ &\texttt{happensAt}(\texttt{active}(P_2), T).\end{aligned} \tag{5.24}$$

$$\begin{aligned}\texttt{terminatedAt}(\texttt{moving}(P_1, P_2)&=\texttt{true}, T) \leftarrow \\ &\texttt{happensAt}(\texttt{active}(P_1), T), \\ &\texttt{happensAt}(\texttt{inactive}(P_2), T).\end{aligned} \tag{5.25}$$

$$\begin{aligned}\texttt{terminatedAt}(\texttt{moving}(P_1, P_2)&=\texttt{true}, T) \leftarrow \\ &\texttt{happensAt}(\texttt{running}(P_1), T).\end{aligned} \tag{5.26}$$

$$\begin{aligned}\texttt{terminatedAt}(\texttt{moving}(P_1, P_2)&=\texttt{true}, T) \leftarrow \\ &\texttt{happensAt}(\texttt{abrupt}(P_1), T).\end{aligned} \tag{5.27}$$

$$\begin{aligned}\texttt{terminatedAt}(\texttt{moving}(P_1, P_2)&=\texttt{true}, T) \leftarrow \\ &\texttt{happensAt}(\texttt{disappear}(P_1), T).\end{aligned} \tag{5.28}$$

According to rule (5.23), `moving` is terminated when either person walks away from the other with respect to the predefined threshold of 34 pixels. Furthermore, `moving` is terminated when both persons are staying active (rule (5.24)), when one of them is inactive while the other remains active (rule (5.25)), when either person is running away from the other (rule (5.26)) or is moving abruptly (rule (5.27)), as well as when one of them disappears from the scene (rule (5.28)).

The last definition concerns the activity that two persons are `fighting`:

$$\begin{aligned}\texttt{initiatedAt}(\texttt{fighting}(P_1, P_2)&=\texttt{true}, T) \leftarrow \\ &\texttt{happensAt}(\texttt{abrupt}(P_1), T), \\ &\texttt{holdsAt}(\texttt{close}(P_1, P_2, 44)=\texttt{true}, T), \\ &\texttt{negate}_1(\texttt{happensAt}(\texttt{inactive}(P_2), T)).\end{aligned} \tag{5.29}$$

Rule (5.29) defines that in order to recognise CE `fighting` both people must be sufficiently close to each other and at least one of them moves abruptly, while the other one is not `inactive` (i.e., participating in the fight). `fighting` is terminated according to the following rules:

$$\begin{aligned}\texttt{terminatedAt}(\texttt{fighting}(P_1, P_2)&=\texttt{true}, T) \leftarrow \\ &\texttt{happensAt}(\texttt{walking}(P_1), T), \\ &\texttt{holdsAt}(\texttt{close}(P_1, P_2, 44)=\texttt{false}, T),\end{aligned} \tag{5.30}$$

$$\begin{aligned}\texttt{terminatedAt}(\texttt{fighting}(P_1, P_2)&=\texttt{true}, T) \leftarrow \\ &\texttt{happensAt}(\texttt{running}(P_1), T), \\ &\texttt{holdsAt}(\texttt{close}(P_1, P_2, 44)=\texttt{false}, T),\end{aligned} \tag{5.31}$$

$$\texttt{terminatedAt}(\texttt{fighting}(P_1, P_2){=}\texttt{true}, T) \leftarrow \texttt{happensAt}(\texttt{disappear}(P_1), T). \tag{5.32}$$

The `fighting` activity ceases to be recognised when either person involved in the activity starts walking away (5.30)), running away ((5.31)), or exits from the scene ((5.32)).

According to the presented definitions, all CEs are not mutually exclusive, in contrast to SDEs. For example, `meeting` may overlap with `moving`: two people interact and then start `moving`, that is, they walk while being `close` to each other. In general, however, there is no defined relationship between CE.

5.3 The Behaviour of the ProbLog–EC

In contrast to a typical pure logic-based Event Calculus (e.g., the Event Calculus in Section 2.2), the input SDEs in ProbLog–EC can be associated with a probability value, i.e., $Prob :: \texttt{happensAt}(SDE, T)$. As a result, CEs are not being initiated or terminated with absolute certainty, but with some probability. This situation affects the behaviour of the Event Calculus, causing the recognition of a CE to hold with some probability. Specifically, an initiation increases the recognition probability of a CE, while a termination degreases that probability. To illustrate this behaviour we present the inference procedure of ProbLog–EC with two examples, one with multiple initiations and terminations, and one with a single initiation and a single termination.

5.3.1 Multiple Initiations and Terminations

To illustrate how multiple initiation and terminations affect the recognition probability of a CE, suppose that two persons (identified as $\texttt{id}_1$ and $\texttt{id}_2$) are engaging in a `moving` activity for a number of time-points. As presented in Figure 5.1, the CE is first initiated at time-point 1, when both $\texttt{id}_1$ and $\texttt{id}_2$ start walking. At time-point 2, person $\texttt{id}_2$ stops walking and stays active (SDE `walking` is required by rule (5.22) to initiate the CE `moving`). Furthermore, person $\texttt{id}_1$ continues to walk but does not move far enough from $\texttt{id}_2$ to terminate `moving`. At time-point 21 $\texttt{id}_2$ resumes `walking`, once again initiating `moving`. Later, at time-point 41 person $\texttt{id}_2$ continues to walk, but $\texttt{id}_1$ stays `inactive` and is thus left behind. Therefore, the two persons are no longer close enough to each other, which triggers the termination of `moving` (rule (5.23)).

For simplicity, let all information pertaining to orientation and coordinates hold with absolute certainty (i.e., probability of 1). Moreover, assume that the SDEs `walking`,

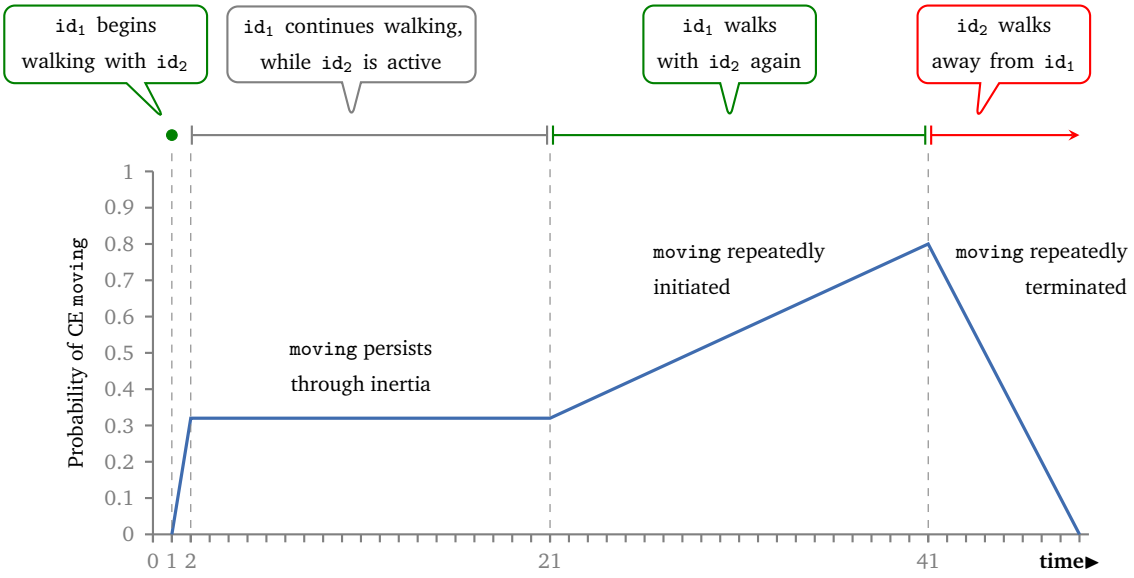


FIGURE 5.1: Example of CE probability fluctuation in the presence of multiple initiations and terminations.

`active` and `inactive` have probabilities attached, as follows:

$$0.70 :: \texttt{happensAt}(\texttt{walking}(\texttt{id}_1), 1)$$
$$0.46 :: \texttt{happensAt}(\texttt{walking}(\texttt{id}_2), 1)$$
$$0.73 :: \texttt{happensAt}(\texttt{walking}(\texttt{id}_1), 2)$$
$$0.55 :: \texttt{happensAt}(\texttt{active}(\texttt{id}_2), 2)$$
$$\ldots$$
$$0.69 :: \texttt{happensAt}(\texttt{walking}(\texttt{id}_1), 21)$$
$$0.58 :: \texttt{happensAt}(\texttt{walking}(\texttt{id}_2), 21)$$
$$\ldots$$
$$0.18 :: \texttt{happensAt}(\texttt{inactive}(\texttt{id}_1), 41)$$
$$0.32 :: \texttt{happensAt}(\texttt{walking}(\texttt{id}_2), 41)$$
$$\ldots$$

At time-point 2, the query $\texttt{holdsAt}(\texttt{moving}(\texttt{id}_1, \texttt{id}_2){=}\texttt{true}, 2)$ has a probability equal to the probability of the initiation condition of time-point 1. Specifically, according to rule (5.22) and given that all coordinate and orientation-related information have probability 1, the probability of initiation at time-point 1 is the product of the probabilities that both $\texttt{id}_1$ and $\texttt{id}_2$ are walking — i.e., $0.70{\times}0.46 = 0.322$. This is visualised at the far left of Figure 5.1, where the CE's probability increases from 0 to 0.322. During the interval 2 to 20 no initiation or termination conditions are fired and the probability of `moving` remains unchanged. This occurs due to the law of inertia (rule (5.9)) and is depicted graphically by the horizontal line between time-points 2 and 21. At time-point

21, $\texttt{id}_1$ starts walking alongside $\texttt{id}_2$ again. Consequently, at time-point 22, the query $\texttt{holdsAt}(\texttt{moving}(\texttt{id}_1, \texttt{id}_2)\texttt{=true}, 22)$ has two initiation conditions to consider: one fired at time-point 1 and one at time-point 21, due to rules (5.8) and (5.9), respectively.

As mentioned in Section 2.3.2, ProbLog computes the probability of a query by first scanning the entire Selective Linear Definite Tree (SLD) tree of the query. Figure 5.2 depicts a fragment of the SLD tree for the query $\texttt{holdsAt}(\texttt{moving}(\texttt{id}_1, \texttt{id}_2)\texttt{=true}, 22)$. Then ProbLog represents these proofs as a Disjunctive Normal Form (DNF) formula. In our case, the DNF is the following:

$$\underbrace{\texttt{initiatedAt}(\texttt{moving}(\texttt{id}_1, \texttt{id}_2)\texttt{=true}, 1)}_{\texttt{init}_1} \bigvee \underbrace{\texttt{initiatedAt}(\texttt{moving}(\texttt{id}_1, \texttt{id}_2)\texttt{=true}, 21)}_{\texttt{init}_{21}} \tag{5.33}$$

We have simplified the representation by omitting the all relevant $\texttt{negate}_2$ clauses since, as can be seen in Figure 5.2, they are provable through negation as failure and therefore have a probability of 1. This occurs because no termination conditions for `moving` have been fired between time-points 1 and 21.

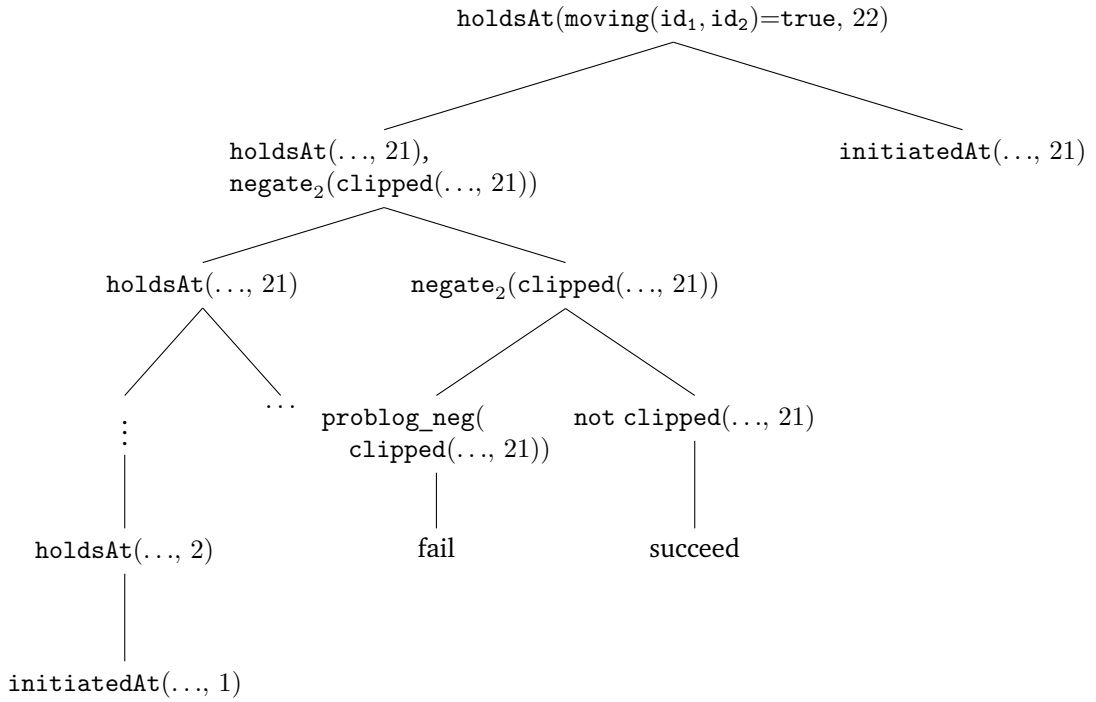


FIGURE 5.2: A fragment of the SLD tree for CE `moving` at time-point 22.

Up to time-point 21, there exist two initiation conditions for the `moving` CE, $\texttt{init}_1$ and $\texttt{init}_{21}$ (see formula (5.33)). In the general case, there may exist many more initiation conditions in the interval between the start of the video and the examined time-point. In addition, for every time-point, rule (5.9) checks recursively whether the CE holds at the previous time-points, without been interrupted (i.e., terminated or initiated with different value). This repeating process leads to numerous redundant computations. We

overcame this problem by implementing an elementary caching technique according to which the probability of $\texttt{holdsAt}(F{=}V, T{-}1)$ is stored in memory and, therefore, $\texttt{holdsAt}(F{=}V, T)$ simply checks to see how the initiation or termination conditions (if any) fired at time-point $T{-}1$ affect this probability. This technique operates under the assumption that the activity recognition system receives the video frames in a temporally sorted manner.

The next step of ProbLog inference involves translating the DNF into a BDD. However, our example is simple enough to allow us to perform manual calculations, as there exist only two proofs for holdsAt, which are easily separable. The probability of DNF formula (5.33) can be computed as the probability of a disjunction of two elements, as explained in Section 2.3.2:

$$\begin{aligned} P(\texttt{init}_1 \vee \texttt{init}_{21}) &= P(\texttt{init}_1) + P(\texttt{init}_{21}) - P(\texttt{init}_1 \wedge \texttt{init}_{21}) \\ &= 0.70 \times 0.46 + 0.69 \times 0.58 - 0.7 \times 0.46 \times 0.69 \times 0.58 \\ &= 0.593 \end{aligned}$$

Therefore, the probability that $\texttt{id}_1$ and $\texttt{id}_2$ are moving together at time-point 22 has increased, owing to the presence of the additional initiation at time-point 21. Similar to MLN–EC, in ProbLog–EC the continuous presence of initiation conditions of a particular CE causes an increase of the probability of the CE. This behaviour is consistent with our intuition: given continuous indication that an activity has (possibly) occurred, we are more inclined to accept that it has indeed taken place, even if the confidence of each individual indication is low. For this reason, from time-point 22 up to and including time-point 41, the probability of moving increases, as is visible in Figure 5.1. In this example, at time-point 41 the activity's probability has increased to around 0.8.

At time-point 42, ProbLog–EC has to take into consideration the satisfaction of the termination condition at time-point 41. This termination, corresponding to rule (5.23), is also probabilistic. According to rule (5.23) and the fact that close is detected with absolute certainty, the probability that $\texttt{id}_2$ walked away from $\texttt{id}_1$. is equal to the probability of the walking SDE itself, which is 0.32. Therefore, when estimating the probability that at time-point 42 $\texttt{id}_1$ and $\texttt{id}_2$ are still moving together, we have to incorporate the probability of all possible worlds in which $\texttt{id}_2$ did not walk away from $\texttt{id}_1$. The probability of these worlds is computed by the use of $\texttt{negate}_2$ in rule (5.9) and is equal to $1{-}0.32 = 0.68$. Consequently, the probability that $\texttt{id}_1$ and $\texttt{id}_2$ are still moving together at time-point 42 is $0.8{\times}0.68 = 0.544$. Similarly to the steady probability increase given continuous initiation conditions, when faced with subsequent termination conditions, the probability of the CE will steadily decrease. The slope of the descent (ascent) is defined by the probability of the termination (initiation) conditions involved. For this example, we assume that $\texttt{id}_2$ keeps walking away from $\texttt{id}_1$ until the end of the video, causing the probability of the CE to converge to 0, as shown at the far right of Figure 5.1.

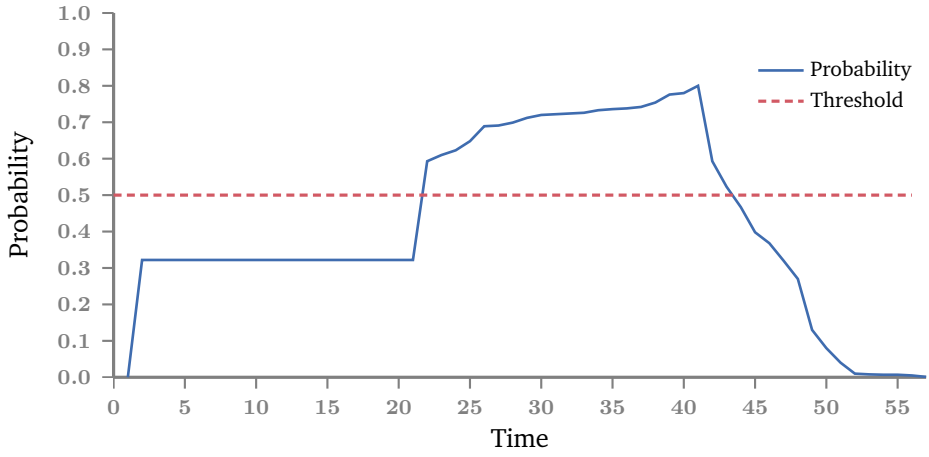


FIGURE 5.3: The probability of CE `moving` for various time-points.

Figure 5.3 shows the real output of ProbLog for our example. Following an abrupt jump from 0 to 0.322, the probability remains stable between time-points 2 and 21, indicating that for this period of time the CE persists through the law of inertia. Between time-points 22 and 41, the probability of the CE increases monotonically, reflecting the repeated initiations of `moving` that occur during that time. After time-point 41, the probability decreases, reflecting the repeated termination conditions occurring at the same time period. The dashed horizontal line at probability 0.5 represents the recognition threshold that we use to discern between CE instances that we consider to be trustworthy enough and those that we do not. The choice of 0.5 as a threshold serves simply the purposes of illustration — other thresholds could have been chosen.

5.3.2 Single Initiation and Termination

In this section we illustrate the behaviour of ProbLog–EC in the case of fluents with a single initiation and a single termination. Assume that person $\texttt{id}_2$ is walking while simultaneously carrying a suitcase for 10 time-points. At time-point 11, $\texttt{id}_2$ leaves the suitcase on the floor and walks away from it. This causes the suitcase to appear in the low-level tracking system, triggering the `leaving_object` initiation condition expressed by rule (5.13). Suppose that this initiation has a probability of 0.6. At time-point 20, $\texttt{id}_2$ picks up the suitcase, causing it to disappear, triggering the termination condition expressed by rule (5.14). Suppose that the termination also has a probability of 0.6. Figure 5.4 depicts the probability fluctuation of `leaving_object` in this case. As can be seen in this figure, in the absence of any initiations after time-point 11, the CE persists entirely due to the law of inertia. The probability of this CE is equal to the probability of the single initiation, i.e., 0.6. Because this probability is above the chosen recognition threshold — 0.5 in this example — the CE is considered to hold for all time-points until $\texttt{id}_2$ picks up the suitcase. If the probability of the initiation was below the threshold, then `leaving_object` would not have been recognised.

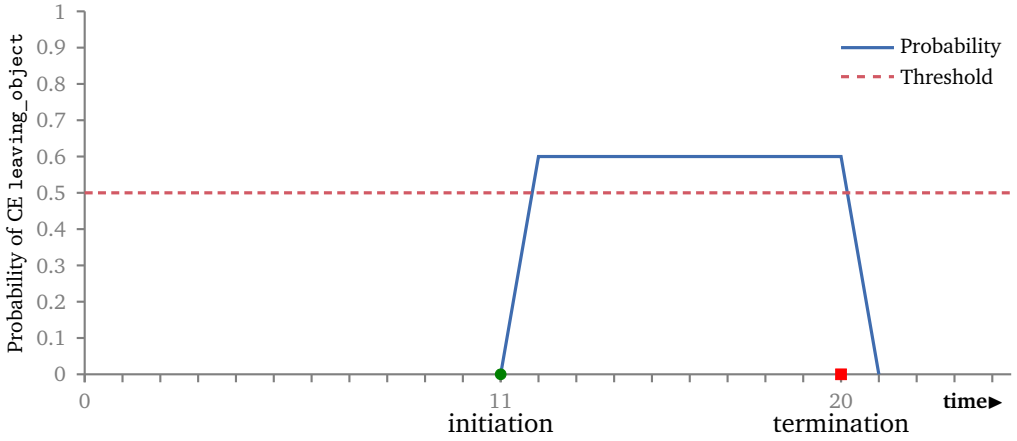


FIGURE 5.4: Example of CE probability fluctuation in the presence of a single initiation and a single termination.

Note that we may transform a fluent with a single initiation to a version of that fluent with multiple initiations — consider the following formalisation:

$$\begin{aligned} &\texttt{initiatedAt}(\mathit{leaving\_object\_mi}(P,\ \mathit{Obj}) = \texttt{true},\ T) \leftarrow \\ &\quad \texttt{holdsAt}(\mathit{leaving\_object}(P,\ \mathit{Obj}) = \texttt{true},\ T) \end{aligned} \tag{5.34}$$

`leaving_object_mi` may have multiple initiations — it is initiated as long as `leaving_object` holds. While this transformation would have been beneficial for our experiments in the CAVIAR dataset, since we would be able to augment the probability of `leaving_object_mi` through multiple initiations and eventually surpass the chosen recognition threshold, it introduces some subtle perils. Consider, for example, a scenario in which a `leaving_object` CE is initiated with a small probability, such as 0.1, indicating that the sensor's confidence about the SDE appearing in rule (5.13) is low. ProbLog–EC will then compute that `leaving_object` holds with a probability of 0.1 up until the time-point, if any, where the object in question is picked up. These positives will be correctly discarded under most recognition thresholds given their small probabilities. `leaving_object_mi`, however, will continue to augment its probability, eventually surpassing the chosen threshold and thus producing a potentially large number of false positives. While it is true that such situations do not arise in CAVIAR, it is very likely that they take place in other activity recognition applications.

The `leaving_object` CE in Figure 5.4 has a single termination. In this example, the probability of the termination degreases the probability of `leaving_object` below the chosen threshold of 0.5. In other examples, the absence of subsequent terminations may not allow the probability of an CE to drop below the chosen recognition threshold, thus possibly resulting in false persistence.

5.4 Conclusions

We presented ProbLog–EC, a discrete Event Calculus dialect for probabilistic reasoning. ProbLog–EC is the first probabilistic Event Calculus dialect that is able to deal with uncertainty in the input SDE. In particular, SDEs that may recognised using other lower-level classification techniques are given to ProbLog–EC with their recognition probabilities. These probabilities represent a degree of belief, which is directly propagated to the ProbLog–EC. As a result, CEs are probabilistically recognised.

Another characteristic of the ProbLog–EC is that, similar to MLN–EC, in situations where the continuous presence of SDEs initiate (terminate) a particular CE causes the increase (degrease) of the belief the CE. The more continuous indication of initiations (terminations), the more confident we are for the occurrence (absence) of the corresponding CE.

6 Experimental Evaluation of ProbLog–EC

"No amount of experimentation can ever prove me right;
a single experiment can prove me wrong."
— Albert Einstein

In this chapter we evaluate our ProbLog–EC method in the domain of video activity recognition. We use a modified version of the publicly available benchmark dataset of the CAVIAR project (see Section 5.2). The aim of the experiments is to assess the effectiveness of ProbLog–EC in recognising composite events (CEs) that occur among people, in the presence of noisy narratives of input simple, derived events (SDEs).

ProbLog–EC combines the benefits of logic-based representation (e.g., direct expression of domain background knowledge) with probabilistic modelling of input SDEs. For comparison purposes, we include the logic-based activity recognition method of Artikis et al. [2010b], which we call here $\texttt{EC}_{\texttt{crisp}}$. Similar to ProbLog–EC, $\texttt{EC}_{\texttt{crisp}}$ employs a variant of the Event Calculus and uses the same definitions of CEs (see Section 5.2). Unlike ProbLog–EC, $\texttt{EC}_{\texttt{crisp}}$ cannot perform any type of probabilistic reasoning in order to handle the noise at the input SDEs.

6.1 Performance evaluation without Artificial Noise

Our empirical analysis is based on the 28 surveillance videos of the CAVIAR dataset which contain, in total, 26419 video frames. These frames have been manually annotated by the CAVIAR team to provide the ground truth for SDEs and CEs. We performed minor editing of the annotation in order to introduce a SDE for abrupt motion (see Section 5.2). According to the manual annotation of the dataset, SDEs are not associated with a probabilities. Table 6.1 shows the recognition results in terms of True Positives, False Positives, False Negatives, Precision, Recall and F-measure. These results have been produced by computing queries of the form $\texttt{holdsAt}(CE{=}\texttt{true}, T)$. SDEs are assumed to happen with probability 1, and therefore ProbLog–EC provides identical results to $\texttt{EC}_{\texttt{crisp}}$.

Composite Event	TP	FP	FN	Precision	Recall	F-measure
meeting	3099	1910	525	0.619	0.855	0.718
moving	4008	2162	2264	0.650	0.639	0.644
fighting	531	97	729	0.421	0.845	0.562
leaving_object	143	1539	55	0.085	0.722	0.152

TABLE 6.1: True Positives (TP), False Positives (FP), False Negatives (FN), Precision, Recall and F-measure on the dataset without artificial noise.

Particularly notable in the results is the low precision for the `leaving_object` CE, owing to a substantial number of false positives. This is due to the problematic annotation of the dataset with respect to this CE. For example, in video 14, the object that is left at frame 946, triggering an initiation of `leaving_object`, is picked up (disappears) at frame 1354. However, the relevant annotation mistakenly reports that `leaving_object` stops occurring at frame 996. We therefore end up with 358 false positives which could have been avoided with a more consistent annotation of this video. Similarly, in the annotation of videos 17 and 18, a large number of annotated frames are missing.

Video 16 includes a particularly interesting case of `leaving_object`. In this video, a person leaves a bag next to a chair, exits the scene, re-enters after a couple of seconds and picks up the bag. When the person re-enters a new identifier is been assigned (this is common in this dataset). Various complications arise due to this. The original `leaving_object` activity is not terminated by our rules when the person in question disappears. This is deliberate on our part: we choose to terminate `leaving_object` when the object is picked up rather than when the person that leaves it disappears from the sensor's view. We thus emphasize on time-points in which a package might be unattended. The annotation, however, views the `leaving_object` CE from a different perspective and thus assumes that the activity is terminated when the person disappears. This difference in perspective leaves us with a substantial number of false positives, one for each frame for which the person is absent from the scene. When the person re-enters the scene, the dataset provides a new identifier and resumes the annotation of `leaving_object` with a < *new_person_id, same_object_id* > tuple. After a couple of frames, the person (described by *new_person_id*) picks up the object, terminating a `leaving_object(new_person_id, same_object_id)` CE occurrence which $\texttt{EC}_{\texttt{crisp}}$ never initiated in the first place. Thus, in addition to a substantial number of false positives, we also generate 55 false negatives (see Table 6.1) because $\texttt{EC}_{\texttt{crisp}}$ never recognises the new `leaving_object` activity.

6.2 Performance evaluation with Artificial Noise

As mentioned above, according to the manual annotation of the CAVIAR dataset all SDEs can be assumed to happen with probability 1. In real-world activity recognition

applications, SDEs frequently are detected with some certainty. In order to experiment with that setting, we added artificial noise to the dataset in the form of probabilities attached to the input facts — SDEs and related coordinate and orientation information. Our experimental procedure may be summarised as follows:

(i) We inject noise into the dataset:

  (a) We add probabilities to SDEs. Toward this end, we use a Gamma distribution with a varying mean, in order to represent different levels of noise. Gamma distribution is commonly used in the literature for noise modelling — see Gonzalez and Woods [2007, 5.2] for an outline of common probability distributions for noise modelling.

  (b) In addition to SDEs, we add probabilities to their associated coordinate and orientation fluents. Although we use the same Gamma distribution for this step, SDEs are not required to have the same probability as their associated coordinate and orientation fluents.

  (c) On top of the above, we introduce spurious SDEs, that is, SDEs that are not part of the original dataset. We do this by augmenting the frames in which there is a `walking` SDE with another `walking` SDE and related coordinate and orientation information about an entity that does not exist in the dataset. We use a uniform distribution to choose the 5% of the frames that will be augmented with spurious facts. The probability of a spurious fact at frame t is $1-p$, where p is the probability of some `walking` SDE at t (recall that p is computed by the Gamma distribution). Consequently, the higher the mean, the lower the probabilities attached to the input SDEs and the higher the probability of the spurious facts, indicating a higher amount of noise.

  Thus we end up with three different noisy versions of the dataset. To facilitate the presentation that follows, we will call the three aforementioned approaches to noise injection *smooth*, *intermediate* and *strong* noise.

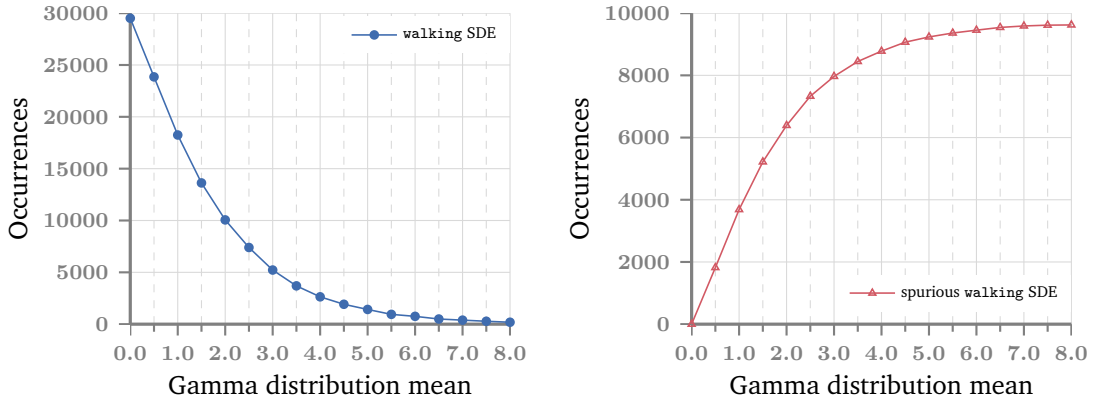
(ii) We feed these data to ProbLog–EC and filter its output — which is a series of CE instances of the form $Prob :: \texttt{holdsAt}(CE{=}\texttt{true}, T)$ — to keep only the ones with probability above a chosen threshold, indicating that we trust these to be accurate.

(iii) Once the recognition with ProbLog–EC is complete, we remove the SDEs with probability below the chosen threshold. We retain the SDEs with probability above the threshold, removing their probability values. Thus we assume that such SDEs have been tracked with certainty. For *smooth* noise, this step means that we remove a certain amount of the SDEs (e.g., `walking`, `running`, `active`, etc.), while for *intermediate* noise we additionally remove coordinate and orientation fluents. Furthermore, for *strong* noise we keep a certain amount of spurious information, based on the probability values.

(iv) We give these filtered versions of the dataset as input to $\texttt{EC}_{\texttt{crisp}}$. With this step we aim to estimate the impact of noise on $\texttt{EC}_{\texttt{crisp}}$, by assuming that $\texttt{EC}_{\texttt{crisp}}$ can only reason on facts that have been tracked with relative certainty, that is, facts with probability above the chosen threshold.

(v) We compare the performance of $\texttt{EC}_{\texttt{crisp}}$ and ProbLog–EC.

We repeated the above experimental procedure 16 times, once for each Gamma distribution mean value between 0.5 and 8.0 inclusive, with a step of 0.5. Noise is added randomly; for example, a SDE that is erased from the input of $\texttt{EC}_{\texttt{crisp}}$ for Gamma mean 6.0 might be present at the dataset produced for Gamma mean 6.5.



(a) `walking` SDE (b) spurious `walking` SDE

FIGURE 6.1: Occurrences of normal and spurious `walking` SDE with probability above 0.5 per Gamma mean value. Mean value 0.0 represents the dataset without artificial noise.

Figure 6.1(a) illustrates the noise injection on the dataset. The number of occurrences of `walking` SDEs with probability above the 0.5 threshold is plotted. Note that the number of `walking` occurrences shown in this figure does not include the spurious facts introduced in the *strong* noise experiments. The number of `walking` occurrences drops exponentially as we increase the level of noise, that is, the Gamma mean value. All other SDE follow the same pattern. In the case of intermediate noise, the occurrences of coordinate and orientation fluents also drop exponentially.

Figure 6.1(b) shows the number of occurrences of the spurious `walking` SDE — these are introduced in the *strong* noise experiments — with probability above the 0.5 threshold. The amount of spurious facts increases as we increase the level of noise (Gamma mean value).

Noise injection may significantly reduce recognition accuracy. The sections that follow illustrate this. However, given that there are mistakes in the original dataset (i.e., dataset without artificial noise), it may occur that a false positive in the original dataset becomes a true negative in the noisy version of the dataset. Similarly for false negatives. To

demonstrate this with an example, assume that two people are moving along together on a pavement. Suddenly they have to step aside to allow a disabled person full access to the pavement. This would fire the 'walk away' termination condition (see rule (5.23)) because the distance between the two people would exceed the pre-specified threshold. This does not mean that the people have stopped moving together — they merely had to distance themselves momentarily. Firing — erroneously — the 'walk away' termination condition creates false negatives for some video frames. If, however, the walking SDE gets a very low probability during the frames at which the people make way to the disabled person, moving will not be terminated, which, in this special case, adds true positives to the evaluation.

6.2.1 Experiments With Smooth Noise

Figure 6.2 compares the recognition accuracy of EC_{crisp} and ProbLog–EC in terms of F-measure under *smooth* noise using a 0.5 threshold. In all experiments we plot the F-measure per Gamma distribution mean averaged over five different runs — i.e., each point in the diagram is the average of five different F-measure values. The vertical error bars display the standard deviations.

The recognition accuracy of ProbLog–EC is higher than that of EC_{crisp} in the meeting, moving and fighting CEs. In meeting (see Figure 6.2(a)), the accuracy of EC_{crisp} starts falling from Gamma mean 5.5 onwards, because a significant number of active and inactive SDEs tend to be erased from its input, since they receive probability below 0.5 — the chosen threshold in the experiments presented in Figure 6.2 — when we generate noise. ProbLog–EC, on the other hand, is able to initiate meeting with a certain degree of probability (recall that rules (5.16) and (5.17) express the conditions in which meeting is initiated). After a number of frames, the repeated initiation of meeting leads to a holdsAt probability of 0.5 or above, and thus ProbLog–EC eventually recognises meeting (refer to Section 5.3.1 for a detailed explanation of this behaviour). Those initiation conditions occur very frequently in the dataset. Therefore, meeting ends up being recognised with a probability close to 1.

In the case of moving (see Figure 6.2(b)), it is the loss of multiple occurrences of walking (due to *smooth* noise) that causes EC_{crisp} to suffer from numerous false negatives (see rule (5.22) for the initiation condition of moving). ProbLog–EC, on the other hand, uses repeated initiation to eventually surpass the 0.5 recognition threshold. The accuracy of EC_{crisp} in moving deteriorates earlier than that of meeting. This occurs because the initiation condition of moving bears two probabilistic conjuncts in its body, corresponding to two separate occurrences of the walking SDE. Therefore, it is affected by noise twice. The initiation conditions of meeting may sometimes have two probabilistic conjuncts, but usually they have just one — the negate_1 conditions are usually crisp.

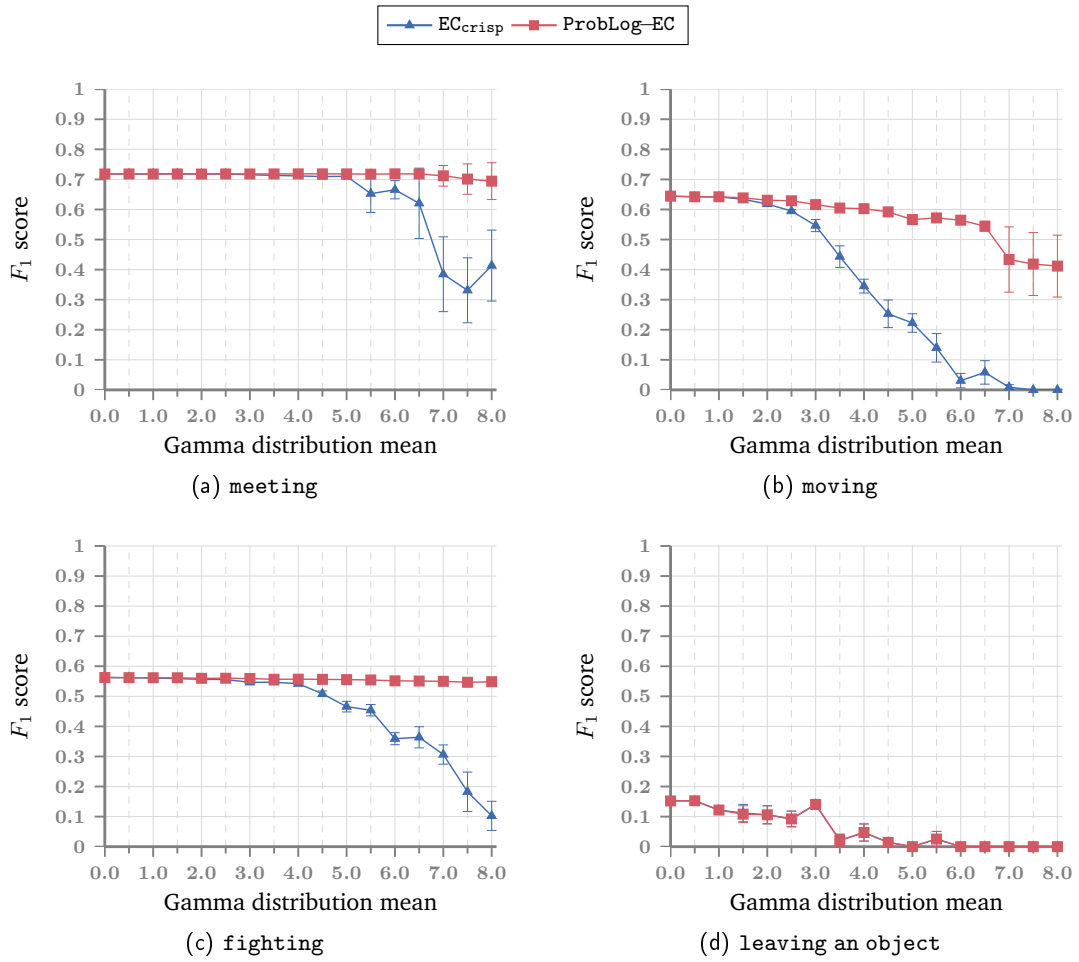


FIGURE 6.2: EC_{crisp} and ProbLog–EC F-measure per Gamma mean value under *smooth* noise and a 0.5 threshold.

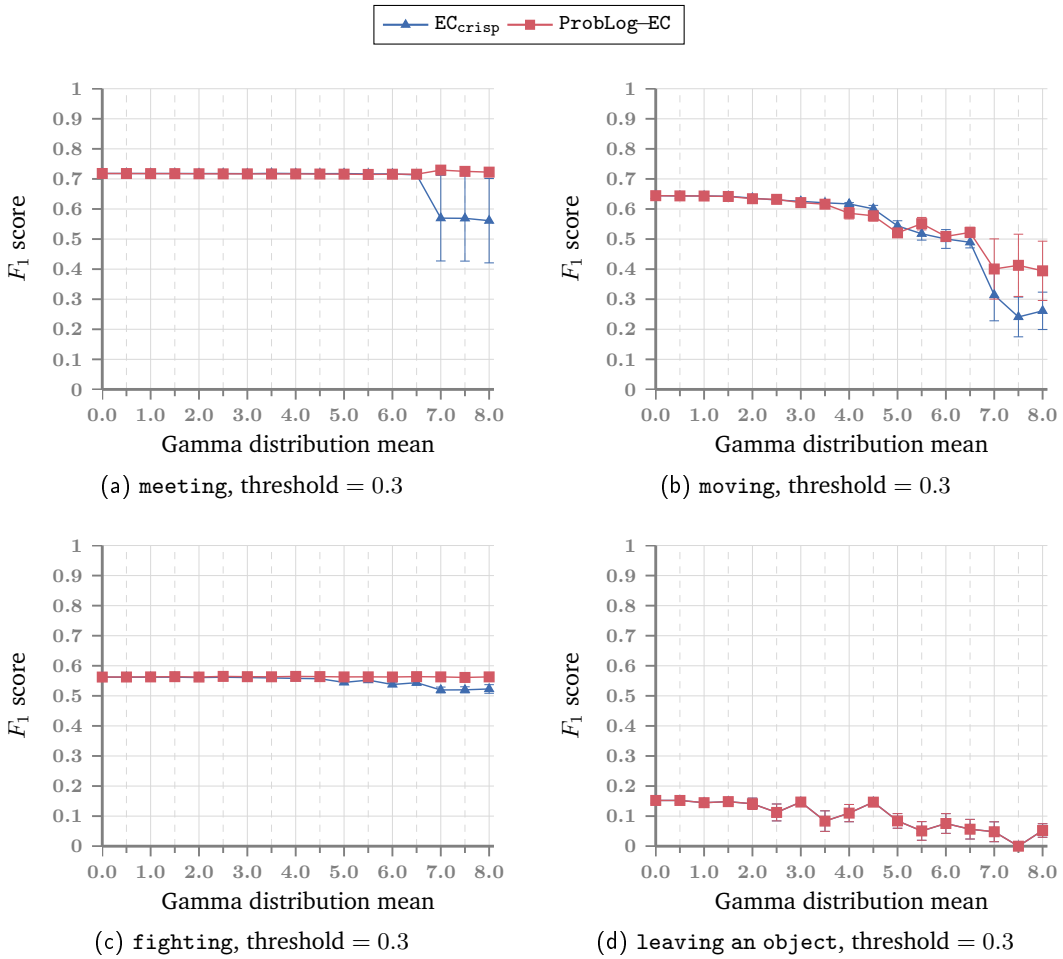
Similar to moving and meeting, ProbLog–EC fares better than EC_{crisp} in fighting (see Figure 6.2(c)). For high levels of noise, i.e., high Gamma mean values, EC_{crisp} has trouble initiating fighting (see rule (5.29)), whereas ProbLog–EC uses repeated initiation to surpass the 0.5 recognition threshold. ProbLog–EC manages to surpass this threshold despite the relatively short duration of fights, compared to other CE.

The leaving_object activity is an interesting special case, owing to the fact that this CE is recognised through a single initiation (see Section 5.3.2). In the dataset, a person leaves an object after a few frames in which the person is walking. In rare situations it is possible that either EC_{crisp} or ProbLog–EC miss the recognition of person due to noise and, consequently, the recognition of leaving_object. Such cases did not arise in our *smooth* noise experiments. In these experiments, ProbLog–EC and EC_{crisp} are equally accurate (see Figure 6.2(d)). When person is recognised by EC_{crisp}, it is recognised with a sufficiently high probability by ProbLog–EC and vice versa.

CEs in the dataset are usually terminated when an entity disappears or when people walk away from each other. In the latter case, the probability of the CE termination depends

solely on that of `walking` (e.g., rule (5.23)). In general, probabilistic terminations are similar to probabilistic initiations. When a termination condition is repeatedly fired with probabilities below 0.5 then ProbLog–EC eventually stops recognising the CE, whereas EC_{crisp} does not, producing false positives.

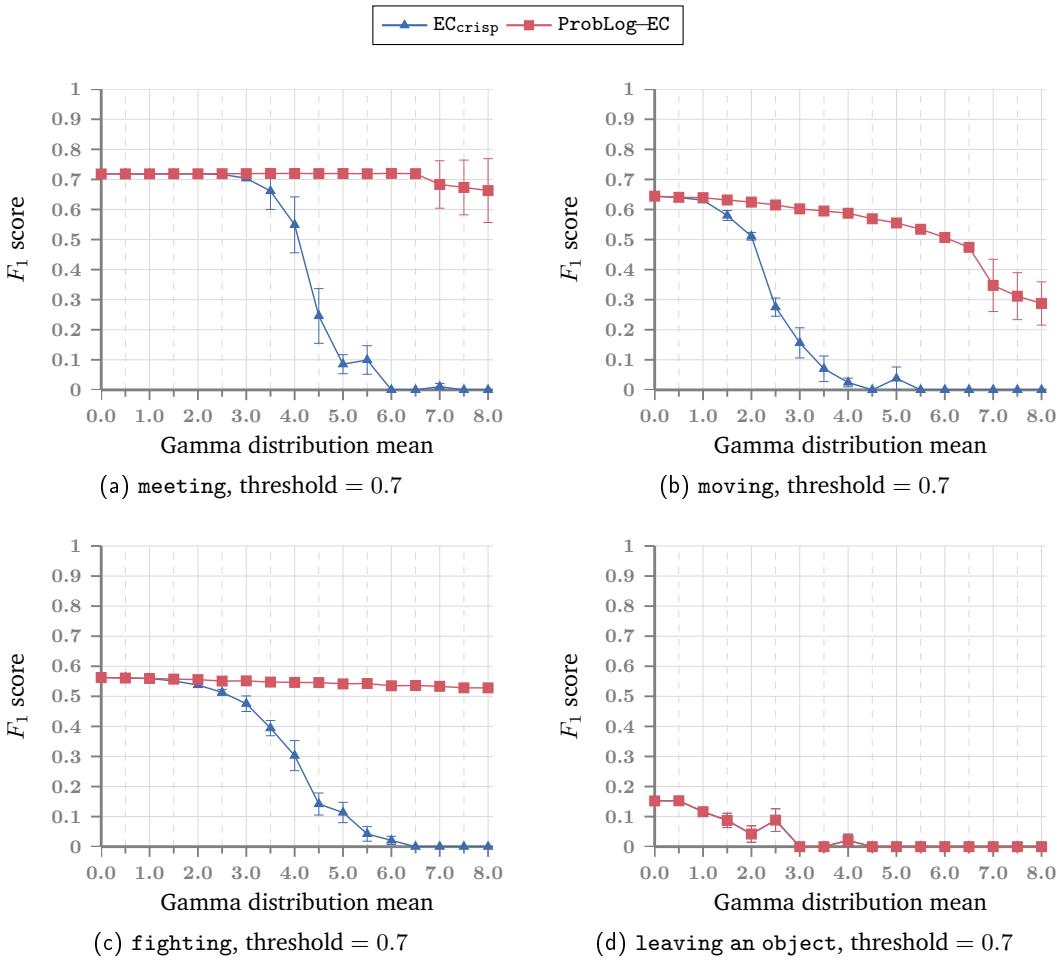
Figures 6.3 and 6.4 compare the recognition accuracy of EC_{crisp} and ProbLog–EC using 0.3 and 0.7 thresholds, respectively. ProbLog–EC is only slightly affected by the threshold change. This is due to the fact that the probabilities of most recognised CEs in ProbLog–EC are either very small (< 0.3) or very high (> 0.7). On the other hand, the recognition accuracy of EC_{crisp} is highly affected by the choice of threshold: the lower the threshold the better the accuracy of EC_{crisp}. This is expected given the nature of the *smooth* noise experiments. As the threshold decreases, the input of EC_{crisp} gets closer to the original, noise-free dataset.



(a) `meeting`, threshold $= 0.3$ (b) `moving`, threshold $= 0.3$

(c) `fighting`, threshold $= 0.3$ (d) `leaving an object`, threshold $= 0.3$

FIGURE 6.3: EC_{crisp} and ProbLog–EC F-measure per Gamma mean value under *smooth* noise, with recognition threshold 0.3.



(a) `meeting`, threshold = 0.7

(b) `moving`, threshold = 0.7

(c) `fighting`, threshold = 0.7

(d) `leaving an object`, threshold = 0.7

FIGURE 6.4: EC_{crisp} and ProbLog–EC F-measure per Gamma mean value under *smooth* noise, with recognition threshold 0.7.

6.2.2 Experiments With Intermediate Noise

Figure 6.5 compares the recognition accuracy of EC_{crisp} and ProbLog–EC under *intermediate* noise using a 0.5 threshold. Overall, attaching probabilities to the coordinates and orientation, in addition to SDEs, leads to accuracy of both EC_{crisp} and ProbLog–EC. Compared to *smooth* noise, the difference between ProbLog–EC and EC_{crisp} in `meeting` increases (see Figure 6.5(a)). In addition to losing the `active` and `inactive` SDEs due to noise, EC_{crisp} now has trouble proving that entities are `close`, because under *intermediate* noise we also remove coordinate-related information, required to compute the distance between two entities. Thus, even in cases where the `active` or `inactive` SDEs are present and indicate that the relevant frame might be an initiation condition for `meeting`, EC_{crisp} is unable to prove that the two entities are `close` enough to initiate the CE. ProbLog–EC, on the other hand, uses repeated initiation to eventually break the 0.5 barrier and produce recognitions.

Concerning `moving` (see Figure 6.5(b)), EC_{crisp} suffers again from the loss of the `walking` SDE. Under *intermediate* noise, this conclusion is more striking: after Gamma mean 2.5,

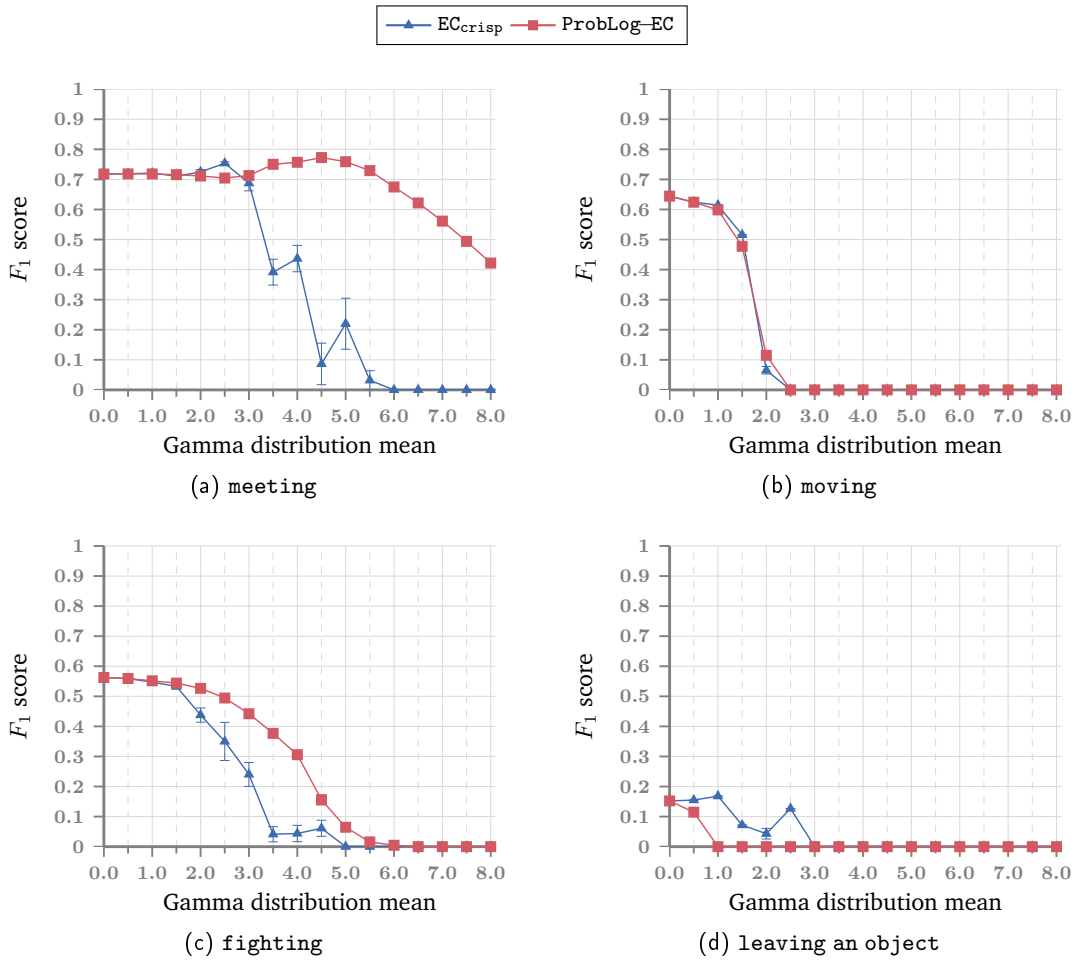


FIGURE 6.5: EC_{crisp} and ProbLog–EC F-measure per Gamma mean value under *intermediate* noise and a 0.5 recognition threshold.

EC_{crisp} is unable to produce a single positive. This is because, in addition to the `walking` SDEs and associated coordinate fluents being erased from the input of EC_{crisp}, orientation fluents are also candidates for removal. As a result, even in cases where both entities potentially involved in a `moving` activity are believed to be `walking` close to each other by the low-level tracking system, the absence of orientation information leads to an inability to initiate `moving`.

What is perhaps more interesting is that ProbLog–EC performs as bad as EC_{crisp} in `moving` (see Figure 6.5(b)). Whenever faced with an initiation condition for `moving`, ProbLog–EC has to calculate the probability of this initiation condition as a product of six probabilities in total. Recall from rule (5.15) that `close` is defined in terms of two coordinate input facts. It therefore contributes as the product of two probabilities. Consequently, ProbLog–EC has trouble surpassing the 0.5 recognition threshold. Even in cases of near-certainty about some conditions of the input, ProbLog's independence assumption leads to very low probability values. For example, consider the two entities id_1 and id_2, whose SDEs (in our case, `walking`) and associated information (coordinates, orientation) are all tracked with a probability of 0.8. Whereas EC_{crisp} will be able to initiate `moving` CE,

since all the facts will make their appearance in the input, ProbLog–EC will produce an initiation condition probability of $(0.8)^6 = 0.262$. Consequently, we will not trust the probability of the relevant holdsAt query. Furthermore, due to the fact that each initiation condition has usually a very low probability, repeated initiation does not exceed the 0.5 threshold.

With respect to fighting (see Figure 6.5(c)), ProbLog–EC outperforms EC_{crisp} for certain Gamma mean values. For high Gamma mean values, ProbLog–EC and EC_{crisp} are equally inaccurate. In the case of *intermediate* noise, there are at least three probabilistic conjuncts per initiation condition. Due to the fact that fights have a relatively short duration (much shorter than meeting, for example), there are not enough initiations to raise ProbLog–EC's probability above the 0.5 threshold in high Gamma mean values.

Regarding leaving_object (see Figure 6.5(d)), EC_{crisp} seems to fair slightly better than ProbLog–EC for low Gamma mean values. Under the *intermediate* noise assumption, ProbLog–EC has to consider more probabilistic conjuncts per initiation condition, due to the presence of close. As a result, and given the single initiation of leaving_object, ProbLog–EC tends to produce probabilities below 0.5 for this CE, even in cases where the SDE probabilities themselves might be above 0.5 and therefore sufficient to allow EC_{crisp} to recognise leaving_object. For higher Gamma mean values, ProbLog–EC and EC_{crisp} are equally inaccurate, because the data required by EC_{crisp} to initiate leaving_object, whether it is the person fluent, the inactive SDE or the coordinate fluents of the entities involved, are missing.

Figure 6.7 compares the recognition accuracy of EC_{crisp} and ProbLog–EC using 0.3 and 0.7 as threshold values. As in the case of the *smooth* noise experiments, EC_{crisp} is affected more than ProbLog–EC by the threshold change — the accuracy of EC_{crisp} improves as the threshold decreases (the lower the threshold, the closer the input of EC_{crisp} to the original, noise-free dataset).

6.2.3 Experiments With Strong Noise

Figure 6.8 compares the recognition accuracy of EC_{crisp} and ProbLog–EC under *strong* noise using a 0.5 threshold. As can be seen from this figure, the recognition accuracy under *strong* noise is very similar to that under *intermediate* noise in the 0.5 threshold. For ProbLog–EC, we need a series of spurious information to surpass the chosen threshold — the introduction of spurious SDEs in the *strong* noise experiments is not that systematic and does not create problems to ProbLog–EC. EC_{crisp} is very slightly affected by the introduction of spurious SDEs: it now has a number of additional false positives, mainly in the case of moving. Even a single initiation condition fired by a spurious walking SDE and some other SDEs may be enough in EC_{crisp} to create a series of false positives. However, in low Gamma values most of the spurious SDE have low probabilities — below

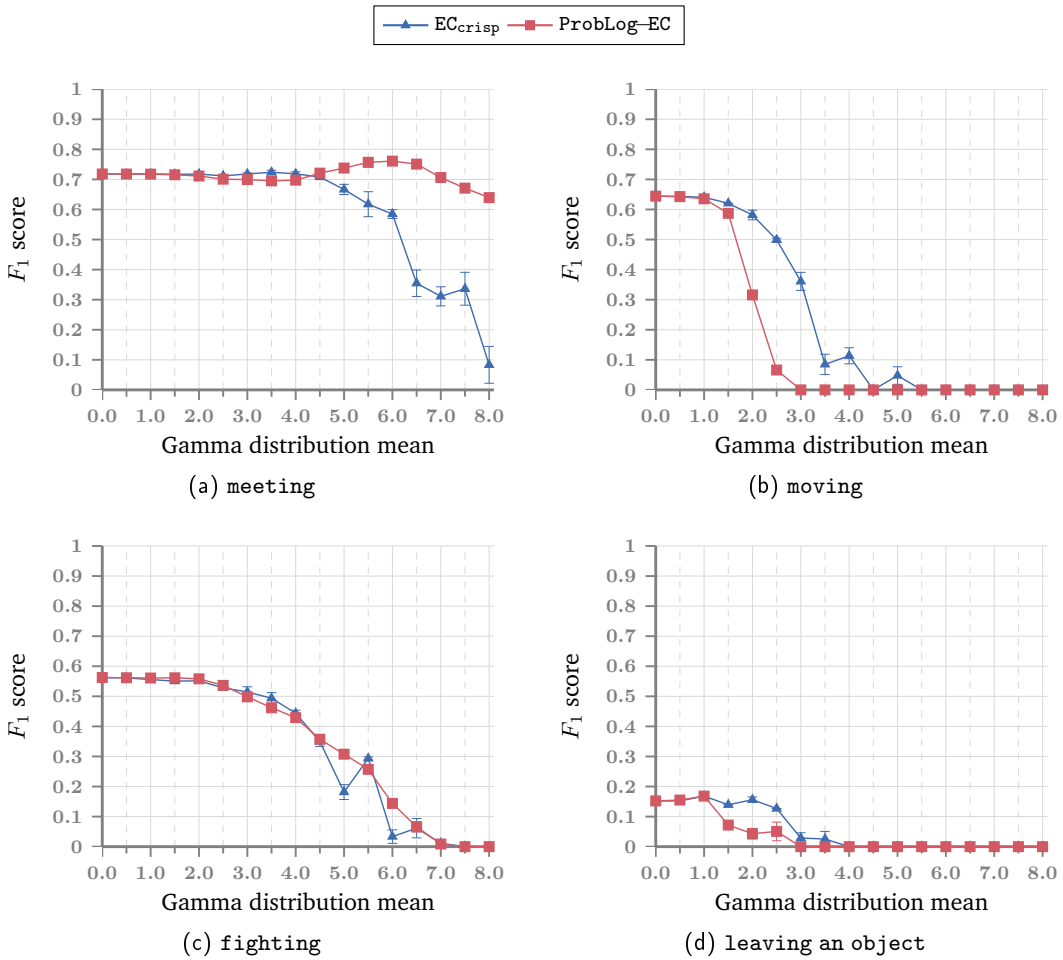


FIGURE 6.6: EC_{crisp} and ProbLog–EC F-measure per Gamma mean value under *intermediate* noise and a recognition threshold 0.3.

the 0.5 threshold — and therefore are not part of the input of EC_{crisp}. Furthermore, in high Gamma values the spurious SDEs cannot initiate a CE because several of the SDEs tend to be deleted from the input of EC_{crisp} as they have low probabilities.

Figure 6.9 compares the accuracy of EC_{crisp} and ProbLog–EC in moving using thresholds of 0.3 and 0.7 (the diagrams for the remaining CEs are omitted as they are similar to those concerning *intermediate* noise). EC_{crisp}, as in the previous experimental settings, is more significantly affected than ProbLog–EC by the threshold change. Unlike the last two experimental settings, reducing the threshold causes problems to EC_{crisp} with respect to moving. With a lower threshold, EC_{crisp} loses fewer SDEs, but at the same time keeps more spurious facts, creating many more false positives. ProbLog–EC, even in the low threshold of 0.3, is almost unaffected by the spurious facts. In the 0.7 threshold EC_{crisp} has a very small number of spurious facts in its input and, therefore, its accuracy is not compromised.

In addition to walking SDEs, we could have added other spurious facts such abrupt, active, or inactive and repeated the experiments. The results would be expected to

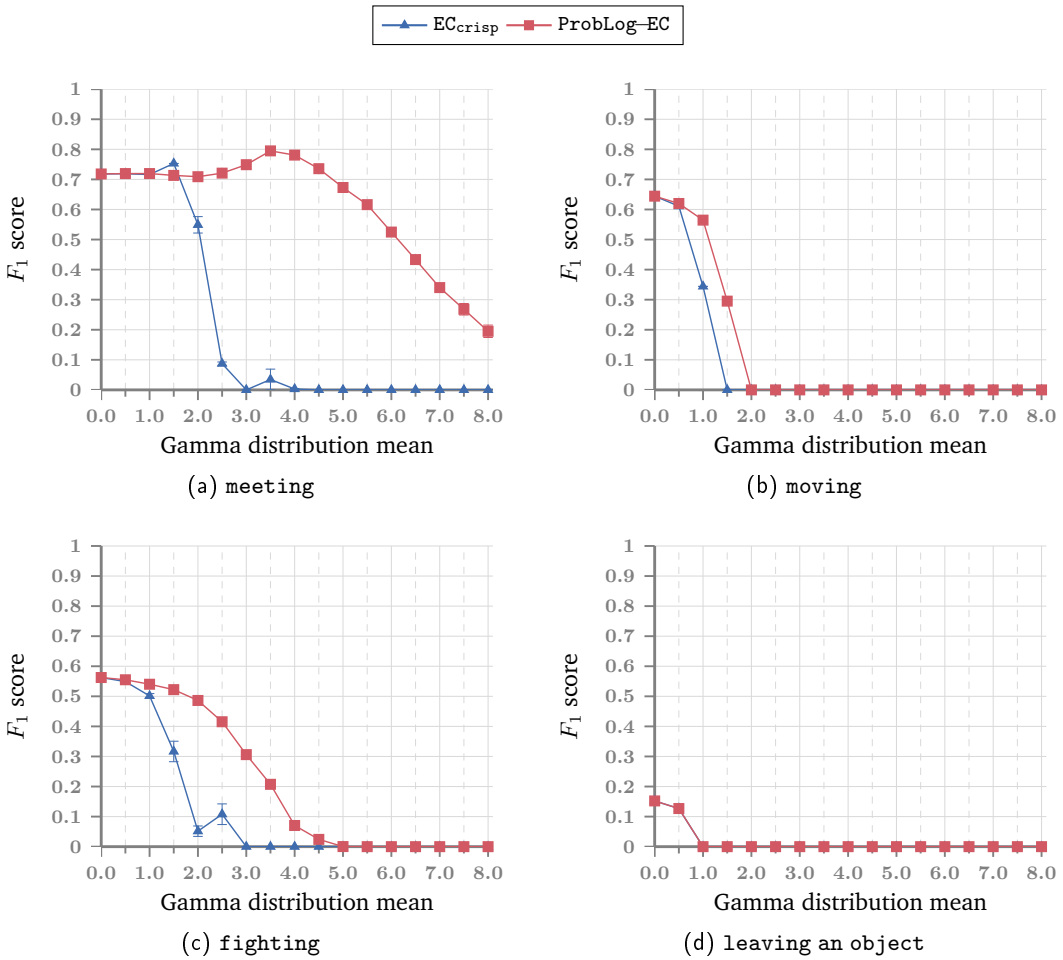


FIGURE 6.7: EC_{crisp} and ProbLog–EC F-measure per Gamma mean value under *intermediate* noise and a recognition threshold 0.7.

be similar to those presented above — for example, spurious `abrupt` SDE would have compromised the accuracy of EC_{crisp} in `fighting`, especially in low thresholds.

6.3 Conclusions

Our experimental evaluation on a benchmark activity recognition dataset showed that ProbLog–EC outperforms EC_{crisp} when:

- a CE has multiple initiations, and
- the CE depends on a small number of probabilistic conjuncts.

ProbLog–EC is more resilient to spurious facts than EC_{crisp}. Even a single such fact may create a series of false positives in EC_{crisp}, whereas this type of noise must be much more systematic to affect ProbLog–EC: to impinge the accuracy of the latter we need a series of spurious facts in order to surpass the recognition threshold.

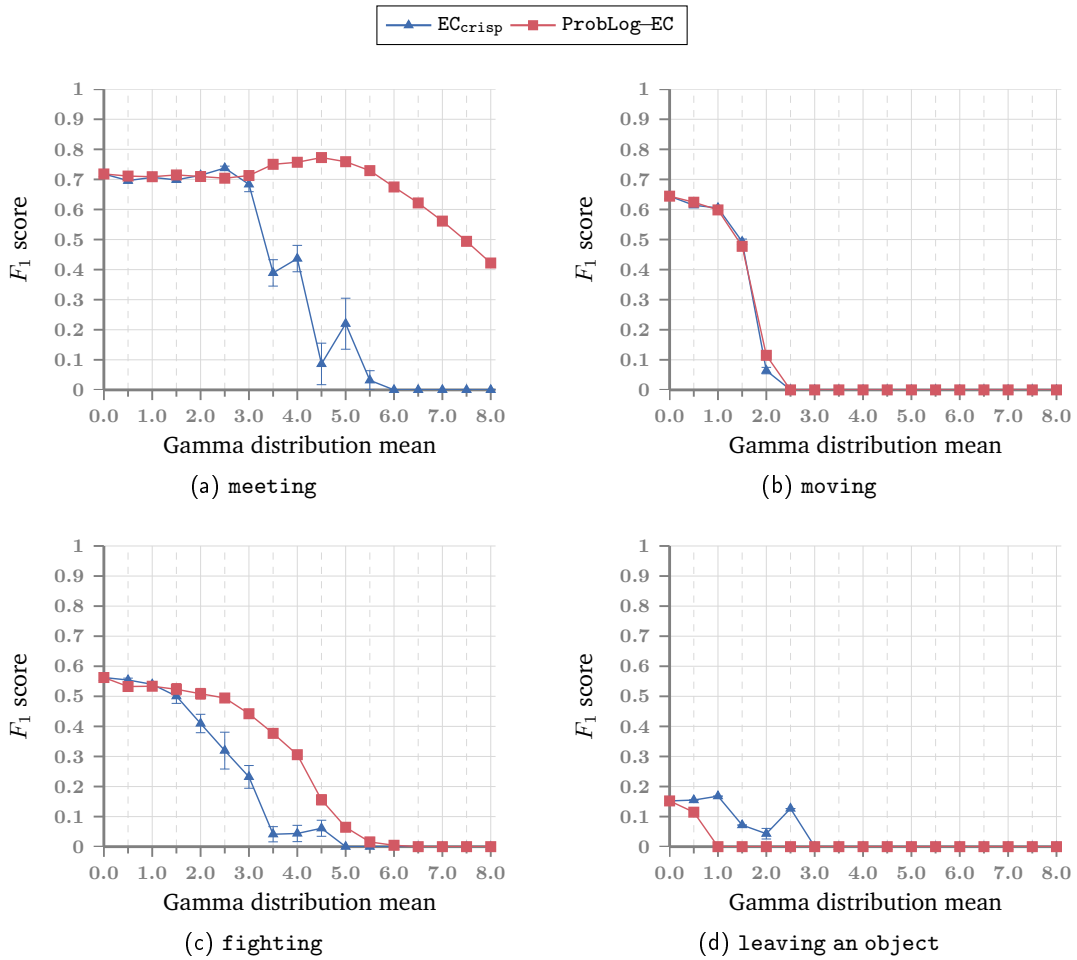
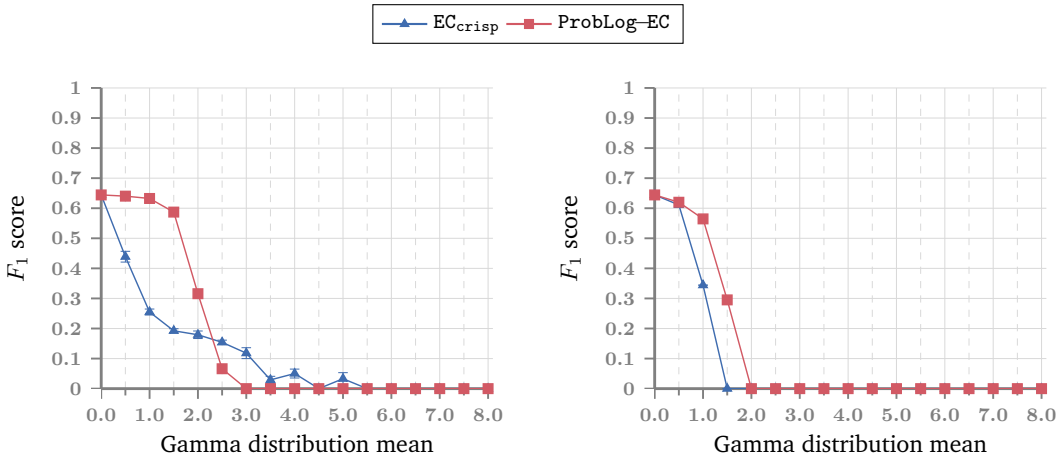


FIGURE 6.8: $\texttt{EC}_{\texttt{crisp}}$ and ProbLog–EC F-measure per Gamma mean value under *strong* noise and a 0.5 recognition threshold.

In the case of a single initiation, there are situations in which ProbLog–EC may fare significantly better than $\texttt{EC}_{\texttt{crisp}}$ and vice versa, but these did not arise in our experiments. On average, ProbLog–EC is expected to have similar performance to $\texttt{EC}_{\texttt{crisp}}$ on CE with a single initiation, in the case of a small number of probabilistic conjuncts, while $\texttt{EC}_{\texttt{crisp}}$ is likely to perform better in the case of a large number of such conjuncts.

The experimental results concerning one or more terminations are similar to those for one or more initiations. For example, ProbLog–EC outperforms $\texttt{EC}_{\texttt{crisp}}$ in the case of CE with multiple terminations that depend on a small number of probabilistic conjuncts. The repeated terminations allow ProbLog–EC to stop recognising a CE when $\texttt{EC}_{\texttt{crisp}}$ continues the recognition producing false positives.

ProbLog–EC makes an independence assumption about input SDEs and thus the product of the probabilities of many probabilistic conjuncts may be very small, even if the probability of each individual conjunct is high. The greater the number of probabilistic conjuncts, the more initiations ProbLog–EC requires to surpass the chosen recognition threshold.



(a) moving, threshold = 0.3 (b) moving, threshold = 0.7

FIGURE 6.9: EC$_{crisp}$ and ProbLog–EC F-measure per Gamma mean value under *strong* noise in moving and different recognition thresholds.

7 Conclusions and Future Work

"Science never solves a problem without creating ten more."
— George Bernard Shaw

In this thesis we focused on symbolic event recognition under situations where uncertainty exists . In Section 2.4.1, we argue that action formalisms can represent the dynamic aspects of event recognition applications and provide an event description language with well-defined formal semantics. Furthermore, in Section 2.4.2 we observed that symbolic approaches can compactly represent composite events (CE) and domain knowledge. Under uncertainty, however, the performance of symbolic methods may be seriously compromised. For that reason, numerous of probabilistic and symbolic methods for event recognition have been proposed in the literature (see Section 2.4.3). In order to address these issues in this thesis we have developed two symbolic methods that combine a discrete variant of the Event Calculus formalism with probabilistic modelling.

7.1 Conclusions

In Chapter 3 we presented the MLN–EC method, in order to address the issue of event recognition using imperfect CE definitions. MLN–EC uses Markov Logic Networks (MLNs) to achieve a probabilistic Event Calculus dialect for event recognition. For reasons of efficiency, we placed emphasis on transforming the probabilistic Event Calculus into compact Markov networks. Furthermore, MLN–EC supports flexible inertia of CEs, ranging from deterministic to probabilistic. The performance of MLN–EC in event recognition is demonstrated in Chapter 4 through a series of experiments on a publicly available benchmark dataset. MLN–EC has been compared with a method that uses crisp Event Calculus and with a purely data-driven probabilistic method that employs linear-chain Conditional Random Fields (l–CRF). In the experiments where the data were sufficient for the data-driven method to estimate the model parameters, MLN–EC outperformed its crisp equivalent and achieved comparable performance to the data-driven method. In

contrast to its crisp equivalent, CEs in MLN–EC are associated with weight values, indicating a degree of confidence. The probabilistic modelling of MLN–EC managed to correct many of the recognition errors that are caused by imperfect definitions of CEs. Furthermore, in the experiments where the input stream was incomplete, MLN–EC performed considerably better than the data-driven l–CRF method. MLN–EC retains its performance when data is incomplete, by exploiting the given domain knowledge and the modelling of inertia by the Event Calculus formalism.

In order to deal with the uncertainty of the input events, we presented the ProbLog–EC method in Chapter 5. Similar to the MLN–EC, the method combines a discrete Event Calculus dialect with probabilistic logic programming. In contrast to MLN–EC, ProbLog–EC can directly exploit input SDEs that are associated with probabilities, indicating a degree of confidence in their recognition. Therefore, the uncertainty of input SDEs is directly propagated to the recognition of CEs. While MLN–EC does not make any independence assumption on the input variables, in ProbLog–EC input SDEs are assumed to be independent. On the other hand, in MLN–EC all input SDEs are represented by Boolean random variables and thus input uncertainty cannot propagated into the event recognition method. Another difference, compared to MLN–EC, is that the rules in the ProbLog–EC knowledge base are expressed only as hard constraints and thus all CE definitions are defined with absolute certainty. Furthermore, the inertia is modelled by the closed-world semantics of logic programming and is restricted to be deterministic.

In the experimental evaluation of ProbLog–EC (see Chapter 6), we have observed that when a CE has multiple initiations and the CE depends on a small number of probabilistic conjuncts, ProbLog–EC outperforms its crisp equivalent.

In summary the proposed probabilistic event recognition methods exhibit the following characteristics:

- Event recognition using a logic-based representation with formal and declarative semantics and probabilistic modelling (MLN–EC and ProbLog–EC).
- Efficient representation of the probabilistic Event Calculus accompanied with a knowledge transformation that produces compact Markov Networks (MLN–EC).
- Flexible inertia of CEs, ranging from deterministic to probabilistic (MLN–EC).
- Event recognition using imperfect knowledge (MLN–EC), incomplete input event stream (MLN–EC) and uncertain input events (ProbLog–EC).

7.2 Future Work

There are several directions in which we would like to extend our work. The main ones are presented below:

Extended Knowledge Representation

The temporal model in both methods presented in this work is linear and discrete, based on time-points. Many event recognition methods support constraints over intervals, using pre-processing techniques (e.g., Morariu and Davis [2011]) or by employing different representation and inference methods (e.g., Brendel et al. [2011] and Selman et al. [2011]). One possible improvement is to extend the temporal model with intervals, in order to support constraints over interval relations.

In addition to event recognition, in many applications forecasting of events is required. The aim of event forecasting is to predict which CEs are likely to happen in future. One solution is to employ a forward-branching time model for future events, e.g., Situation Calculus [Reiter, 2001] and Branching Event Calculus [Mueller, 2007], in order to represent and reason about hypothetical future CEs, given the observed time-line of input SDEs.

The two probabilistic and symbolic methods that have been developed in this thesis assume that input can only be symbolic. For example, the spatial constraints of distance between two persons are discretised using some specified threshold values. The threshold values require off-line analysis, which can be tedious and error-prone. Therefore, it would be ideal to directly support quantitative information in the model (e.g., Hybrid MLNs [Sadilek and Kautz, 2012; Wang and Domingos, 2008], Hybrid ProbLog [Gutmann et al., 2010] and Probabilistic Soft Logic [Bröcheler et al., 2010]). Using real-valued functions we can directly affect the confidence value of CE definitions — e.g., to define that the closer are two persons, are the more confident we are for the meeting activity.

Other variants of the Event Calculus (e.g., Mueller [2008]) provide additional domain-independent axioms, in order to support discrete change of fluents, control of when a fluent is not subject to inertia, etc. We would like to investigate the effects of introducing these additional features in the probabilistic environment of the proposed methods.

Scalable Inference

In terms of inference there are various improvements that we would like to explore. The preprocessing step that we perform in `MLN-EC` (see Section 3.3) reduces the complexity and the size of the resulting Markov network. Further simplifications can be performed using lifted unit propagation (e.g., den Broeck et al. [2011]; Papai et al. [2011]), in order to reduce the number of first-order clauses in the knowledge base. In particular, hard-constrainted unit clauses (such as the initial state of fluents) can be propagated to the knowledge base and create tautological clauses or create other unit clauses.

`MLN-EC` deals with the uncertainty in CE definitions, while `ProbLog-EC` deals with the uncertainty in input SDEs. We would like to combine these features into a single model.

By introducing auxiliary probabilistic facts for each rule in CE definitions, ProbLog–EC may model the uncertainty of CEs. One issue with that approach is that the probabilistic conjuncts in the definition will increase, leading to low probability values (see Chapter 6). On the other hand, according to Jain and Beetz [2010] there are two directions for supporting probabilistic SDEs in MLN–EC. First, we can employ *virtual evidence* (e.g., Li [2009]; Tran and Davis [2008]), in which each probabilistic SDE is represented by an auxiliary Boolean random variable in the model and a utility unit clause with weight equal to the log-odds of its probability. Virtual evidence, however, assumes that SDEs are independent. Alternatively, by modifying the inference algorithm [Jain and Beetz, 2010], MLN–EC can be extended using *soft evidence* and directly support probabilistic SDEs without an independence assumption.

Event recognition in both MLN–EC and ProbLog–EC is performed offline. On-line event recognition requires dynamic inference, during the evolution of input SDEs. Although exact inference in MLN is intractable (see Section 2.3.1), in Section 4.2 we showed that approximate inference times in MLN–EC are faster than real-time in activity recognition. Therefore, approximate inference could be useful for on-line event recognition. Along those lines, Geier and Biundo [2011] perform on-line approximate inference in MLNs, using the marginal probabilities of ground atoms of a past time step to construct weighted formulas to be included in the network at the next time step. However, the approximate inference propagates a degree of error which may become large after a few time steps. A different direction is to employ belief-propagation based inference. For example, Nath and Domingos [2010] perform incremental network construction and inference that reuses previously sent messages. Additionally, Kersting et al. [2009] reduce the inference cost by exploiting the network symmetries. However, belief-propagation in MLNs may fail to converge to the correct solution [Poon and Domingos, 2006]. Yet another direction is to employ the model of slice-normalised dynamic MLN of Papai et al. [2012], in which the knowledge base is transformed into a network in which the clique potential between adjacent slices (e.g., the inertia rules (3.24) and (3.25)) is always normalised.

As outlined in Section 2.3.2, in applications with large knowledge bases building a Binary Decision Diagram for all proofs may become intractable. Therefore, more efficient inference methods are needed for on-line inference in ProbLog–EC. Examples are the k-best of Kimmig et al. [2011], k-optimal of Renkens et al. [2012] or by performing MLN inference using the transformation of Fierens et al. [2011].

In situations where we prefer to query only for a fraction of time-points and CEs, lifted inference techniques may be more suitable and efficient, as they avoid grounding the entire knowledge base. We are planning to investigate lifted inference methods for event recognition — e.g., Apsel and Brafman [2012]; den Broeck et al. [2014, 2011]; Gogate and Domingos [2011]; Sarkhel and Gogate [2013]; Sarkhel et al. [2014].

Machine Learning

In Chapter 4, we evaluated the performance of `MLN-EC` using different types of inertia, which proved useful under various experimental conditions. We would like to extend our method to automatically soften the right subset of inertia axioms during weight learning.

Both `MLN-EC` and `ProbLog-EC` assume that CE definitions are known, e.g., given by domain experts. Although CE definitions are usually simple common sense rules, in many applications their manual specification may be a tedious process. An important research direction is to automatically extract or refine CE definitions from training data, using structure learning techniques. As in Chapter 4, the training set for such methods would be composed of SDEs (i.e., ground `happens` predicates) and their annotation of CEs (i.e., ground `holdsAt` predicates). Furthermore, the knowledge transformation procedure presented in Section 3.3.2 eliminates the `initiatedAt` and `terminatedAt` predicates, allowing the application of fully supervised learning methods. If such transformation of the domain knowledge is not desired and event definitions need to be learned in terms of initiation and termination conditions, initiation and termination predicates would also need to be observable in the training set [Artikis et al., 2010b]. In that case, structure learning would require semi-supervised techniques that combine abduction and induction. Abductive reasoning techniques [Kakas, 2010; Kakas and Flach, 2009] can fill the missing observations in the training set by generating a set of explanations about initiation and termination predicates — see, for example, the works of Blythe et al. [2011]; Corapi et al. [2009]; Ray [2009]; Ray et al. [2003]; Singla and Mooney [2011] and Katzouris et al. [2014].

Bibliography

Akman, V., Selim T. Erdoğan, J. L., Lifschitz, V., and Turner, H. (2004). Representing the Zoo World and the Traffic World in the language of the Causal Calculator. *Artificial Intelligence*, 153(1):105–140.

Albanese, M., Chellappa, R., Cuntoor, N., Moscato, V., Picariello, A., Subrahmanian, V. S., and Udrea, O. (2010). PADS: A Probabilistic Activity Detection Framework for Video Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2246–2261.

Albanese, M., Chellappa, R., Cuntoor, N. P., Moscato, V., Picariello, A., Subrahmanian, V. S., and Udrea, O. (2008). A Constrained Probabilistic Petri Net Framework for Human Activity Detection in Video. *IEEE Transactions on Multimedia*, 10(6):982–996.

Albanese, M., Molinaro, C., Persia, F., Picariello, A., and Subrahmanian, V. S. (2011). Finding "Unexplained" Activities in Video. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1628–1634.

Albanese, M., Moscato, V., Picariello, A., Subrahmanian, V. S., and Udrea, O. (2007). Detecting Stochastically Scheduled Activities in Video. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1802–1807.

Allen, J. F. (1983). Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843.

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

Artikis, A. and Paliouras, G. (2009). Behaviour recognition using the event calculus. In Iliadis, L. S., Maglogiannis, I., Tsoumakas, G., Vlahavas, I. P., and Bramer, M., editors, *Proceedings of the 5th IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI)*, IFIP Advances in Information and Communication Technology, pages 469–478. Springer.

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

Artikis, A., Sergot, M. J., and Paliouras, G. (2012a). Run-time Composite Event Recognition. In Bry, F., Paschke, A., Eugster, P. T., Fetzer, C., and Behrend, A., editors, *Proceedings of the Sixth ACM International Conference on Distributed Event-Based Systems (DEBS)*, pages 69–80. ACM.

Artikis, A., Skarlatidis, A., and Paliouras, G. (2010b). Behaviour Recognition from Video Content: a Logic Programming Approach. *International Journal on Artificial Intelligence Tools (JAIT)*, 19(2):193–209.

Artikis, A., Skarlatidis, A., Portet, F., and Paliouras, G. (2012b). Logic-based event recognition. *Knowledge Engineering Review*, 27(4):469–506.

Bacchus, F., Halpern, J. Y., and Levesque, H. J. (1995). Reasoning about Noisy Sensors in the Situation Calculus. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1933–1940. Morgan Kaufmann.

Bacchus, F., Halpern, J. Y., and Levesque, H. J. (1999). Reasoning about Noisy Sensors and Effectors in the Situation Calculus. *Artificial Intelligence*, 111(1-2):171–208.

Batsakis, S. and Petrakis, E. G. M. (2011). SOWL: A Framework for Handling Spatiotemporal Information in OWL 2.0. In Bassiliades, N., Governatori, G., and Paschke, A., editors, *Rule-Based Reasoning, Programming, and Applications - 5th International Symposium (RuleML)*, volume 6826 of *Lecture Notes in Computer Science*, pages 242–249. Springer.

Biba, M., Xhafa, F., Esposito, F., and Ferilli, S. (2011). 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.

Biswas, R., Thrun, S., and Fujimura, K. (2007). Recognizing Activities with Multiple Cues. In *Proceedings of the 2nd Workshop on Human Motion - Understanding, Modeling, Capture and Animation*, Lecture Notes in Computer Science, pages 255–270. Springer.

Blockeel, H. (2011). Statistical Relational Learning. In Bianchini, M., Maggini, M., and Jain, L., editors, *Handbook on Neural Information Processing*. Springer.

Blythe, J., Hobbs, J. R., Domingos, P., Kate, R. J., and Mooney, R. J. (2011). Implementing Weighted Abduction in Markov Logic. In *In Proceedings of the International Conference on Computational Semantics*, pages 55–64, Oxford, England.

Boros, E. and Hammer, P. L. (2002). Pseudo-Boolean optimization. *Discrete Applied Mathematics*, 123(1–3):155–225.

Boutilier, C., Reiter, R., and Price, B. (2001). Symbolic Dynamic Programming for First-Order MDPs. In Nebel, B., editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 690–700. Morgan Kaufmann.

Brand, M., Oliver, N., and Pentland, A. (1997). 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.

Brendel, W., Fern, A., and Todorovic, S. (2011). Probabilistic Event Logic for Interval-based Event Recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3329–3336. IEEE Computer Society.

Bröcheler, M., Mihalkova, L., and Getoor, L. (2010). Probabilistic similarity logic. In Grünwald, P. and Spirtes, P., editors, *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 73–82. AUAI Press.

Bryant, R. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691.

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

Byrd, R. H., Nocedal, J., and Schnabel, R. B. (1994). Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1):129–156.

Callens, L., Carrault, G., Cordier, M.-O., Fromont, E., Portet, F., and Quiniou, R. (2008). Intelligent adaptive monitoring for cardiac surveillance. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI)*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 653–657. IOS Press.

Casella, G. and George, E. I. (1992). Explaining the Gibbs Sampler. *The American Statistician*, 46(3):167–174.

Castel, C., Chaudron, L., and Tessier, C. (1996). What is going on? a high level interpretation of sequences of images. In *ECCV Workshop on Conceptual Descriptions from Images*, pages 13–27. Citeseer.

Chen, L., Nugent, C., Mulvenna, M., Finlay, D., Hong, X., and Poland, M. (2008). Using Event Calculus for Behaviour Reasoning and Assistance in a Smart Home. In Helal, S., Mitra, S., Wong, J., Chang, C., and Mokhtari, M., editors, *Smart Homes and Health Telematics*, volume 5120 of *Lecture Notes in Computer Science*, pages 81–89. Springer Berlin Heidelberg.

Chittaro, L. and Montanari, A. (2000). Temporal Representation and Reasoning in Artificial Intelligence: Issues and Approaches. *Annals of Mathematics and Artificial Intelligence*, 28(1):47–106.

Choppy, C., Bertrand, O., and Carle, P. (2009). Coloured Petri Nets for Chronicle Recognition. In *Proceedings of the 14th International Conference on Reliable Software*

Technologies - Ada-Europe, volume 5570 of *Lecture Notes in Computer Science*, pages 266–281. Springer.

Collins, M. (2002). 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.

Corapi, D., Ray, O., Russo, A., Bandara, A. K., and Lupu, E. C. (2009). Learning Rules from User Behaviour. In Iliadis, L. S., Maglogiannis, I., Tsoumakas, G., Vlahavas, I. P., and Bramer, M., editors, *Proceedings of the 5TH IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI)*, volume 296 of *IFIP Advances in Information and Communication Technology*, pages 459–468. Springer.

Craven, R. (2006). *Execution mechanisms for the action language C+*. PhD thesis, University of London.

Cugola, G. and Margara, A. (2010). TESLA: A Formally Defined Event Specification Language. In *Proceedings of the 4th ACM International Conference on Distributed Event-Based Systems*, DEBS '10, pages 50–61, New York, NY, USA. ACM.

Cugola, G., Margara, A., Matteucci, M., and Tamburrelli, G. (2014). Introducing uncertainty in complex event processing: model, implementation, and validation. *Computing*, pages 1–42.

Culotta, A. and McCallum, A. (2004). Confidence estimation for information extraction. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 109–112. Association for Computational Linguistics.

Cussens, J. (1999). Loglinear models for first-order probabilistic reasoning. In *Proceedings of the 15th conference on Uncertainty in Artificial Intelligence*, pages 126–133. Morgan Kaufmann Publishers Inc.

Damlen, P., Wakefield, J., and Walker, S. (1999). 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.

David, R. and Alla, H. (1994). Petri Nets for Modeling of Dynamic Systems: A Survey. *Automatica*, 30(2):175–202.

de Raedt, L. and Kersting, K. (2008). Probabilistic Inductive Logic Programming. In *Probabilistic Inductive Logic Programming - Theory and Applications*, volume 4911 of *Lecture Notes in Computer Science*, pages 1–27. Springer.

de Raedt, L. and Kersting, K. (2010). Statistical Relational Learning. In Sammut, C. and Webb, G. I., editors, *Encyclopedia of Machine Learning*, pages 916–924. Springer.

de Raedt, L., Kimmig, A., and Toivonen, H. (2007). ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2462–2467.

de Salvo Braz, R., Amir, E., and Roth, D. (2008). A Survey of First-Order Probabilistic Models. In *Innovations in Bayesian Networks: Theory and Applications*, volume 156 of *Studies in Computational Intelligence*, pages 289–317. Springer.

den Broeck, G. V., Meert, W., and Darwiche, A. (2014). Skolemization for Weighted First-Order Model Counting. In Baral, C., De Giacomo, G., and Eiter, T., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference (KR)*. AAAI Press.

den Broeck, G. V., Taghipour, N., Meert, W., Davis, J., and de Raedt, L. (2011). 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.

Doherty, P., Gustafsson, J., Karlsson, L., and Kvarnström, J. (1998). TAL: Temporal Action Logics Language Specification and Tutorial. *Electronic Transactions on Artificial Intelligence*, 2(3–4):273–306.

Doherty, P., Lukaszewicz, W., and Szalas, A. (1997). Computing Circumscription Revisited: A Reduction Algorithm. *Journal of Automated Reasoning*, 18(3):297–336.

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

Dousson, C. (1996). Alarm Driven Supervision for Telecommunication Network: II-Online Chronicle Recognition. *Annals of Telecommunications*, 51(9):501–508.

Dousson, C., Gaborit, P., and Ghallab, M. (1993). Situation recognition: Representation and algorithms. In Bajcsy, R., editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 166–174. Morgan Kaufmann.

Dousson, C. and Maigat, P. L. (2007). Chronicle Recognition Improvement Using Temporal Focusing and Hierarchization. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 324–329.

Eiter, T. and Lukasiewicz, T. (2003). Probabilistic Reasoning about Actions in Nonmonotonic Causal Theories. In Meek, C. and Kjærulff, U., editors, *Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 192–199. Morgan Kaufmann.

Etzion, O. and Niblett, P. (2010). *Event Processing in Action*. Manning Publications Company.

Fagin, R. (1996). Combining fuzzy information from multiple systems. In *In Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 216–226. ACM Press.

Fierens, D., Van den Broeck, G., Thon, I., Gutmann, B., and De Raedt, L. (2011). Inference in probabilistic logic programs using weighted CNF's. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 211–220.

Filippaki, C., Antoniou, G., and Tsamardinos, I. (2011). Using Constraint Optimization for Conflict Resolution and Detail Control in Activity Recognition. In *Proceedings of the 2nd International Joint Conference on Ambient Intelligence (AmI)*, volume 7040 of *Lecture Notes in Computer Science*, pages 51–60. Springer.

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

Geier, T. and Biundo, S. (2011). Approximate Online Inference for Dynamic Markov Logic Networks. In *ICTAI*, pages 764–768. IEEE.

Getoor, L. (2001). *Learning Statistical Models from Relational Data*. PhD thesis, Stanford.

Getoor, L. (2003). Learning Structure From Statistical Models. *IEEE Data Eng. Bull.*, 26(3):11–18.

Getoor, L. (2006). An Introduction to Probabilistic Graphical Models for Relational Data. *IEEE Data Eng. Bull.*, 29(1):32–39.

Getoor, L. and Taskar, B. (2007). *Introduction to statistical relational learning*. MIT press.

Ghanem, N., DeMenthon, D., Doermann, D., and Davis, L. (2004). Representation and Recognition of Events in Surveillance Video Using Petri Nets. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on*, pages 112–112.

Ghezzi, C., Mandrioli, D., and Morzenti, A. (1990). TRIO: A Logic Language for Executable Specifications of Real-time Systems. *Journal of Systems and Software*, 12(2):107–123.

Ginsberg, M. L. (1988). Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316.

Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., and Turner, H. (2004). Nonmonotonic Causal Theories. *Artificial Intelligence*, 153(1):49–104.

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

Gong, S. and Xiang, T. (2003). 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.

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

Gonzalez, R. C. and Woods, R. E. (2007). *Digital Image Processing*. Prentice Hall, 3nd edition.

Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., and Sattler, U. (2008). Owl 2: The next step for owl. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309–322. Semantic Web Challenge 2006/2007.

Gutmann, B., Jaeger, M., and de Raedt, L. (2010). Extending ProbLog with Continuous Distributions. In Frasconi, P. and Lisi, F. A., editors, *ILP*, volume 6489 of *Lecture Notes in Computer Science*, pages 76–91. Springer Berlin Heidelberg.

Hajishirzi, H. and Amir, E. (2008). Sampling First Order Logical Particles. In *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence (UAI), Helsinki, Finland*, pages 248–255. AUAI Press.

Hajishirzi, H. and Amir, E. (2010). Reasoning about deterministic actions with probabilistic prior and application to stochastic filtering. In Lin, F., Sattler, U., and Truszczynski, M., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference (KR)*. AAAI Press.

Helaoui, R., Niepert, M., and Stuckenschmidt, H. (2011). Recognizing Interleaved and Concurrent Activities: A Statistical-Relational Approach. In *Proceedings of the 9th Annual International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9. IEEE Computer Society.

Helaoui, R., Riboni, D., and Stuckenschmidt, H. (2013). A Probabilistic Ontological Framework for the Recognition of Multilevel Human Activities. In Mattern, F., Santini, S., Canny, J. F., Langheinrich, M., and Rekimoto, J., editors, *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, pages 345–354. ACM.

Hölldobler, S., Karabaev, E., and Skvortsova, O. (2006). FLUCAP: a heuristic search planner for first-order MDPs. *Journal of Artificial Intelligence Research (JAIR)*, 27(1):419–439.

Hongeng, S. and Nevatia, R. (2003). Large-Scale Event Detection Using Semi-Hidden Markov Models. In *Proceedings of the 9th International Conference on Computer Vision (ICCV)*, volume 2, pages 1455–1462. IEEE Computer Society.

Hoos, H. H. and Stützle, T. (2004). *Stochastic Local Search: Foundations & Applications*. Elsevier / Morgan Kaufmann.

Horrocks, I. (2005). OWL: A Description Logic Based Ontology Language. In van Beek, P., editor, *11th International Conference on the Principles and Practice of Constraint Programming (CP)*, volume 3709 of *Lecture Notes in Computer Science*, pages 5–8. Springer Berlin Heidelberg.

Huynh, T. N. and Mooney, R. J. (2009). 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.

Huynh, T. N. and Mooney, R. J. (2011). 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.

Jain, D. and Beetz, M. (2010). Soft Evidential Update via Markov Chain Monte Carlo Inference. In *Proceedings of the 33rd Annual German Conference on AI (KI)*, volume 6359 of *Lecture Notes in Computer Science*, pages 280–290. Springer.

Joachims, T., Finley, T., and Yu, C.-N. (2009). Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59.

Kakas, A. C. (2010). Abduction. In Sammut, C. and Webb, G. I., editors, *Encyclopedia of Machine Learning*, pages 3–9. Springer.

Kakas, A. C. and Flach, P. A. (2009). Abduction and Induction in Artificial Intelligence. *Journal of Applied Logic*, 7(3):251.

Katzouris, N., Artikis, A., and Paliouras, G. (2014). Incremental learning of event definitions with inductive logic programming. *CoRR*, abs/1402.5988.

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

Kersting, K. (2006). *An Inductive Logic Programming Approach to Statistical Relational Learning*. Dissertations in artificial intelligence. IOS Press.

Kersting, K. (2012). Lifted probabilistic inference. In de Raedt, L., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., and Lucas, P. J. F., 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.

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

Kersting, K., de Raedt, L., and Raiko, T. (2006). Logical Hidden Markov Models. *Journal of Artificial Intelligence Research (JAIR)*, 25(1):425–456.

Kimmig, A., Costa, V. S., Rocha, R., Demoen, B., and de Raedt, L. (2008). On the efficient execution of ProbLog programs. In *Logic Programming*, pages 175–189. Springer.

Kimmig, A., Demoen, B., de Raedt, L., Costa, V. S., and Rocha, R. (2011). On the implementation of the probabilistic logic programming language ProbLog. *Theory and Practice of Logic Programming*, 11:235–262.

Kok, S., Singla, P., Richardson, M., Domingos, P., Sumner, M., Poon, H., and Lowd, D. (2005). The Alchemy system for statistical relational AI. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://alchemy.cs.washington.edu.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Kosmopoulos, D., Antonakaki, P., Valasoulis, K., Kesidis, A., and Perantonis, S. (2008). Human behaviour classification using multiple views. In Darzentas, J., Vouros, G., Vosinakis, S., and Arnellos, A., editors, *Proceedings of Hellenic Conference on Artificial Intelligence*, volume 5138 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Kowalski, R. and Sadri, F. (1997). Reconciling the Event Calculus with the Situation Calculus. *The Journal of Logic Programming*, 31(1):39–58.

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

Kvarnström, J. (2005). *TALplanner and Other Extensions to Temporal Action Logic*. PhD thesis, Linköping.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). 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.

Lavee, G., Borzin, A., Rivlin, E., and Rudzsky, M. (2007). Building Petri Nets from video event ontologies. In *Advances in Visual Computing*, pages 442–451. Springer.

Lavee, G., Rudzsky, M., and Rivlin, E. (2013). Propagating Certainty in Petri Nets for Activity Recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(2):326–337.

Lévy, F. and Quantz, J. (1998). Representing Beliefs in a Situated Event Calculus. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI)*, pages 547–551.

Li, X. (2009). On the Use of Virtual Evidence in Conditional Random Fields. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1289–1297. ACL.

Liao, L., Fox, D., and Kautz, H. A. (2005). 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.

Lifschitz, V. (1994). Circumscription. In *Handbook of logic in Artificial Intelligence and Logic Programming*, volume 3, pages 297–352. Oxford University Press, Inc.

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

Lowd, D. and Domingos, P. (2007). 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.

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

Luckham, D. C. (2002). *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc.

Luckham, D. C. (2011). *Event Processing for Business: Organizing the Real-Time Enterprise*. Wiley.

Manfredotti, C. (2009). Modeling and Inference with Relational Dynamic Bayesian Networks. In Gao, Y. and Japkowicz, N., editors, *Advances in Artificial Intelligence*, volume 5549 of *Lecture Notes in Computer Science*, pages 287–290. Springer Berlin / Heidelberg.

Manfredotti, C., Hamilton, H., and Zilles, S. (2010). Learning RDBNs for Activity Recognition. In *NIPS Workshop on Learning and Planning from Batch Time Series Data*.

Mateus, P., Pacheco, A., Pinto, J., Sernadas, A., and Sernadas, C. (2001). Probabilistic Situation Calculus. *Annals of Mathematics and Artificial Intelligence*, 32(1):393–431.

McCarthy, J. (1980). Circumscription - A Form of Non-Monotonic Reasoning. *Artificial Intelligence*, 13(1-2):27–39.

McCarthy, J. (1983). Situations, Actions, and Causal Laws. Technical Report Memo 2, Stanford Artificial Intelligence Project, Stanford University.

McCarthy, J. (2002). Actions and other events in situation calculus. In Fensel, D., Giunchiglia, F., McGuinness, D. L., and Williams, M.-A., editors, *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR)*, pages 615–628. Morgan Kaufmann.

McCarthy, J. and Hayes, P. J. (1968). *Some philosophical problems from the standpoint of artificial intelligence*. Stanford University.

Miller, R. and Shanahan, M. (2002). 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.

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

Molinaro, C., Moscato, V., Picariello, A., Pugliese, A., Rullo, A., and Subrahmanian, V. S. (2014). PADUA: Parallel Architecture to Detect Unexplained Activities. *ACM Transactions on Internet Technology (TOIT)*, 14(1):3:1–3:28.

Morariu, V. I. and Davis, L. S. (2011). Multi-agent event recognition in structured scenarios. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3289–3296. IEEE Computer Society.

Morzenti, A., Mandrioli, D., and Ghezzi, C. (1992). A Model Parametric Real-time Logic. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 14(4):521–573.

Mueller, E. T. (2006). *Commonsense Reasoning*. Morgan Kaufmann.

Mueller, E. T. (2007). Discrete event calculus with branching time. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, pages 126–131.

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

Muggleton, S. (2000). Learning Stochastic Logic Programs. *Electronic Transactions on Artificial Intelligence*, 4(B):141–153.

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580.

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

Natarajan, P. and Nevatia, R. (2007). Hierarchical Multi-channel Hidden Semi Markov Models. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2562–2567.

Natarajan, S., Bui, H. H., Tadepalli, P., Kersting, K., and Wong, W.-K. (2008). Logical Hierarchical Hidden Markov Models for Modeling User Activities. In *Proceedings of the 18th International Conference Inductive Logic Programming (ILP)*, volume 5194 of *Lecture Notes in Computer Science*, pages 192–209. Springer.

Nath, A. and Domingos, P. (2010). Efficient Lifting for Online Probabilistic Inference. In Fox, M. and Poole, D., editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press.

Ng, R. T. and Subrahmanian, V. S. (1992). Probabilistic logic programming. *Information and Computation*, 101(2):150–201.

Ngo, L. and Haddawy, P. (1997). Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171(1-2):147 – 177.

Niepert, M., Noessner, J., and Stuckenschmidt, H. (2011). Log-Linear Description Logics. In Walsh, T., editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2153–2158.

Noessner, J., Niepert, M., and Stuckenschmidt, H. (2013). RockIt: Exploiting Parallelism and Symmetry for MAP Inference in Statistical Relational Models. In desJardins, M. and Littman, M. L., editors, *Proceedings of the 27th AAAI Conference on Artificial Intelligence, 2013, USA*. AAAI Press.

Nonnengart, A. and Weidenbach, C. (2001). Computing Small Clause Normal Forms. In Robinson, J. A. and Voronkov, A., editors, *Handbook of Automated Reasoning*, pages 335–367. Elsevier and MIT Press.

Okeyo, G., Chen, L., and Wang, H. (2014). Combining Ontological and Temporal Formalisms for Composite Activity Modelling and Recognition in Smart Homes. *Future Generation Computer Systems*, 39:29–43.

Okeyo, G., Chen, L., Wang, H., and Sterritt, R. (2012). A Hybrid Ontological and Temporal Approach for Composite Activity Modelling. In Min, G., Wu, Y., Liu, L. C., Jin, X., Jarvis, S. A., and Al-Dubai, A. Y., editors, *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1763–1770. IEEE Computer Society.

Papai, T., Kautz, H., and Stefankovic, D. (2012). Slice Normalized Dynamic Markov Logic Networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors,

Advances in Neural Information Processing Systems 25, pages 1907–1915. Curran Associates, Inc.

Papai, T., Singla, P., and Kautz, H. A. (2011). Constraint Propagation for Efficient Inference in Markov Logic. In Lee, J. H.-M., editor, *In Proceedings of the 17th International Conference in Principles and Practice of Constraint Programming (CP)*, Lecture Notes in Computer Science, pages 691–705. Springer.

Paschke, A. and Kozlenkov, A. (2009). Rule-Based Event Processing and Reaction Rules. In *Proceedings of the 3rd International Symposium on Rules (RuleML)*, volume 5858 of *Lecture Notes in Computer Science*, pages 53–66. Springer.

Patkos, T. and Plexousakis, D. (2008). A Theory of Action, Knowledge and Time in the Event Calculus. In *Artificial Intelligence: Theories, Models and Applications*, pages 226–238. Springer.

Petri, C. A. (1966). *Communication with automata*. PhD thesis, Universität Hamburg.

Pinto, J. and Reiter, R. (1993). Temporal Reasoning in Logic Programming: A Case for the Situation Calculus. In Warren, D. S., editor, *Proceedings of the 10th International Conference on Logic Programming (ICLP)*, pages 203–221. MIT Press.

Pinto, J. and Reiter, R. (1995). Reasoning about time in the Situation Calculus. *Annals of Mathematics and Artificial Intelligence*, 14(2):251–268.

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

Provetti, A. (1996). Hypothetical reasoning about actions: From Situation Calculus to Event Calculus. *Computational Intelligence*, 12(3):478–498.

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

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

Ray, O., Broda, K., and Russo, A. (2003). Hybrid Abductive Inductive Learning: A Generalisation of Progol. In Horváth, T., editor, *Inductive Logic Programming: 13th International Conference (ILP)*, volume 2835 of *Lecture Notes in Computer Science*, pages 311–328. Springer.

Reiter, R. (1991). Artificial intelligence and mathematical theory of computation. chapter The Frame Problem in Situation the Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression, pages 359–380. Academic Press Professional, Inc., San Diego, CA, USA.

Reiter, R. (1996). Natural actions, concurrency and continuous time in the situation calculus. In Aiello, L. C., Doyle, J., and Shapiro, S. C., editors, *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 2–13. Morgan Kaufmann.

Reiter, R. (1998). Sequential, Temporal GOLOG. In Cohn, A. G., Schubert, L. K., and Shapiro, S. C., editors, *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 547–556. Morgan Kaufmann.

Reiter, R. (2001). *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Renkens, J., Van den Broeck, G., and Nijssen, S. (2012). k-Optimal: a novel approximate inference algorithm for ProbLog. *Machine learning*, 89(3):215–231.

Riboni, D., Pareschi, L., Radaelli, L., and Bettini, C. (2011). Is Ontology-based Activity Recognition Really Effective? In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 427–431.

Riedel, S. (2008). 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.

Romdhane, R., Crispim, C., Bremond, F., and Thonnat, M. (2013). Activity Recognition and Uncertain Knowledge in Video Scenes. In *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*, pages 377–382.

Rota, N. and Thonnat, M. (2000). Activity recognition from video sequences using declarative models. In Horn, W., editor, *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI)*, pages 673–680. IOS Press.

Ryoo, M. S. and Aggarwal, J. K. (2006). Recognition of Composite Human Activities through Context-Free Grammar Based Representation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1709–1718. IEEE Computer Society.

Ryoo, M. S. and Aggarwal, J. K. (2009). Semantic Representation and Recognition of Continued and Recursive Human Activities. *International Journal of Computer Vision*, 82(1):1–24.

Sadilek, A. and Kautz, H. A. (2012). Location-Based Reasoning about Complex Multi-Agent Behavior. *Journal of Artificial Intelligence Research (JAIR)*, 43:87–133.

Saguna, Zaslavsky, A., and Chakraborty, D. (2011). Recognizing Concurrent and Interleaved Activities in Social Interactions. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, pages 230–237.

Sandewall, E. (1994). *Features and Fluents: A Systematic Approach to the Representation of Knowledge About Dynamical Systems*. Oxford University Press, Oxford.

Sarkhel, S. and Gogate, V. (2013). Lifting WALKSAT-Based Local Search Algorithms for MAP Inference. In *Statistical Relational Artificial Intelligence, Papers from the 2013 AAAI Workshop*, volume WS-13-16 of *AAAI Workshops*. AAAI.

Sarkhel, S., Venugopal, D., Singla, P., and Gogate, V. (2014). Lifted MAP Inference for Markov Logic Networks. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 33 of *JMLR Proceedings*, pages 859–867. JMLR.org.

Sato, T. (1995). A Statistical Learning Method for Logic Programs with Distribution Semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP)*, pages 715–729. MIT Press.

Sato, T. and Kameya, Y. (2001). Parameter Learning of Logic Programs for Symbolic-statistical Modeling. *Journal of Artificial Intelligence Research*, 15:391–454.

Schiffel, S. and Thielscher, M. (2006). Reconciling Situation Calculus and Fluent Calculus. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 287. AAAI Press.

Selman, J., Amer, M. R., Fern, A., and Todorovic, S. (2011). PEL-CNF: Probabilistic event logic conjunctive normal form for video interpretation. In *Proceedings of the International Conference on Computer Vision Workshops (ICCVW)*, pages 680–687. IEEE Computer Society.

Shanahan, M. (1997). *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. MIT Press.

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

Shet, V. D., Harwood, D., and Davis, L. S. (2005). VidMAP: Video Monitoring of Activity with Prolog. In *Proceedings of the International Conference on Video and Signal Based Surveillance (AVSS)*, pages 224–229. IEEE Computer Society.

Shet, V. D., Neumann, J., Ramesh, V., and Davis, L. S. (2007). 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.

Shet, V. D., Singh, M., Bahlmann, C., Ramesh, V., Neumann, J., and Davis, L. S. (2011). Predicate Logic Based Image Grammars for Complex Pattern Recognition. *International Journal of Computer Vision*, 93(2):141–161.

Shi, Y., Bobick, A. F., and Essa, I. A. (2006). Learning Temporal Sequence Model from Partially Labeled Data. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1631–1638. IEEE Computer Society.

Shterionov, D., Kimmig, A., Mantadelis, T., and Janssens, G. (2010). DNF sampling for ProbLog inference. In *Proceedings International Colloquium on Implementation of Constraint and LOgic Programming Systems (CICLOPS)*, page 15.

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

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

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

Singla, P. and Mooney, R. J. (2011). Abductive Markov Logic for Plan Recognition. In *In Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 1069–1075. AAAI Press.

Siskind, J. M. (2001). Grounding the Lexical Semantics of Verbs in Visual Perception using Force Dynamics and Event Logic. *Journal of Artificial Intelligence Research (JAIR)*, 15:31–90.

Song, Y. C., Kautz, H., Allen, J., Swift, M., Li, Y., Luo, J., and Zhang, C. (2013a). A Markov Logic Framework for Recognizing Complex Events from Multimodal Data. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, ICMI '13, pages 141–148, New York, NY, USA. ACM.

Song, Y. C., Kautz, H. A., Li, Y., and Luo, J. (2013b). A General Framework for Recognizing Complex Events in Markov Logic. In *AAAI Workshop: Statistical Relational Artificial Intelligence*. AAAI.

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

Thielscher, M. (1999). From Situation Calculus to Fluent Calculus: State update axioms as a solution to the inferential frame problem. *Artificial intelligence*, 111(1):277–299.

Thielscher, M. (2001). The qualification problem: A solution to the problem of anomalous models. *Artificial Intelligence*, 131(1):1–37.

Tran, S. D. and Davis, L. S. (2008). Event Modeling and Recognition Using Markov Logic Networks. In *Proceedings of the 10th European Conference on Computer Vision (ECCV)*, volume 5303 of *Lecture Notes in Computer Science*, pages 610–623. Springer.

Turaga, P., Chellappa, R., Subrahmanian, V. S., and Udrea, O. (2008). Machine Recognition of Human Activities: A Survey. *IEEE Transactions on Circuits and Systems for Video Technology.*, 18(11):1473–1488.

Vail, D. L., Veloso, M. M., and Lafferty, J. D. (2007). 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.

Valiant, L. G. (1979). The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421.

Van Belleghem, K., Denecker, M., and De Schreye, D. (1997). On the relation between Situation Calculus and Event Calculus. *The Journal of Logic Programming*, 31(1):3–37.

Vu, V.-T., Brémond, F., and Thonnat, M. (2003). Automatic Video Interpretation: A Novel Algorithm for Temporal Scenario Recognition. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1295–1302.

Wang, J. and Domingos, P. (2008). Hybrid Markov Logic Networks. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1106–1111. AAAI Press.

Wasserkrug, S., Gal, A., and Etzion, O. (2005). A Model for Reasoning with Uncertain Rules in Event Composition Systems. In *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 599–608. AUAI Press.

Wasserkrug, S., Gal, A., Etzion, O., and Turchin, Y. (2008). Complex Event Processing Over Uncertain Data. In Baldoni, R., editor, *Proceedings of the Second International Conference on Distributed Event-Based Systems (DEBS)*, volume 332 of *ACM International Conference Proceeding Series*, pages 253–264. ACM.

Wasserkrug, S., Gal, A., Etzion, O., and Turchin, Y. (2012). Efficient Processing of Uncertain Events in Rule-Based Systems. *IEEE Transactions on Knowledge and Data Engineering*, 24(1):45–58.

Welty, C. A. and Fikes, R. (2006). A Reusable Ontology for Fluents in OWL. In Bennett, B. and Fellbaum, C., editors, *Formal Ontology in Information Systems, Proceedings of the Fourth International Conference (FOIS)*, volume 150 of *Frontiers in Artificial Intelligence and Applications*, pages 226–236. IOS Press.

Wu, T.-y., Lian, C.-c., and Hsu, J. Y.-j. (2007). 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.

Index

Glossary

ALP	**A**bductive **L**ogic **P**rogramming
ASP	**A**nswer **S**et **P**rogramming
AUPRC	**A**rea **U**nder **P**recision **R**ecall **C**urve
BDD	**B**inary **D**ecision **D**iagram
BN	**B**ayesian **N**etwork
CAVIAR	**C**ontext **A**ware **V**ision using **I**mage-based **A**ctivity **R**ecognition
CE	**C**omposite **E**vent
CEP	**C**omplex **E**vent **P**rocessing
CLL	**C**onditional **L**og-**L**ikelihood
CNF	**C**onjunctive **N**ormal **F**orm
CRF	**C**onditional **R**andom **F**ield
CRS	**C**hronicle **R**ecognition **S**ystem
DEC	**D**iscrete **E**vent **C**alculus
DNF	**D**isjunctive **N**ormal **F**orm
$\mathtt{EC}_{\mathtt{crisp}}$	**Crisp** **E**vent **C**alculus
HMM	**H**idden **M**arkov **M**odel
ILP	**I**nductive **L**ogic **P**rogramming
l–CRF	**l**inear-chain **C**onditional **R**andom **F**ield
LLDL	**L**og **L**inear **D**escription **L**ogic
LP	**L**inear **P**rogramming
MAP	**M**aximum **a**-**p**osteriori
MAX-SAT	**Max**imum **Sat**isfiability Problem

MCMC	**M**arkov **C**hain **M**onte **C**arlo
MLN	**M**arkov **L**ogic **N**etwork
`MLN–EC`	**M**arkov **L**ogic **N**etwork based **E**vent **C**alculus
MN	**M**arkov **N**etwork
MRF	**M**arkov **R**andom **F**ield
OWL	Web Ontology Language
PEL	**P**robabilistic **E**vent **L**ogic
PPG	**P**robabilistic **P**enalty **G**raph
ProbLog	**Prob**abilistic **Log**ic Programming
`ProbLog–EC`	**ProbLog** based **E**vent **C**alculus
SA	**S**tochastic **A**utomaton
SAT	Boolean **Sat**isfiability Problem
SDE	**S**imple, **D**erived **E**vents
SLD	**S**elective **L**inear **D**efinite

Symbols

Σ	The set of CE definitions
Σ'	The set of complimentary CE definitions
$\mathcal{F}$	The finite domain of fluents
$\mathcal{E}$	The finite domain of events
$\mathcal{T}$	The finite domain of time-points
$\mathcal{O}$	The finite domain of individual entities, e.g. persons, objects, etc.
$\mathcal{C}$	The finite domain of all constants, i.e. $\mathcal{C}=\mathcal{T} \cup \mathcal{O} \cup \mathcal{E} \cup \mathcal{F}$
$\mathbb{R}$	The set of real numbers
w_i	The weight value of the i-th clause/formula, where $w_i \in \mathbb{R}$
F_i	The i-th formula
L	A knowledge base of weighted formulas
F_c	The set of clauses produced from L and constants $\mathcal{C}$
$M_{L,\mathcal{C}}$	The ground Markov network that is created from weighted formulas of L and constants $\mathcal{C}$
Z	Partition function
$\|A\|$	Cardinality of set A
$\|A\|\times\|B\|$	The Cartesian product of cardinalities $\|A\|$ and $\|B\|$
$\wedge$	Conjunction
$\vee$	Disjunction
$\Leftrightarrow$	Equivalence
$\Leftarrow$ or $\Rightarrow$	Implication
$\exists$	Existential quantification
$\forall$	Universal quantification
$P(Y \mid X)$	Conditional distribution of random variables Y, given X

Γ	ProbLog program
BK	Background knowledge
F	Set of probabilistic facts in a ProbLog program
$p_i :: f_i$	The i-th probabilistic fact, where p_i indicates the probability of fact f_i
L_Γ	Maximal set of ground logical facts
L_g	Ground logic program or sub-program
$P_s(q\|\Gamma)$	Success probability of query q in a ProbLog program Γ

“Everything not saved will be lost”
— Nintendo “Quit Screen” message