Scalable Data Analytics,
Scalable Algorithms, Software Frameworks
and Visualization ICT-2013 4.2.a

Project **FP7-619435/SPEEDD**
Deliverable **D3.3**
Distribution **Public**

http://speedd-project.eu

Final version of event recognition and forecasting technology – Part I

Fabiana Fournier (IBM) and Inna Skarbovsky (IBM)

Status: Submitted (Version 1.0)

September 2016

Project

Project Ref. no	FP7-619435
Project acronym	SPEEDD
Project full title	Scalable ProactivE Event-Driven Decision Making
Project site	http://speedd-project.eu/
Project start	February 2014
Project duration	3 years
EC Project Officer	Stefano Bertolo

Deliverable

Deliverable type	Prototype
Distribution level	Restricted
Deliverable Number	D3.3
Deliverable Title	Final version of event recognition and forecasting technology
Contractual date of delivery	M32 (September 2016)
Actual date of delivery	September 2016
Relevant Task(s)	WP3/Tasks 3.1-3.3
Partner Responsible	IBM
Other contributors	NCSR
Number of pages	16
Author(s)	Fabiana Fournier (IBM) and Inna Skarbovsky (IBM)
Internal Reviewers	Ivo Correia (Feedzai) and Elias Alevizos (NCSR)
Status & version	Final
Keywords	Complex event processing, forecasted/predicted event, event recognition, uncertain event

Executive summary

This document is the first part of Deliverable 3.3 “Final version of event recognition and forecasting technology” and its purpose is to present the latest results of T3.2 (event recognition under uncertainty) and T3.3 (event forecasting under uncertainty), and the third version of the event recognition and forecasting component (software).

This report summarizes our work during months M30-M32, specifically we describe the performance analysis of recall and precision for the credit card fraud detection use case; and enhancements made to the event processing network for the traffic management use case. Our findings are consistent in all our tests in the two use cases. We can achieve a very high level of precision and recall when comparing PROTON’s performance to the annotated data provided to us.

More importantly, is the comparisons we made between our “certain” and “uncertain” implementations. In both use cases, the forecasted events are emitted in an average 3 minutes before their counterparts in the certain case while maintaining a high level of accuracy. This enables operators in both use cases, to make proactive actions to alleviate the undesired situation, whether it is a congestion or a fraud.

Document history

Version	Date	Author	Change Description
0.1	31/08/2016	Fabiana Fournier (IBM)	First draft
0.2	15/09/2016	Fabiana Fournier (IBM)	Second draft for internal review
0.3	25/09/2016	Fabiana Fournier (IBM)	Updates per internal review
1.0	30/09/2016	Fabiana Fournier (IBM)	Final fixes and cleanup

Table of contents

List of tables

List of figures

Acronyms

CEP	Complex Event Processing
DM	Decision Making
EEP	Extendable Expression Parser
EPA	Event Processing Agent
EPN	Event Processing Network
PETITE	Proton EvenT Injection & Time comprEssion
PROTON	PROactive Technology ONline
SPEEDD	Scalable ProactivE Event-Driven Decision making
WP	Work Package

1 Introduction

1.1 Purpose and scope of the document

Work Package 3 (WP3) "Real-Time Event Recognition and Forecasting under Uncertainty" deals with all the developments around event processing technologies under uncertainty. This report is the final version of SPEEDD (Scalable ProactivE Event-Driven Decision) event recognition and forecasting technology and it includes final results for T3.2 (event recognition under uncertainty) and T3.3 (event forecasting under uncertainty), and the final version of the event recognition and forecasting component (software). This report covers the improvements made to both the use case implementations and to the CEP tool engine IBM PROactive Technology ONline (PROTON) since the last report in August 2016 (M30). Specifically, we describe the recall and precision for the credit card fraud detection use case and an extended event-driven implementation for the traffic management use case. In addition, we describe the extensions made to PROTON to support this new implementation.

This report is structured as follows: Section 12 summarizes the extensions made to PROTON in these last couple of months. Section 3 presents our results of our evaluation of the event processing application for the fraud use case in terms of recall and precision. Sections 34 presents enhancements made to the traffic management use case implementation. We summarize our report in Section 45.

1.2 Relationship with other documents

At the heart of the SPEEDD prototype resides the complex event processing component, therefore, this report is strongly related to D6.7 "Final Integrated Prototype" (under preparation to be submitted one month later than this report, i.e.; M33). The requirements for the CEP engine are dictated from the use cases in the project, thus, this report is also strongly related to system requirements and evaluation for the Proactive Traffic Management use case described in D8.3 and for the Proactive Credit Card Fraud Management described in D7.3. With relation to the traffic management use case, this report is also related to the developments in the micro simulator, and therefore related to D8.4 "Final Version of Micro-Simulator", submitted at month 24.

The main goal of the CEP component is to derive forecasted events that feed the decision making (DM) component so actions can be taken before an undesired event (such as a congestion situation in the high way) takes place. Therefore our work is also related to D4.3 "Final version of real-time decision-making technology".

1.3 Updates since the D3.2 re-submission

As aforementioned, we describe in this report the additions made during months M30-M32, specifically we refer to:

Performance analysis of recall and precision for the credit card fraud detection use case; similarly to the analysis made to the traffic management use case and reported in D3.2 revised version[1] (see Section 5.2.2.3). We also recall that the overall performance of the system has been detailed in D3.2 revised (refer to Section 6) submitted at M32.

An improved event-driven application for the traffic use case that relates to the detection of congestions due to incidents in the road (Section 4). This includes analysis of recall and precision for this case, relation to noise in the input events, and the inclusion of new Event processing Agents (EPAs) into our Event Processing network (EPN).

We note that we follow the semantics and language from [1].

2 Extensions made to PROTON

In order to support the decision making component in the SPEEDD prototype, a new built-in function for standard deviation has been added to the **PROTON EEP** (Expandable Expression Parser). The expressions and functions are tested at build-time and evaluated at runtime by EEP. For a description of EEP please refer to Section 2.2.5 in D3.1 "First version of event recognition and forecasting technology V1".

Taking into account standard deviations help us dealing with potential noise inherent in the input events that result from the inaccuracy of the sensors themselves.

The aim of standard deviation with regards to the traffic use case is to provide the DM module a way to deal with uncertainty inherent in the input events (for details refer to D4.3 "Final version of real-time decision-making technology". The standard deviation is applied in the *flow* and *density* attributes in the corresponding derived events from all the EPAs that compute averages, that is, the *AverageOffRampValuesOverInterval*, *AverageOnRampValuesOverInterval,* and *AverageDensityAndSpeedPerLocationOverInterval* (For a complete description of these EPAs refer to[1] section 5.1).

[1] http://speedd-project.eu/sites/default/files/D3.2_-_Second_version_of_event_recognition_and_forecasting_technology-revised.pdf

3 Evaluation of the event processing application for the credit card fraud detection use case

In this section we describe the evaluation results of the event processing application for the credit card fraud detection use case in terms of recall and precision. We recall that the overall performance of the system has been detailed in D3.2 (revised)[1] submitted at M32.

For a full description of the EPN for the credit card use case refer to Section 5 in[1]. Henceforth, we describe the tests performed in order to validate the application with regards to recall and precision of the event rules.

3.1 Implementation and evaluation for the fraud detection use case

The main goal of the CEP component with regards to the credit card fraud detection use case is to assess whether the event patterns can indeed detect fraudulent situations, and whether inclusion of uncertainty can help in the decision making process. That is, we are looking to answer two questions:

- Can we detect fraudulent situations based on the implemented event patterns?
- Moreover, can we any benefits when including uncertainty aspects?

To address the first question, we had to overcome the main limitation of testing on real data due to privacy issues. As it has been cleared to us, the thresholds for the patterns and windows given to us were not the ones used by Feedzai's system. Furthermore, in order to be able to compare our results, we needed a data set that can be run using Feedzai's system and can be annotated by the system according to specific patterns. Moreover, we should recall that fraud patterns are quite rare to happen. To deal with all these challenges and being able to compare the outputs for the two systems, Feedzai has generated a sufficient number of synthetic transactions that derived the following patterns (for a full description of the patterns refer to[1]):

- Consecutive withdrawals of increasing or decreasing amounts for a single card (*IncreasingAmounts* and *DecreasingAmounts,* respectively).
- A high number of transactions in a short time window for a single card (*FlashAttack*).
- Consecutive attempts to use the same card in different physical locations (*TransactionsInFarAwayPlaces*).
- Sudden card use near the expiration date (*SuddenCardUseNearTheExpirationDate*).

The way to address the second question is to have two applications or EPNs, once including uncertainty aspects and the other one without uncertainty, i.e. deterministic, and compare between the results of these two EPNs. This is a common approach in CEP engines dealing with uncertainty (see for example in [2]).

3.1.1 Dataset

The dataset comprised 30,000 transactions over a time span of 16 days. We applied our utility PETITE (Proton EvenT Injection & Time compression) developed in SPEEDD in order to shrink the data into a one hour run (for a description of PETITE refer to[1]).

As aforementioned we tested for the detection of five event patterns: *IncreasingAmounts, DecreasingAmounts, FlashAttack, TransactionsInFarAwayPlaces, SuddenCardUseNearTheExpirationDate.*

3.1.2 Evaluation results

As explained above, we performed two types of tests, one to address the accuracy of our patterns, and the other to address the potential benefit of applying uncertainty aspects in the patterns.

For the first case, we run our application without applying uncertainty, and compared to the annotations received from Feedzai's system. Feedzai's system flagged 9038 transactions as fraud, while PROTON flagged 8908. Our run yielded a precision of 1 and recall of 0.985. That is all PROTON's patterns derived corrected output (fraudulent transactions) yielding a precision value of 1; while we haven't "caught" 130 fraudulent transactions marked by Feedzai but not by PROTON . Further investigation of the results, revealed the main reason for the discrepancies. The most frequent pattern is the *FlashAttack*. In this particular pattern, Feedzai's system used a sliding overlapping window of seven minutes, while we used a non-overlapping window. Thus, we "missed" borderline fraudulent transactions that happened at the end of the windows. Evidently, changing the non-overlapping windows policy to the sliding one will improve the results from the recall point of view, but in turn; it will result in multiple reports of the same card (and transactions) possibly annoying the operators. Therefore, we reserve this change for the future if indeed required by the domain experts despite the potential multiple reports of the same card.

For the second case, we run the same dataset but using the uncertain version of PROTON and compared the results with our previous ones (with no use of uncertainty, i.e., certain patterns). In this case, we derive a suspicious fraud alert whenever the certainty attribute in the derived event is bigger than 0.6. In other words, we start alerting using a threshold of 0.6. In this case, our results shown that while we were able to detect all the "certain" derived events (recall = 1), we sometimes had false positives giving a precision of 0.97. In other words, we had 250 transactions that we mistakenly flagged as suspicious fraud. The explanation is straightforward. In the uncertain case we start reporting with fewer transactions in the matching set. For example, in the *IncreasingAmounts* we start alerting after 6 events in the matching set (the threshold for our *Sigmoid* function and certainty > 0.6) while in the certain case, we alert only after 7 events are encountered in the matching set). Regarding recall, we are able to report correctly all found fraudulent transactions, giving no false negatives.

However, more importantly, is the fact that in the uncertain case we were able to alert in an average 3 minutes before its counterpart in the certain case, thus giving enough time to an operator to block a credit card earlier than they can today.

3.1.2.1 Summary of results

Our results are twofold: first we were able to validate our implementation by comparing our results to Feedzai's system, getting a very high level of accuracy. Second, our results show the benefit of including uncertainty aspects in an event driven application in the domain of credit card fraud. As for the first case, also when comparing certainty and uncertainty, our results were of a high level of accuracy. Furthermore, the fact that we can alert ahead of time can have a significant financial impact as we can avoid allowing "future fraud transactions". The "extra" time saved, enables an automated system to block a transaction in an average 3 minutes earlier.

4 Event processing application for the traffic management use case

In this section we describe the work carried out in the traffic management use case for the last two months, including the addition of a new EPA for forecasting a rapid congestion as a result of an incident in the road, and inclusion of noise in the input data. For the full implemented EPN and analysis of recall and precision refer to [1].

4.1 Addition of a new EPA for forecasting a congestion due to a possible incident on the road

The motivation for the addition of this new EPA was to forecast a congestion that is caused by a "sudden" build-up on the road, as opposed to a forecasted congestion which is caused by a "moderate" build-up. Build-up is defined by a combination of a number of parameters describing the traffic flow, such as decrease in speed, or increase in occupancy and density. Therefore any of these can be indicators of a possible congestion.

Therefore, while in the *PredictedTrend* EPA (refer to[1]) we were looking for a an increase in the density values of at least 5 consecutive input events until either we detect a *Congestion* or a *ClearCongestion* event, in this case, we derive a *PossibleIncident* event whenever we detect an increase in the occupancy parameter of at least three events over a (short) temporal window of two minutes. Henceforth, we describe in detail this new EPA.

Motivation: The CEP run-time engine will derive a *PossibleIncident* event whenever it detects a rapid increase in the average occupancy values of at least 3 consecutive input events. In other words, we perceive that a rapid build-up is taking place possibly indicating an incident.

As was mentioned before, the indication for a buildup is dependent on values of a number of parameters of the flow, and we would like to take these values into account when calculating the certainty of the *PossibleIncident* event. As in the past, we use the *Sigmoid* function for the probability of the derived event. Therefore the Sigmoid function receives three parameters and four coefficients and returns 1/ (1 + e^-(a + b1x1+b2x2+b3x3)), where:

- X1 – average occupancy
- X2 – denotes the density
- X3 – *trend.count* (number of events in the matching set)

We derive an output event (*PossibleIncident*) only if the ratio of increase in *average_occupancy* is bigger than 1.5 (drastic changes in the occupancy values indicating fast build-up due to incident as opposed to "regular" congestion) and if the certainty of the derived event is above 0.75 (see Figure 1). We take the speed into account in the initiation of the temporal window for incident detection – the rule is enabled only if we see a drop in speed below a certain threshold indicating something is "going on" in the section of the road.

Event recognition process:

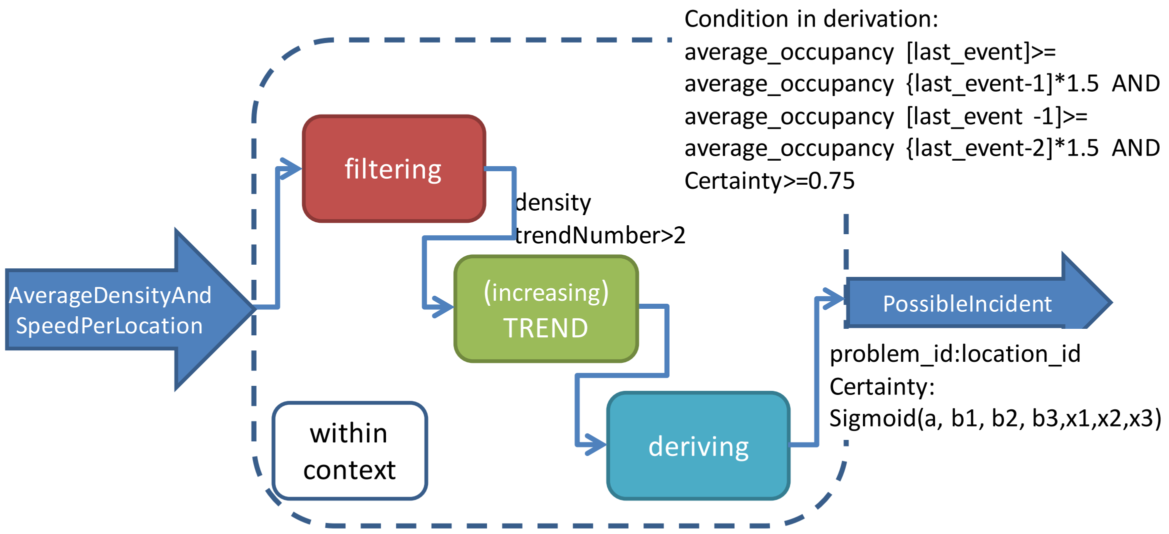
Condition in derivation:
average_occupancy [last_event]>=
average_occupancy {last_event-1]*1.5 AND
average_occupancy [last_event -1]>=
average_occupancy {last_event-2]*1.5 AND
Certainty>=0.75
filtering
density
trendNumber>2
AverageDensityAnd SpeedPerLocation
(increasing) TREND
PossibleIncident
problem_id:location_id
Certainty:
Sigmoid(a, b1, b2, b3,x1,x2,x3)
within context
deriving

Figure 1: Event recognition process for PossibleIncident EPA

Pattern policies:

Evaluation	Cardinality	Repeated	Consumption
DEFERRED	UNRESTRICTED	FIRST	REUSE

Context:

Segmentation: by location_id

Initiator policy: IGNORE

Initiator: AverageDensityAndSpeedPerLocation.average_speed ≤ speed_threshold3

Meaning: We open a single context for a location in order to detect an increasing TREND pattern and close it after 2 minutes. To this end, the context is opened with the first input event that comes with an average_speed value smaller than threshold3 (we used the value of 40km/hour that was selected in all our tests up to now to denote congestion). As we use the DEFERRED policy, we only derive a *PredictedIncident* event at the end of the temporal window (2 min) if the pattern and the conditions on the derivation are satisfied.

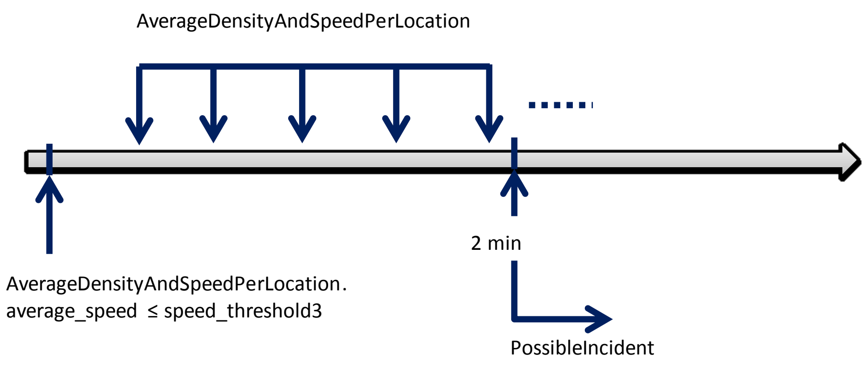
AverageDensityAndSpeedPerLocation
AverageDensityAndSpeedPerLocation.
average_speed ≤ speed_threshold3
2 min
PossibleIncident

Figure 2: Context for PossibleIncident EPA

4.1.1 Values of operator operands and coefficients of the Sigmoid function

The following values for the *Sigmoid* function of the *PredictedIncident* EPA have been chosen (Table 1). The values have been selected by assigning probabilities for the two extreme conditions of close to 1 and close to 0, and looking at the simulated data.

Table 1: Variable values calculation for Sigmoid function for the PossibleIncident EPA

x1	X2	X3	probability
43.01	111.51	3	0.269
60	308.419	3	0.5
66.66	218.83	4	0.731
91.44	583.707	6	0.99

The coefficients that match are:

a: -5.089; b1: 0.042; b2: 0.004; and b3: 0.635

4.1.2 Results

In order to test the addition of the new EPA, 10 new simulations with one incident annotation by the Aimsun simulator have been generated. Unfortunately, only in 2 out of the 10 incident simulations, happen in the main road. These have been detected by the pattern. However, as this number is very low we can generalize on our findings.

4.2 Inclusion of noise in the input event streams

Generally speaking, there are two types of noise in the sensor input events. The first one is “missing data” represented by “-1” that appear when there was not an actual value in an attribute (as a result of no communication, for example). These values can be easily filtered out during the first step in each EPA in the application. The second type of noise is inherent in the values received due to inaccuracy in the measurements. In order to cope with this second type of noise, we take the following steps:

1. We calculate the standard deviation over the values of flow and density of the events in the matching sets of three EPAs: *AverageOffRampValuesOverInterval*, *AverageOnRampValuesOverInterval,* and *AverageDensityAndSpeedPerLocationOverInterval*. The corresponding derived events of these EPAs are then consumed by the DM, which in turn uses these standard deviations values.
2. Some of our operators in the EPN, filter events by density and speed, that is, we only take into consideration for the pattern matching step, events in between “normal ranges”. This enables filtering out potential “outliers” events that can result from erroneous measurements in these attributes and filtering in events in “normal” ranges.
3. In all our patterns, we base our derived events on matching sets with cardinality > 1, thus errors in measurements in single events are averaged out.
4. Handling “empty” values as “null” and explicitly relate to them at the application level (e.g., checking null values and replacing them with some average value).

5 Summary

This report summarizes our work during months M30-M32, specifically we describe the performance analysis of recall and precision for the credit card fraud detection use case; and enhancements made to the event processing network for the traffic management use case.

Our findings are consistent in all our tests in the two use cases. We can achieve a very high level of precision and recall when comparing PROTON’s performance to the annotated data provided to us.

More importantly, is the comparisons we made between our “certain” and “uncertain” implementations. In both use cases, the forecasted events are emitted in an average 3 minutes before their counterparts in the certain case while maintaining a high level of accuracy. This enables operators in both use cases, to make proactive actions to alleviate the undesired situation, whether it is a congestion or a fraud.

6 References

[1]. O. Etzion and P. Niblet. *Event processing in action*. Manning, 2010

[2]. G. Cugola, A. Margara, M. Matteucci, and G. Tamburrelli. Introducing uncertainty in complex event processing: model, implementation, and validation. Computing, pages 1–42, 2014. ISSN 0010-485X. doi: 10.1007/s00607-014-0404-y.

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

Project	**FP7-619435 / SPEEDD**
Deliverable	**D3.3**
Distribution	**Public**

http://speedd-project.eu/

Event Recognition and Forecasting Technology – Part II

Evangelos Michelioudakis, Nikos Katzouris, Alexander Artikis, Georgios Paliouras

Status: Final (Version 1.0)

February 2017

Project

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

Deliverable

Deliverabe type	report
Distribution level	Public
Deliverable Number	D3.3
Deliverable title	Event Recognition and Forecasting Technology – Part II
Contractual date of delivery	M33 (October 2016)
Actual date of delivery	February 2017
Relevant Task(s)	WP3/Task 3.1
Partner Responsible	NCSR "D"
Other contributors	
Number of pages	18
Author(s)	Evangelos Michelioudakis, Nikos Katzouris, Alexander Artikis, Georgios Paliouras
Internal Reviewers	
Status & version	Final
Keywords	Uncertainty, event calculus, statistical relational learning

Contents

List of Tables

Executive Summary

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

SPEEDD implements event recognition methods (also known as event pattern matching or event pattern detection systems), in order to extract useful information, in the form of events, by processing time-evolving data that comes from various sources (e.g., various types of sensor, network activity logs, ATMs, transactions, etc.). The extracted information — recognized and/or forecasted events — can be exploited by other systems or human experts, in order to monitor an environment and respond to the occurrence of significant events. Event recognition methods employ rich representation that can naturally and compactly represent events with complex relational structure, e.g., events that are related to other input/derived events with spatio-temporal constraints. Unfortunately, uncertainty is an unavoidable aspect of real-world event recognition applications and it appears to be a consequence of several factors. Consider for example, noisy or incomplete observations from road sensors, as well as imperfect definitions fraudulent activity. Under situations of uncertainty, the performance of an event recognition system may be seriously compromised. Another important characteristic of the SPEEDD project, is that machine learning algorithms must deal with large amounts of data that continuously evolves. As a result, the current base of event definitions may need to be refined or new events may appear. Therefore, the traditional approach of non-incremental batch machine learning algorithms cannot be applied in SPEEDD.

In the previous deliverable D3.2 we presented our recently developed algorithm for scalable probabilistic incremental learning of event definitions ($\texttt{OSL}\alpha$) that addresses the requirements imposed by the presence of uncertainty and large training sets. $\texttt{OSL}\alpha$ takes advantage of the succinct, structured and declarative representation of the Event Calculus formalism, in order to formally express events and their effects. To handle the uncertainty, it employs the state-of-the-art probabilistic and relational framework of Markov Logic Networks. The combination of probabilistic and logical modeling gives $\texttt{OSL}\alpha$ the advantage of learning event definitions with well defined probabilistic and logical schematics. In this document, we begin by briefly presenting our developed probabilistic structure learning method that exploits the background knowledge axiomatization to effectively constrain the search space of possible structures. Then we present experimental results on fraud management using a synthetic dataset that was provided by Feedzai.

Effective as it may be, $\texttt{OSL}\alpha$ has some limitations that restrict its applicability in the fraud domain. $\texttt{OSL}\alpha$ cannot effectively learn "long" patterns over large constant domains, such as "time" and thus

restricts its learning capabilities in learning patterns over the same timepoint due to high computational complexity. However in the fraud domain there is a natural requirement for learning sequences of events for card transactions over time in order for the learner to be able to discriminate between the positive and negative examples. This requirement stems OSLα inadequate for the learning task. To address this issue, we also describe in this document OLED, an alternative relational learning system that is able to overcome some of these limitations. We present experimental results with OLED that demonstrate that it is capable of learning efficiently relational patterns of fraudulent behavior. OLED (Online Learning of Event Definitions) is an Inductive Logic Programming system that, like OSLα learns in online fashion, i.e. in a single-pass over a stream of training data. It can also handle uncertainty, though less robustly than OSLα since it uses a less formal uncertainty handling mechanism than does not involve a probabilistic semantics. However, this makes OLED's functionality depend on less expensive operations, allowing it to overcome some of OSLα's limitations in the particular domain.

1 Introduction

1.1 History of the Document

Version	Date	Author	Change Description
0.1	3/10/2016	Evangelos Michelioudakis	Set up of the document
0.2	11/10/2016	All Deliverable Authors	First version
0.3	12/10/2016	All Deliverable Authors	Content adjusted: Fraud Experiments
0.4	1/2/2017	Nikos Katzouris	Content adjusted: OLED
0.5	8/2/2017	Evangelos Michelioudakis	Content adjusted: OSLα

1.2 Purpose and Scope of the Document

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

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

- SPEEDD researchers
- SPEEDD audit

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

1.3 Relationship with Other Documents

This document is related to project deliverable D3.1 and D3.2 “Event Recognition and Forecasting Technology” that present previous work on event recognition and machine learning for the SPEEDD prototype. Moreover, deliverables D6.1 and D6.5 “The Architecture Design of the SPEEDD Prototype and Second Integrated Prototype” which present the SPEEDD prototype architecture as well as the updated version of deliverable D7.1 which outlines the requirements and characteristics of the “Proactive Credit Card Fraud Management” project use case.

2

OSLα: Online Structure Learner

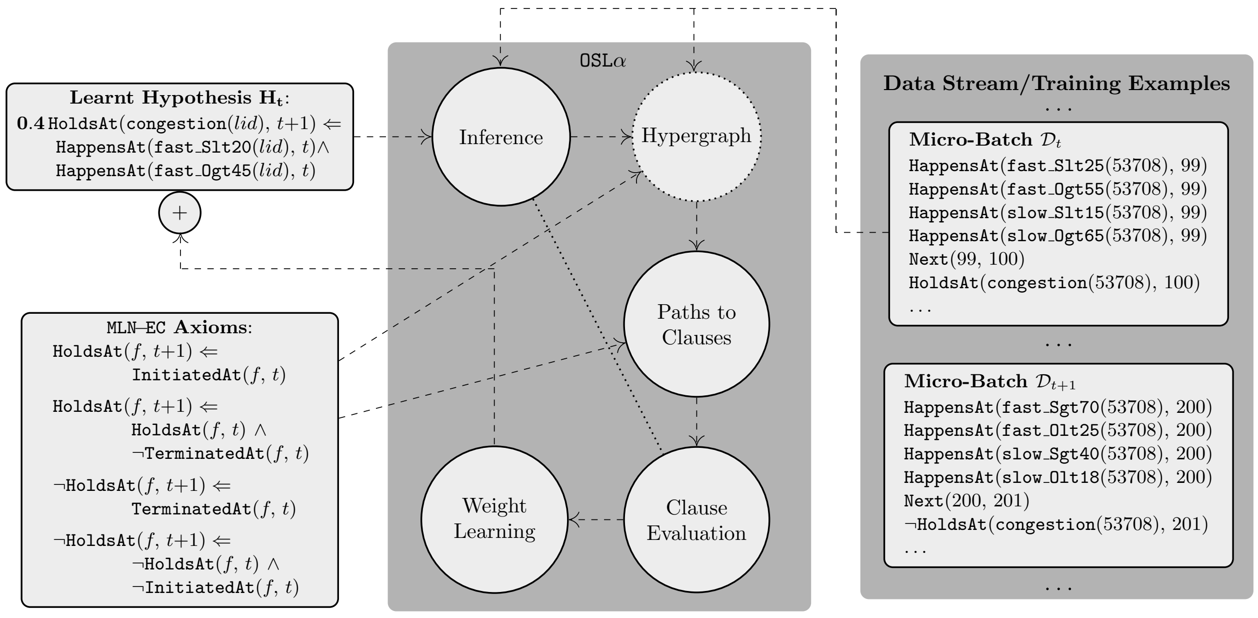
OSLα
Learnt Hypothesis $\mathbf{H_t}$:
0.4 HoldsAt(congestion(lid), t+1) ⇐
HappensAt(fast_Slt20(lid), t)∧
HappensAt(fast_Ogt45(lid), t)
+
MLN–EC Axioms:
HoldsAt(f, t+1) ⇐
InitiatedAt(f, t)
HoldsAt(f, t+1) ⇐
HoldsAt(f, t) ∧
¬TerminatedAt(f, t)
¬HoldsAt(f, t+1) ⇐
TerminatedAt(f, t)
¬HoldsAt(f, t+1) ⇐
¬HoldsAt(f, t) ∧
¬InitiatedAt(f, t)
Inference
Hypergraph
Paths to Clauses
Clause Evaluation
Weight Learning
Data Stream/Training Examples
. . .
Micro-Batch $\mathcal{D}_t$
HappensAt(fast_Slt25(53708), 99)
HappensAt(fast_Ogt55(53708), 99)
HappensAt(slow_Slt15(53708), 99)
HappensAt(slow_Ogt65(53708), 99)
Next(99, 100)
HoldsAt(congestion(53708), 100)
. . .
. . .
Micro-Batch $\mathcal{D}_{t+1}$
HappensAt(fast_Sgt70(53708), 200)
HappensAt(fast_Olt25(53708), 200)
HappensAt(slow_Sgt40(53708), 200)
HappensAt(slow_Olt18(53708), 200)
Next(200, 201)
¬HoldsAt(congestion(53708), 201)
. . .
. . .

Figure 2.1: The procedure of OSLα.

In this section we briefly describe the procedure of OSLα which was presented extensively in the second year deliverable D3.2 Michelioudakis et al. (2015). OSLα extends the procedure of OSL, by exploiting a given background knowledge. Figure 2.1 presents the components of OSLα. The background knowledge consists of the MLN–EC axioms (i.e., domain-independent rules) and an already known (possibly empty) hypothesis (i.e., set of clauses). Each axiom contains *query predicates* HoldsAt $\in \mathcal{Q}$ that consist the supervision and *template predicates* InitiatedAt, TerminatedAt $\in \mathcal{P}$ that specify the conditions under which a CE starts and stops being recognized. The latter form the target CE definitions that we want to learn. OSLα exploits these axioms in order to create mappings of supervision predicates into template predicates and search only for explanations of these template predicates. Upon doing so, OSLα does not need to search over time sequences, instead only needs to find appropriate bodies over the current time-point for the following definite clauses:

$$\texttt{InitiatedAt}(f,\, t) \Leftarrow \text{body}$$
$$\texttt{TerminatedAt}(f,\, t) \Leftarrow \text{body}$$

At any step t of the online procedure a training example (micro-batch) $\boldsymbol{\mathcal{D}}_t$ arrives containing simple derived events (SDEs), e.g. a card transaction happens having amount between 200 and 300 euros. $\boldsymbol{\mathcal{D}}_t$ is used together with the already learnt hypothesis to predict the truth values y_t^P of the composite events (CEs) of interest. This is achieved by (maximum a posteriori) MAP inference based on LP-relaxed Integer Linear Programming Huynh and Mooney (2009). Given $\boldsymbol{\mathcal{D}}_t$ $\texttt{OSL}\alpha$ constructs a hypergraph that represents the space of possible structures as graph paths. Then for all incorrectly predicted CEs the hypergraph is searched, guided by $\texttt{MLN–EC}$ axioms and path mode declarations Huynh and Mooney (2011) using relational pathfinding Richards and Mooney (1992) up to a predefined length, for definite clauses explaining these CEs. The paths discovered during the search correspond to conjunctions of true ground atoms and are generalized into first-order clauses by replacing constants in the conjunction with variables. Then, these conjunctions are used as a body to form definite clauses using as head the template predicate present in each path. The resulting set of formulas is converted into clausal normal form and evaluated. The weights of the retained clauses are then optimized by the AdaGrad online learner Duchi et al. (2011). Finally, the weighted clauses are appended to the hypothesis $\boldsymbol{\mathcal{H}}_t$ and the procedure is repeated for the next training example $\boldsymbol{\mathcal{D}}_{t+1}$.

2.1 Experimetal Evaluation

In this section we evaluate $\texttt{OSL}\alpha$, our recently developed online structure learner, in the domain of fraud management. The aim of the experiments is to assess the effectiveness of our learner in learning both the structure and their parameters used for recognizing CEs, based on imperfect CE definitions and in the presence of incomplete narratives of SDEs. We demonstrate $\texttt{OSL}\alpha$, in the fraud management use case by using the synthetic transaction data containing only positive examples provided by Feedzai. The fraud occurrences in this dataset are produced using a set of predefined fraud patterns. Examples of such patterns include the “increasing/decreasing amounts” pattern, where fraudulent behavior is inferred if a number of consecutive transactions occur for a particular card, where the amount of each transaction is respectively larger/smaller than the amount of the previous one. In this task, the aim is to recognize fraudulent transactions, by exploiting card transaction sequences to essentially learn the patterns that produced the synthetic data. The dataset comprises 30000 transactions (5.3MiB) and 2079 timepoints, where each timepoint represents a distinct card transaction and is annotated either fraud or not. These transaction information constitute the simple derived events (SDEs) that concern activity on individual cards. Finally, we also attempted to learn from a similar synthetic dataset generated using the same fraud patterns but including non-fraudulent sequences (negative examples) of transactions. As we describe at the end of the experimental results $\texttt{OSL}\alpha$ was unable to learn qualitative rules.

2.2 Learning Challenges

The learning problem at hand is characterized by a set of learning challenges that render $\texttt{OSL}\alpha$ unable to learn qualitative fraud patterns using the Event Calculus formalism. The first challenge is the absence of negative examples in the training data, rendering $\texttt{OSL}\alpha$ unable to discriminate the qualitative rules and render the use of inertia, that Event Calculus incorporates, harmful because we $\texttt{OSL}\alpha$ cannot learn termination conditions without negative examples. Secondly, $\texttt{OSL}\alpha$ currently cannot learn sequences using Event Calculus because it does support learning hierarchical definitions. In order to tackle the aforementioned problems we removed the Event Calculus formalism as background knowledge from OSLα (and therefore the law of inertia) and learned simple, short time sequences of transactions patterns that trigger a fraud event for a specific card.

2.3 Experimental Setup

The data are stored in a PostgreSQL database and the training sequence for each micro-batch is constructed dynamically by querying the database. A set of predicates are used to discretize the numerical data (transaction amounts) and produce input events such as, for instance $\texttt{High\_Amount}(\texttt{b67a9a5}, 100)$, representing that the transaction amount for a particular card is between 200 and 300 euro at timepoint 100. The CE supervision indicates when a fraudulent transaction is present. Each training sequence is composed of input SDEs and the corresponding CE annotations (Fraud). The total length of the training sequence consists of 2079 timepoints. We consider only SDEs for amounts. The evaluation results are presented in terms of F1 score. All reported statistics are micro-averaged over the instances of recognized CEs using 10-fold cross validation over the entire dataset using 20 timepoint micro-batches. At each fold, an interval of 270 timepoints was left out and used for testing. The experiments were performed in a computer with an Intel i7 4790@3.6GHz processor (4 cores and 8 threads) and 16GiB of RAM, running Ubuntu 16.04.

2.4 Experimental Results

Table 2.1 presents the evaluation results for OSLα for learned patterns of length 3 and 4. The predictive accuracy of the learned model is very high and almost identical for both cases, while the training time for the shorter patterns decreases considerably. The results indicate that patterns having length 3 are sufficient to learn useful rules for this particular dataset, that is due to the absence of negative examples.

Max Length	**Precision**	**Recall**	**F_1 score**	**Avg. training time (minutes)**
3	0.9950	0.8282	0.9040	2
4	0.9950	0.8296	0.9048	88

Table 2.1: Accuracy and training time.

Below we present a selection of the rules learned by OSLα in order to give an intuition about the training data and the learner's ability to achieve such high accuracy in this particular dataset. The weight in front of each rule indicate the rule's confidence. Rule (2.1) captures the pattern small amount before big amount which is a small sequence consisting of only a pair of events, while rules (2.2) and (2.3) capture the larger sequential patterns for flash attacks or increasing/decreasing amounts. Note that the learner can capture this type of patterns by using only a subset of the events appearing in a large sequence of amounts. That is because there are no examples of sequences of transactions for the same card that are not fraudulent. Therefore OSLα learns that if there is a small sequence of amounts in a short period of time then probably is a fraud regardless if the sequence has more transactions.

$$0.49\ \texttt{Fraud}(card,\ t) \Leftarrow \\ \texttt{Massive\_Amount}(card,\ t) \wedge \texttt{Tiny\_Amount}(card,\ t-1) \tag{2.1}$$

$$0.63\ \texttt{Fraud}(card,\ t) \Leftarrow \\ \texttt{Very\_High\_Amount}(card,\ t) \wedge \texttt{Very\_High\_Amount}(card,\ t-1) \tag{2.2}$$

$$0.37\ \texttt{Fraud}(card,\ t) \Leftarrow \\ \texttt{High\_Amount}(card,\ t) \wedge \texttt{Very\_High\_Amount}(card,\ t-1) \wedge \\ \texttt{Very\_High\_Amount}(card,\ t-2) \tag{2.3}$$

Finally, we also ran experiments for varying batch sizes using maximum pattern length 3 as it achieves identical F1 score to length 4 and is much faster (see Table 2.1). Figure 2.2 presents the F1 score and average batch processing time change as the batch size increases. Note that, as expected, there are minor changes in accuracy by having a different batch size. The reason behind the fluctuations in accuracy for different batch sizes is the fixed window size of timepoints that each micro-batch contains, because some fraud sequences may split into different batches. As expected the average batch processing time increases exponentially as the batch size increase due the larger amount of data that the learner has to process.

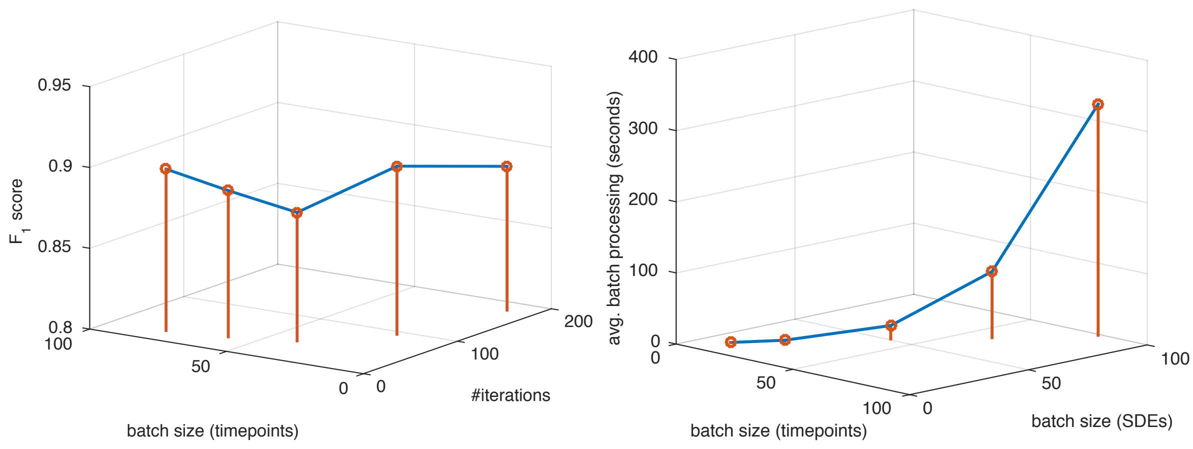

Figure 2.2: F1 score (left) and avg. batch processing time (right) for $\texttt{OSL}\alpha$. In the left figure, the Y axis shows the number of learning iterations.

The experimental evaluation on the synthetic data containing only positive examples clearly shows that $\texttt{OSL}\alpha$ can achieve high accuracy by only learning short sequences of events, but it also witness the exponential growth in computational complexity for larger sequences (see Table 2.1). Therefore, in the presence of negative examples, were the learner should discover patterns of much larger sequences in order to discriminate between fraudulent and non-fraudulent sequences $\texttt{OSL}\alpha$ fails to meet the requirement. As stated in the beginning of our experimental evaluation, we attempted to run experiments, using $\texttt{OSL}\alpha$, in a synthetic dataset containing non-fraudulent sequences. The experiments resulted in yielding very high running times and rendering us unable to present any quantitative result.

3

OLED: Online Learning of Event Definitions

Fraudulent behaviour often manifests in the form of relations between different transactions for a particular card, temporal or otherwise. For instance, fraud analysts tell us that several types of temporal relations, such as a large-amount transaction that follows after a small-amount one within a small period of time, are strong indications of fraudulent behaviour. Such relations between entities may be expressed by means of logic programming (Paschke and Bichler, 2008; Artikis et al., 2010). As an illustrating example, we present below two fraudulent behaviour patterns represented as logical rules.

$$\begin{aligned}&\mathit{fraud}(\mathit{CardId}, T_2) \leftarrow\\&\quad \mathit{transaction}(\mathit{CardId}, \mathit{massive\_amount}, T_2),\\&\quad \mathit{transaction}(\mathit{CardId}, \mathit{tiny\_amount}, T_1),\\&\quad \mathit{before}(T_1, T_2),\\&\quad \mathit{within}(T_1, T_2, 1)\end{aligned} \tag{3.1}$$

$$\begin{aligned}&\mathit{fraud}(\mathit{CardId}, T_2) \leftarrow\\&\quad \mathit{transactionsAtLeast}(\mathit{CardId}, 6),\\&\quad \mathit{within}(T_1, T_2, 7),\\&\quad \mathit{before}(T_1, T_2)\end{aligned} \tag{3.2}$$

The "$\leftarrow$" connective separates the *head* of the rule (left-hand side) from the *body* of the rule. The head consists of a single condition, represented by a logical atom, which is the consequent of the rule, while the body consists of a set of conditions whose conjunction, denoted by commas, is the antecedent of the rule. Informally the semantics of such rules is that the head of a rule is true whenever the conjunction of the conditions in its body are true. We follow Prolog's notation whereby predicates (e.g. *fraud*, or *transactions*) start with a lower-case letter and variables (e.g. *CardId*, or T_1) start with an upper-case letter. Rule (3.1) expresses the so-called "big-after-small" fraud pattern, stating that fraud is detected at time T_2 for CardId, if a $\mathit{massive\_amount}$ transaction for that card occurs at time T_2, a $\mathit{tiny\_amount}$ transaction occurs at some earlier time T_1, and the temporal interval between T_1 and T_2 is less than 1 minute (see $\mathit{within}(T_1, T_2, 1)$ in Rule (3.1)). Rule (3.2) is a version of the so-called "flash-attack" pattern. It states that fraud is detected at time T_2 for card CardId if at least 6 transactions for that card occur within 7 minutes. A common feature of both rules, therefore, concerns the need to represent the temporal relations between transactions.

Typically, in credit card fraud detection relations are propositionalized in a data preprocessing step, and are “injected” in the data in the form of propositional features. An alternative approach, which is the one we follow in this work, is to use methods that allow to express and reason with relations, and attempt to learn relational patterns of fraudulent behaviour.

To this end, we use Inductive Logic Programming (ILP) (De Raedt, 2008), a machine learning approach that uses logic programming as a unifying language for the representation of training examples, background domain knowledge and learnt hypotheses. It is therefore well-suited for the task of learning relational patterns. Given a set of positive and negative examples (logical facts) and, possibly, a logical theory that expresses some existing knowledge about the domain, the goal of an ILP algorithm is to learn a logical theory (called *hypothesis*), which, along with the background knowledge, *covers* — logically entails — the positive examples and does not cover the negative ones. In practice, the requirement for a hypothesis that perfectly discriminates between positive and negative examples is relaxed, to account for noise, and instead various heuristics are used, guiding the search towards a hypothesis with a good fit in the data.

In addition to its relational nature, the temporal nature of fraudulent behaviour detection allows to treat this task as an event recognition problem (Cugola and Margara, 2012). Event recognition systems process sequences of *simple, derived events*, such as sensor data, and recognize *complex (composite) events* of interest, that is, events that satisfy some temporal (and possibly spatial) pattern. In the credit card fraud management, a simple event is the occurrence of a transaction in time, while a complex event is the occurrence of fraudulent behaviour, defined via temporal (and spatial) relations between individual transactions. Event recognition applications deal with data streams, that is, continous flows of information. This poses an extra challenge for machine learning, since learning from such streams requires highly efficient, online algorithms that are able to construct decision models with a single pass over the training data (Gama and Gaber, 2007; Gama, 2010).

3.1 Online Inductive Logic Programming

Frameworks for online learning are under-explored in ILP. Most ILP systems assume a batch setting, where all data is typically in place when learning begins. Alternatively, some ILP systems are capable of theory revision (Esposito et al., 2000), that is, they accept examples over time and gradually alter previously hypotheses to fit new evidence. Still, such systems need multiple scans of the data to optimise their theories.

To handle the volume and velocity of training data in the fraud domain, we use OLED (Online Learning of Event Definitions) (Katzouris et al., 2016), an online relational learner designed for constructing complex event patterns in single pass over the training data. OLED uses a search heuristic based on a statistical significance test that allows to learn such event patterns using only a small subset of the data; OLED relates the size of this subset to a user-defined confidence level on the error margin of not making a (globally) optimal decision during learning. Below we discuss OLED in more detail.

ILP learners usually employ a divide-and-conquer strategy: Starting with an empty set of rules, a rule that covers a subset of the examples is constructed, and this process is repeated recursively until all examples are covered by some rule, or some stopping criteria are met. Each individual rule is constructed in a top-down fashion, starting from an overly general rule (such as a rule with an empty body), and gradually *specializing* it, that is, adding extra conditions to its body. The process is guided by some heuristic function G that assesses the quality of each specialisation *on the entire training set*. At each step, the condition (or set of conditions) with the optimal G-score is selected and the process continues until certain criteria are met, at which point the rule is considered to be complete and is not further specialised.

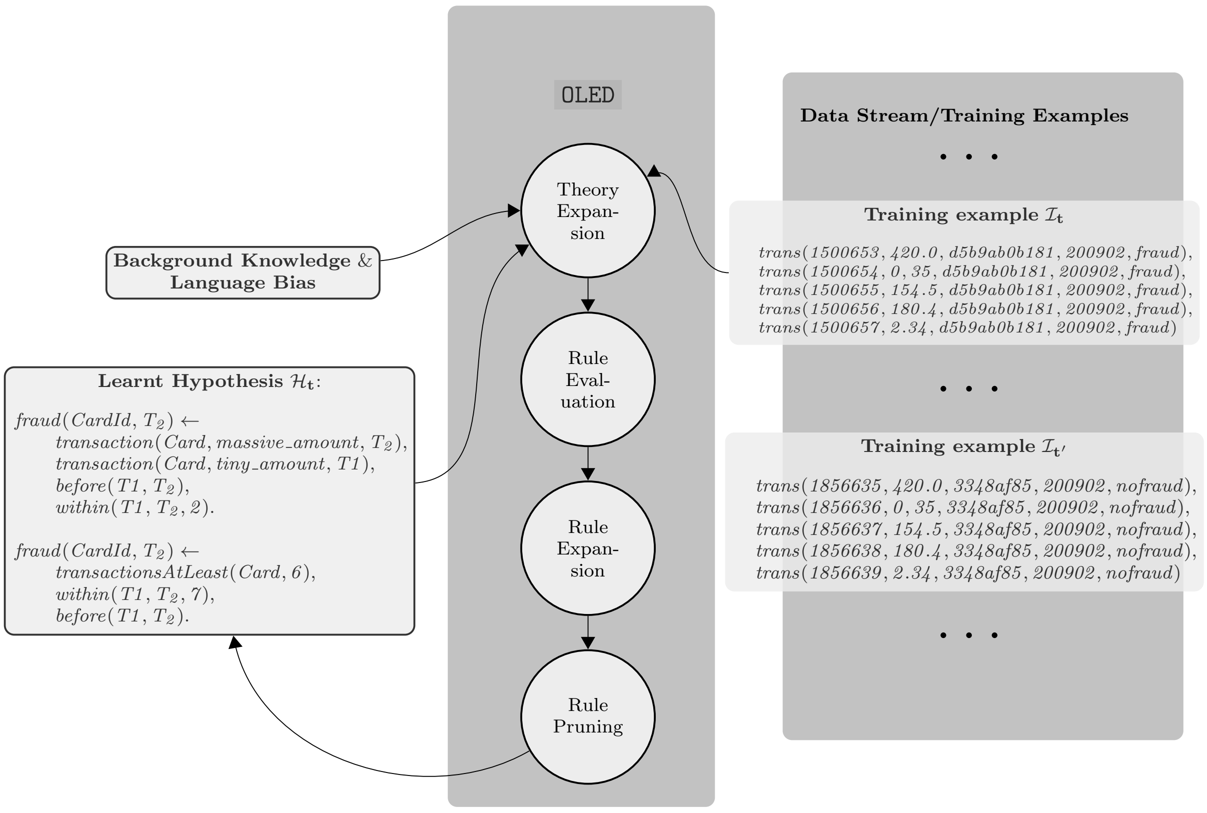
OLED
Theory Expansion
Rule Evaluation
Rule Expansion
Rule Pruning
Background Knowledge & Language Bias
Learnt Hypothesis $\mathcal{H}_t$:
$fraud(CardId, T_2) \leftarrow transaction(Card, massive\_amount, T_2), transaction(Card, tiny\_amount, T1), before(T1, T_2), within(T1, T_2, 2).$
$fraud(CardId, T_2) \leftarrow transactionsAtLeast(Card, 6), within(T1, T_2, 7), before(T1, T_2).$
Data Stream/Training Examples
Training example $\mathcal{I}_t$
$trans(1500653, 420.0, d5b9ab0b181, 200902, fraud),$
$trans(1500654, 0, 35, d5b9ab0b181, 200902, fraud),$
$trans(1500655, 154.5, d5b9ab0b181, 200902, fraud),$
$trans(1500656, 180.4, d5b9ab0b181, 200902, fraud),$
$trans(1500657, 2.34, d5b9ab0b181, 200902, fraud)$
Training example $\mathcal{I}_{t'}$
$trans(1856635, 420.0, 3348af85, 200902, nofraud),$
$trans(1856636, 0, 35, 3348af85, 200902, nofraud),$
$trans(1856637, 154.5, 3348af85, 200902, nofraud),$
$trans(1856638, 180.4, 3348af85, 200902, nofraud),$
$trans(1856639, 2.34, 3348af85, 200902, nofraud)$

Figure 3.1: An illustration of OLED's learning strategy.

To adapt this strategy to an online setting, we use the Hoeffding bound (Hoeffding, 1963). Assume that X is a random variable whose mean value over a stream needs to be approximated. Given a parameter δ, the Hoeffding bound states that the true mean of X over the stream may be approximated by the one obtained after n observations from the stream, with probability $1 - \delta$, and within an error margin ϵ that decreases with the number of observations n, that is, larger values for n yield better approximations. OLED adapts the generic ILP strategy mentioned above as follows: First, it relates the random variable X of the Hoeffding bound principle to the rule evaluation function G, that indicates the best specialisation of a rule out of a pool of candidates. (The evaluation function that we use in this work will be discussed shortly.) This is done by simply comparing specialisations' scores as new examples stream-in, and by monitoring the mean value $\bar{X}$ of the G-score difference between the best and the second-best rule at each time. Second, we use the Hoeffding bound to approximate variable X using only n training examples from the input stream, instead of using the entire training set to find the specialisation with the optimal G-score. The value of n depends on the error that one is willing to tolerate, of not selecting the truly (globally) optimal specialisation at some step. Each training example is processed once, to extract the necessary statistics for the computation of the G-score of candidate specialisations and is subsequently discarded. This gives rise to a single-pass data strategy.

3.2 The OLED system

In this section we briefly discuss the main functionality of OLED. Its overall strategy is presented in Figure 3.1. The input consists of a stream of examples. Each example consists of a set of transactions and it is obtained via windowing, that is, by joining together transactions that fall within a "slice" of time, such as 15 minutes, or one hour of transactions. In Figure 3.1, a transaction is represented by a predicate *trans*, which carries various attributes of the transaction. Only a small number of transaction attributes are depicted in Figure 3.1 for illustration purposes, namely the time-stamp of the transaction, its amount, card id and expiration date, as well as a label (*fraud/nofraud*). In addition to the stream of training examples, input to OLED is also some background knowledge and language bias. The former consists of definitions for auxiliary predicates that are necessary during learning, such as the predicates before, within, or $\mathit{transactionsAtLeast}$, appearing in Rules (3.1) and (3.2) above. The latter (language bias) defines the predicates' payloads.

Learning begins with an empty hypothesis $\mathcal{H}$. On the arrival of new training examples, OLED either expands $\mathcal{H}$, by generating a new rule, or it tries to expand (specialise) an existing rule. A new rule is added whenever some positive example is not covered from any of the existing rules in $\mathcal{H}$, while it is specialised when it covers a large number of negative examples. Rules of low quality (low G-score) are pruned away, after they have been evaluated on a sufficient number of examples. Each incoming example is processed once, to extract the necessary statistics for rule evaluation. OLED's main operations are discussed next in more detail.

Theory Expansion. Theory expansion consists of the addition of a new rule to theory $\mathcal{H}$, in order to cover an example $\mathcal{I}$. A new rule is generated in a data-driven fashion, by constructing a *bottom rule* from $\mathcal{I}$ (De Raedt, 2008). A bottom rule $r_\perp$ is a *most-specific* rule that covers the example $\mathcal{I}$, that is, a rule that contains in its body the maximum number of conditions that are true with respect to $\mathcal{I}$. These conditions are derived from $\mathcal{I}$ and the background knowledge. The goal is to use $r_\perp$ as a search space to obtain a rule r with a good fit in the data. This is done by searching within the space of candidates, that is, rules that result by combining different conditions from the bottom rule, to find a rule that covers a large number of positive and a small number of negative examples. For instance, if the bottom rule is

$$\begin{aligned}
&\mathit{fraud}(\mathit{CardId}, T_2) \leftarrow \\
&\quad \mathit{transaction}(\mathit{CardId}, \mathit{massive\_amount}, T_2),\\
&\quad \mathit{transaction}(\mathit{CardId}, \mathit{tiny\_amount}, T_1),\\
&\quad \mathit{transaction}(\mathit{CardId}, \mathit{medium\_amount}, T_3),\\
&\quad \mathit{before}(T_1, T_2),\\
&\quad \mathit{before}(T_3, T_2),\\
&\quad \mathit{before}(T_3, T_1),\\
&\quad \mathit{within}(T_1, T_2, 1)\\
&\quad \mathit{within}(T_3, T_1, 10)
\end{aligned} \tag{3.3}$$

then any rule with the same head as that of Rule (3.3) and a body consisting of a subset of conditions from its body, is a candidate in the search space defined by this bottom rule. Below are two candidate examples:

$$\begin{aligned}
&\mathit{fraud}(\mathit{CardId}, T_2) \leftarrow \\
&\quad \mathit{transaction}(\mathit{CardId}, \mathit{massive\_amount}, T_2)
\end{aligned} \tag{3.4}$$

$$\begin{aligned}
&\mathit{fraud}(\mathit{CardId}, T_2) \leftarrow \\
&\quad \mathit{transaction}(\mathit{CardId}, \mathit{tiny\_amount}, T_1),\\
&\quad \mathit{before}(T_1, T_2)
\end{aligned} \tag{3.5}$$

Formally, the search space is structured by *θ-subsumption* (De Raedt, 2008) and the search for a good rule is guided by the evaluation function G the assesses the quality of candidate rules. The search space may be traversed either in bottom-up fashion, starting from the bottom rule itself and searching for more general rules, or in a top-down fashion, that is, starting from an overly general rule and gradually searching for more specific ones. OLED follows the latter strategy. Therefore, theory expansion consists of the addition of a rule r with an empty body to theory H. From that point on, r is gradually specialised by the addition of extra conditions from the bottom rule to its body, as discussed below.

Rule Evaluation. Each rule and all of its candidate specialisations are constantly evaluated to assess their quality over the input stream. Towards this, OLED calculates statistics for each rule and its specialisations, cumulatively, as new training examples arrive. These statistics are used for calculating a rule's score via the evaluation function G. Any rule evaluation function may be plugged in OLED — see (Fürnkranz et al., 2012) for an overview of such functions. In this work, we use precision as the rule evaluation function. Therefore, the statistics collected for each rule consist of the true positive and false positive example counts, which are updated each time a new training example arrives.

Rule Expansion. This is the process of specialising a rule r by adding conditions to its body from a bottom rule. As mentioned earlier, we use the Hoeffding bound to select among competing specialisations of a rule. A rule r is expanded, that is, replaced by its best-scoring (so far) specialisation from its pool of candidates, when a sufficient number n of examples has been observed, where n is determined by the Hoeffding bound-based search heuristic. To ensure that no rule r is replaced by a specialisation of lower quality (lower G-score), r itself is also considered as a potential candidate along with its specialisations from the pool of candidates. This ensures that expanding a rule to its best-scoring specialisation after n examples is better, with probability $1 - \delta$, than not expanding it at all.

OLED features a tie-breaking mechanism that allows to handle situations where two or more specialisations turn out to be equally good. We refer to (Katzouris et al., 2016) for more details.

Rule pruning. Often, bad rules may be constructed, whose quality cannot be further improved. OLED does not attempt to *generalise* a rule, that is, remove conditions from its body in an effort to improve its quality due to complexity reasons. Keeping these rules in the hypothesis $\mathcal{H}$ and constantly evaluating them on new examples may be pointless and wasteful. Therefore, OLED supports the removal of rules whose score is smaller than a quality threshold S_{min}. Note that a rule that is actually good (over the entire stream) may score poorly on a sequence of incoming examples. To address this issue, we also use the Hoeffding bound to determine the number of examples n that suffice to derive the conclusion, with probability $1 - \delta$ and within an error margin ϵ, that the quality of the rule is indeed below S_{min}.

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

3.3 Experimental Evaluation

To evaluate our approach for automated fraud pattern construction, we used a synthetic dataset created by Feedzai, that is representative of real credit card transaction streams[1]. The dataset includes instances of the following fraud patterns:

- 'Increasing/Decreasing Amounts'. A sequence of transactions using the same card, with an increasing/decreasing amount withdrawn or spent.
- 'Big after Small'. An outstandingly large amount after one or a series of small amounts.
- 'Flash Attack'. A high number of transactions in a very short period of time.

[1]http://speedd-project.eu/data

System	F_1-score	Precision	Recall	Time (min)
OLED	0.830	0.894	0.776	**21**
SC	**0.892**	**0.912**	**0.874**	188

Table 3.1: Performance comparison: OLED vs batch ILP.

- ‘Transactions in Faraway Places’. Two or more subsequent transactions in a short period of time in large distances.
- ‘Card expires’. A transaction occurs too close (e.g. a day before) to a card’s expiration date.

A detailed discussion of these patterns may be found in (Correia et al., 2015).

Positive (fraudulent) examples in the dataset amount to 0.2% of the total number of transactions — the remaining transactions concern non-fraudulent activity. This is in accordance to the positive/negative example ratio found in real transactions streams. This imbalance of positive and negative examples makes the learning task very challenging. An additional challenge is that fraud patterns often consist of long transaction sequences. This intensifies the learning task since the complexity of a rule increases with its length. The dataset consists of $10,000,000$ transactions, which amounts to approximately 200 MBs of data.

In our experiments, the training set was consumed in windows, data batches of a pre-defined time-span, where the length of each window ranged from a few minutes to one day. The goal of our first experiment was to assess the trade-off between efficiency and quality of the outcome. To this end, we compared OLED, performing online learning, to a classic offline (batch) ILP learner, which learns one rule at a time in a standard set cover loop (De Raedt, 2008), requiring several passes over the data. To this end, we implemented such an off-line algorithm. We did not use one of the existing ILP learners for this task, like Aleph[2] or Progol (Muggleton, 1995), because these systems do not support learning from batches, but instead accept training examples in the form of single logical atoms (**?**). Therefore, the complexity of the learning task increases substantially.

We performed 10-fold cross-validation with OLED and the batch ILP algorithm. The experiments were conducted on a Linux computer with 8 Intel i7-4770 cores at 3.40GHz and 16 GBs of RAM. Both OLED and the batch ILP algorithm were implemented in the Scala programming language, using the Clingo answer set solver[3] as the main reasoning component.

Table 3.1 presents the experimental results in terms of accuracy (precision, recall and their harmonic mean) and efficiency. SC denotes OLED’s set-cover-based batch ILP rival. As expected, SC achieved better accuracy; each rule learnt by SC is highly optimised over the entire dataset. On the other hand, SC’s average training time is larger than 3 hours, while OLED learns fraud patterns of comparable quality over the entire dataset in approximately 21 minutes.

In our second experiment, we studied how the quality of the outcome, in terms of F_1-score, and the average processing time per batch are affected in OLED, by varying the batch size. To this end, we conducted experiments with windows of 2.5, 5, 10, 15, 20, 25 minutes. The top graphs of Figure 3.2 present the results. The top-left graph in Figure 3.2 presents F_1-score as a function of batch (window) size in minutes and the total number of batches in the training set. Each F_1-score value is a micro-average obtained from a 5-fold cross-validation. The top-right graph in Figure 3.2 presents the average processing time per batch as a function of the batch size (in average number of transactions and minutes).

[2] http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/aleph
[3] http://potassco.sourceforge.net/

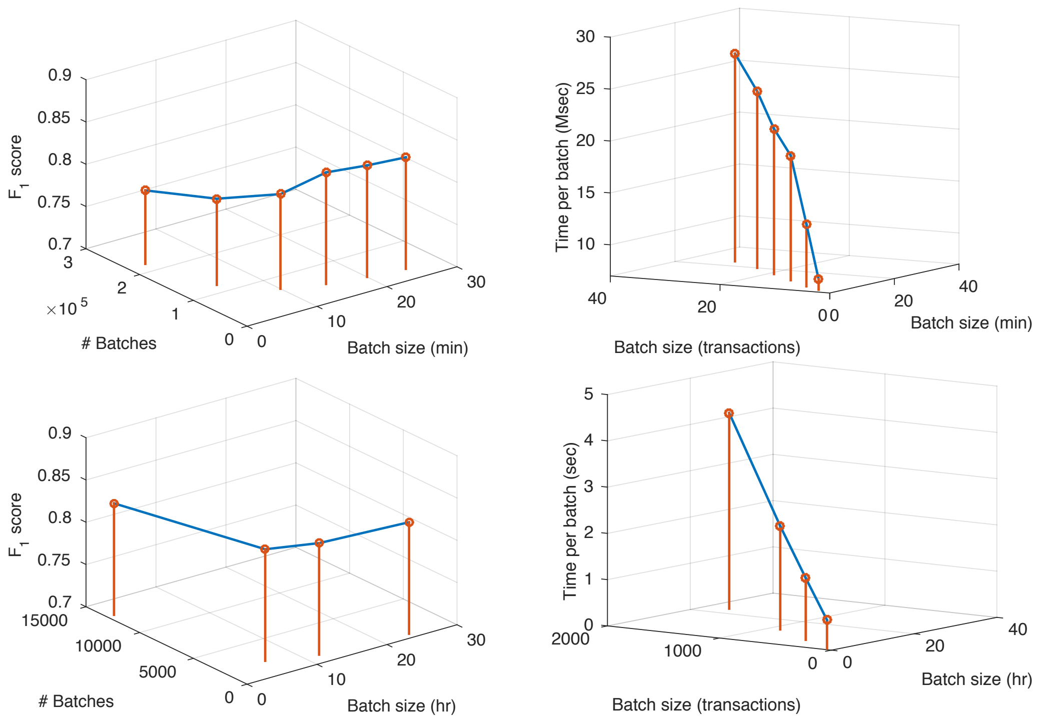

Figure 3.2: F_1 score (left) and average processing time per batch (right) for data batches of 2.5, 5, 10, 15 and 20 minutes (top), and 1, 6, 12 and 24 hours (bottom).

This figure shows that the processing time grows almost linearly with the window size. Regarding predictive accuracy, the results indicate that the F_1 score grows for window sizes up to 15 minutes, while it remains almost constant for larger window sizes. Windows smaller than 15 minutes often contain only part of the transaction sequence involved in a fraudulent activity. Consequently, OLED learns incomplete patterns of lower quality.

To stress-test OLED, we also performed experiments with larger window sizes: 1, 6, 12 and 24 hours. The bottom graphs of Figure 3.2 present the results. The F_1-score remains almost constant for varying window sizes. Furthermore, the processing time still grows linearly with window size.

3.3.1 Experiments with Real Data

We also performed experiments with Feedzai's real dataset for credit card fraud. Learning a model of high quality from this dataset requires to use a large number of specialized features to discriminate between fraud and non-fraud instances. We did not use most of these features, since, due to privacy issues, we had limited access to the dataset and thus we were not able to perform feature engineering, which requires a significant amount of data analysis and experimentation.

We used a summary of the dataset (whose total size is approximately 1 TB) for training with OLED. This summary consisted of 430 MBs of data and it contained the entirety of positive examples from the original dataset, plus a number of negative examples, so that the positive/negative example ratio in the summary was 30/70. Similar data summaries are used by Feedzai for the training of their algorithms.

Our experimental setting was as follows: We used the summary dataset to learn a set of fraud patterns with OLED. We used a 10-fold cross-validation process on the summary, where in each fold we used 90% of the positive and 90% of the negative examples for training, retaining the remaining 10% of positive

Test set size	**Precision**	**Recall**
4.3 GBs	0.758	0.203
43 GBs	0.682	0.203

Table 3.2: Precision and recall for OLED on two test sets of different sizes.

and negative examples for testing. In this process, OLED was able to learn theories that achieved a precision score of approximately 0.95 and a recall score of 0.2 on average, where these scores were obtained by micro-averaging results from each fold. We subsequently evaluated the quality of our learnt theories (the best theory from the 10-fold cross-validation process) on larger test sets, obtained by adding a number of extra negative examples to the summary (recall that the summary already contained the entirety of positive examples for the original dataset).

We created two such testing sets, of size of one and two orders of magnitude larger than the summary dataset (4.3 GBs and 43 GBs respectively). Table 3.2 presents the results, which indicate that OLED learned a small set of relatively good rules, i.e. a set of rules that recognize a small number of true fraud instances (low precision), but also a small number of false fraud instances (satisfactory recall). During both learning and testing, the data were present in windows of length of 1 day. We attempted to learn form smaller windows (15 minutes, 1 hour, 10 hours), but we obtained poor results with much worse precision scores (much larger numbers of false positives) and with no significant gain in recall.

Bibliography

A. Artikis, G. Paliouras, F. Portet, and A. Skarlatidis. Logic-based representation, reasoning and machine learning for event recognition. In *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems (DEBS)*, pages 282–293, 2010. doi: 10.1145/1827418.1827471. URL http://doi.acm.org/10.1145/1827418.1827471.

I. Correia, F. Fournier, and I. Skarbovsky. The uncertain case of credit card fraud detection. In *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems*, pages 181–192. ACM, 2015.

G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44(3):15:1–15:62, 2012. doi: 10.1145/2187671.2187677. URL http://doi.acm.org/10.1145/2187671.2187677.

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

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2021068.

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

J. Fürnkranz, D. Gamberger, and N. Lavrač. *Foundations of rule learning*. Springer Science & Business Media, 2012.

J. Gama. *Knowledge discovery from data streams*. CRC Press, 2010.

J. Gama and M. M. Gaber. *Learning from data streams*. Springer, 2007.

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

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

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

N. Katzouris, A. Artikis, and G. Paliouras. Online learning of event definitions. *TPLP*, 16(5-6): 817–833, 2016. doi: 10.1017/S1471068416000260. URL http://dx.doi.org/10.1017/S1471068416000260.

E. Michelioudakis, A. Skarlatidis, E. Alevizos, A. Artikis, G. Paliouras, N. Katzouris, C. Vlassopoulos, and I. Vetsikas. D3.2: Second Version of Event Recognition and Forecasting Technology – Part II, 2015. URL http://speedd-project.eu/sites/default/files/D3.2_-_Second_version_of_event_recognition_and_forecasting_technology-revised.pdf.

S. Muggleton. Inverse Entailment and Progol. *New Generation Computing*, 13:245–286, 1995. doi: 10.1007/BF03037227.

A. Paschke and M. Bichler. Knowledge representation concepts for automated SLA management. *Decision Support Systems*, 46(1):187–205, 2008. doi: 10.1016/j.dss.2008.06.008. URL http://dx.doi.org/10.1016/j.dss.2008.06.008.

B. L. Richards and R. J. Mooney. Learning relations by pathfinding. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, pages 50–55. AAAI Press, 1992. ISBN 0-262-51063-4. URL http://dl.acm.org/citation.cfm?id=1867135.1867143.